Factibilidad y Diseño de un Sistema de Consultas para el Area de Energía, Industrias y Recursos Naturales no Renovables utilizando un Call Center

Vinicio Bustamante
Patricio Valarezo

29/10/2007

Resumen

El presente trabajo es el producto de una investigación de ingeniería de software que tiene como propósito generar una Solución Informática para disponer de un Callcenter Automatizado que permita ofrecer un servicio de consultas para el Área de Energía Industria y Recursos Naturales No Renovables de la Universidad Nacional de Loja. Se han identificado los procesos académico administrativos pertinentes de ser automatizados, se determinó los procesos de consulta que podrían interactuar mediante una llamada telefónica a un Callcenter Automático.

Haciendo uso de la metodología de desarrollo GRAPPLE y del lenguaje de modelado UML para la documentación de requerimientos, se ha definido las características del Callcenter, se diseño los componentes de software adaptable a los requerimientos. También se ha determinado la Factibilidad Técnica, Económica y Operativa de implementar esta solución.

Se ha planteado el uso del Software Libre como una alternativa válida en el desarrollo de soluciones efectivas y eficientes, el uso de sistemas operativos de última generación, la compartición de código como un "bien" en beneficio de nuestra sociedad y la total apertura al conocimiento informático que es el camino a liberarnos del analfabetismo tecnológico y acortar la brecha digital.

Índice

Índice de figuras

Autoría

El presente trabajo es autoría y responsabilidad de los aspirantes a Ingenieros en Sistemas Lider Vinicio Bustamante Yepez e Iván Patricio Valarezo Lozano. Los autores del presente trabajo se hacen responsables directamente por el contenido publicado en este trabajo. También dan fé de que el presente trabajo cita las fuentes respectivas de autores de los cuales se han tomado algunos contenidos.

Lider Vinicio Bustamante *Iván Patricio Valarezo*

Agradecimiento

Agradecemos a la Universidad Nacional de Loja y al Área de Energía, Industrias y Recursos Naturales No Renovables por los conocimientos impartidos a nosotros como estudiantes durante toda nuestra carrera Universitaria. Agradecemos también al personal Docente y Administrativo del Área de Energía, Industrias y Recursos Naturales No Renovables por su colaboración directa e indirecta en la formación de nosotros como futuros Ingenieros en Sistemas.

Los Autores

Dedicatoria

Dedico el presente trabajo a mi familia, de manera muy especial a mis Padres principales precursores de mi educación, les agradezco por su apoyo e inmenso sacrificio incondicional en mi formación humana y profesional.
Patricio Valarezo

Dedico esta tesis a mi esposa por su incondicional apoyo y especialmente a mis padres quienes han sido la base y guia para mi formación académica.
Vinicio Bustamante

1. Introducción

Introducción: El avance tecnológico y la necesidad natural del ser humano de mantenerse comunicado, unido a la necesidad práctica de contar con servicios automatizados y modernos, que sobrevivan de forma efectiva a los nuevos retos y desafíos tecnológicos, exige concebir soluciones técnicamente factibles y económicas.

En la actualidad, la Universidad Nacional de Loja, particularmente el Área de Energía, Industrias y Recursos Naturales no Renovables, no cuenta con un sistema de consultas automatizado que permita a los usuarios (estudiantes, profesores y público en general) requerir información sobre: carreras que se ofrecen, requisitos de matriculación, duración, módulos, costos, calendario de actividades, calificaciones, asistencias; estadísticas de uso e historial, entre otras.

Existen tres soluciones principales que se podrían implementar para enfrentar automatizadamente el problema enunciado, cada una de las cuales presenta ventajas y desventajas:

1. Acceso a la información por medio de INTERNET
2. Consulta de información mediante el sistema WAP (wireless application protocol)
3. Acceso utilizando un Call Center (Central de llamadas automatizadas)

El acceso a la información mediante un portal de Internet, es una alternativa muy buena, sin embargo, no es aún un servicio masificado en el medio y requiere disponer de un computador con acceso a la red. Para los estudiantes y público en general, particularmente de la provincia de Loja, no se dispone de un efectivo servicio de Internet. Por otro lado, esta solución implicaría acomodar un servidor web[1] dedicado en Internet. Esto implica costos muy elevados de comunicación que hacen que esta solución no sea factible.

El acceso a la información mediante un portal en INTERNET utilizando sistema WAP, es también una opción interesante; sin embargo demanda disponer del servicio de teléfono móvil, pero el servicio WAP en el país a más de ser costoso no está disponible en todas las ciudades. También requiere un teléfono de última tecnología, de elevado costo.

El sistema de consultas utilizando un Call Center automatizado, es una solución informática orientada a facilitar el acceso de los usuarios a información específica, mediante una simple llamada telefónica desde cualquier parte de la provincia o el país, constituyéndose en un medio cómodo, sencillo, económico y práctico. La interfase de comunicación está formada por el sistema telefónico digital y un sistema informático automatizado de recepción de peticiones y consultas. En las condiciones expuestas se consideró pertinente realizar una investigación para responder a los siguientes interrogantes:

[1]Servidor de Páginas HTML y derivados, es el servicio más difundido en Internet.

1. Qué procesos académico administrativos del AEIRNR son pertinentes de ser automatizados en un Call Center?

2. Cuáles deberán ser las características del Call Center y su correspondiente diseño, para satisfacer las necesidades de acceso automático a la información; así como su integración a otros sistemas informáticos existentes?.

3. Cuál será la factibilidad técnica, económica y social de implementar el Call Center Automatizado en el AEIRNR?.

4. Cuáles serán las características de los componentes de software necesarios para el funcionamiento coordinado del Call Center?.

Sobre la base de las interrogantes expuestas, el objetivo general de la investigación fue el siguiente:

Objetivo General:

Generar una solución informática para disponer de un Call Center automatizado que permita ofrecer un servicio de consultas para el Área de Energía, Industrias y Recursos Naturales no Renovables.

Como objetivos específicos se propusieron los siguientes:

Objetivos específicos

- Identificar los procesos académico administrativos del AEIRNR que son pertinentes de ser automatizados en un Call Center.

- Definir las características del Call Center y su correspondiente diseño, para satisfacer las necesidades de acceso automático a la información; así como su integración a otros sistemas informáticos existentes.

- Establecer la factibilidad técnica, económica y social de implementar el Call Center Automatizado en el AEIRNR, de las características diseñadas.

- Desarrollar los componentes de software necesarios para el funcionamiento coordinado del Call Center.

2. Revisión de Literatura

2.1. Evolución de las comunicaciones telefónicas.

Desde su concepción original, se han ido introduciendo mejoras sucesivas tanto en el propio aparato telefónico, como en los métodos y sistemas de explotación de la red (wikipedia enciclopedia web, véase: historia del teléfono).

En lo que se refiere al propio aparato telefónico, se pueden señalar:

- La introducción del micrófono de carbón, que aumentaba de forma considerable la potencia emitida y por tanto el alcance máximo de la comunicación.
- El dispositivo *antilocal*para evitar la perturbación en la audición causada por el ruido ambiente del local donde está instalado el teléfono.
- La marcación por pulsos mediante el denominado disco de marcar.
- La marcación por tonos multifrecuencia.
- La introducción del micrófono de electrete o electret, prácticamente usado en todos los aparatos modernos, que mejora de forma considerable la calidad del sonido.

En cuanto a los métodos y sistemas de explotación de la red telefónica se puede señalar :

- La telefonía fija o convencional que es aquella que hace referencia a las líneas y equipos que se encargan de la comunicación entre terminales telefónico s no portables y generalmente enlazados entre ellos o con la central por medio de conductores metálicos.
- La centralita telefónica de conmutación manual para la interconexión mediante la intervención de un operador/a de distintos teléfonos, creando de esta forma un primer modelo de red.
- La introducción de las «centrales telefónicas de conmutación automática», construidas usando dispositivos electromecánicos, de las que han existido, y en algunos casos aún existen, diversos sistemas (rotatorios, barras cruzadas y otros más complejos).
- Las centrales de conmutación automática electromecánicas, pero controladas por computador.
- Las centrales digitales de conmutación automática totalmente electrónicas y controladas por ordenador, que permiten multitud de servicios complementarios al propio establecimiento de la comunicación (los denominados servicios de valor añadido).
- La introducción de la Red Digital de Servicios Integrados (RDSI) y las técnicas xDSL o de banda ancha (ADSL, HDSL, etc,) que permiten la transmisión de datos a más alta velocidad.

- La telefonía móvil o celular, que posibilita la transmisión inalámbrica de voz y datos, pudiendo ser estos a alta velocidad en los nuevos equipos de tercera generación.

- Existen casos particulares en telefonía fija en los que la conexión con la central se hace por medios radioeléctricos, como es el caso de la telefonía rural mediante acceso celular, en la que se utiliza parte de la infraestructura de telefonía móvil para facilitar servicio telefónico a zonas de difícil acceso para las líneas convencionales de hilo de cobre. No obstante estas líneas siguen siendo lineas fijas.

La telefonía fue influenciada por la era digital. En los laboratorios Bell en 1948, alrededor de la misma fecha el transistor había sido inventado, el matemático Dr. Claude Shannon publicó *Una teoría matemática de Comunicación*, la cual promovió el concepto de comunicación en código binario.(wikipedia enciclopedia web, véase: evolución hacia sistemas automatizados de contestación)

Las publicaciones del Dr.Shannon formaron las bases de toda la revolución de comunicación digital, desde teléfonos celulares hasta Internet.

Le llevó 15 años a AT&T adaptar las ideas del Dr. Shannon. En noviembre 1963, Los laboratorios bell introdujeron el sistema de marcación por tonos, que permitía a las llamadas ser conmutadas digitalmente y luego, permitieron toda clase de menús automatizados y la funcionalidad que eliminó la necesidad de operadores humanos. La nueva red digital inicio el reemplazo de los teléfonos de disco por los de teclado con botones.

A mediados de los años 90, el concepto de «CallerID» (identificador de llamadas), la habilidad de ver un nombre y un número telefónico del llamante en una pequeña pantalla ganó popularidad. Este desarrollo ocurrió casi al mismo tiempo que las máquinas contestadoras digitales, capaces de grabar mensajes en un chip, comenzaron a reemplazar las unidades basadas en cintas magnéticas de hace décadas.

La evolución hacia los servicios personalizados por parte de las empresas, entidades y negocios llevó a la necesidad de inventar sistemas automatizados basados en telefonía (como lo son los callcenters) para mejorar la atención al cliente y dar servicios de valor agregado.

2.2. Definiciones y Terminología.

IVR Interactive voice response system - Sistema interactivo de respuesta de voz

CTI: Computer Telephone Integration (Integración Teléfono/Computador), un sistema que combina sistemas de telefonía con sistemas computarizados de tratamiento de la información.

Call-Center Es una solución que se deriva del concepto de la integración computador-teléfono CTI, es decir la interacción física y funcional entre un sistema telefónico y un sistema informático que facilita el intercambio de información (asterisk www.asterisk.org).

PSTN: Public Switching Telephony Network, (Red telefónica pública). Una de las más grandes redes de comunicación extendida alrededor del mundo, la red telefónica pública. Las principales funciones de esta red son las siguientes:

- Distribución: Colecta la voz del emisor y distribuye la voz al receptor sobre el lazo abonado, Maneja señales analógicas sobre un par de cobre, direccionamiento físico de la terminal telefónica.
- Codificación: La central telefónica que colecta la voz trasforma esta señal analógica en una señal digital, Digitalización PCM(G.711) con un ancho de banda de 4Khz 8000 muestras/seg. y 8 bits por muestra =>64Kbps.
- Conmutación: Encuentra una ruta entre los dos abonados, Uso de la conmutación de circuitos, Una vez que se establece la conexión, se tiene un circuito de transmisión bidireccional dedicado entre los dos abonados, Este circuito tiene un ancho de banda de 300-3400 Hz y presenta un retardo de propagación constante. Para encontrar un destino existe una red jerárquica a nivel mundial, la codificación esta formada por 3 dígitos de código de país y 10 dígitos para el número local.
- Transmisión: Transporta la señal de voz codificada, Jerarquías de multiplexaje, La red telefónica fue creada para transportar una señal analógica. Ofrece un ancho de banda de 3400-300 Hz a cada canal. Los módems permiten intercambiar información digital sobre un medio analógico

PBX: Private Branch-Excange - Conmutador tradicional de voz.
(PBX Info www.pbxinfo.org) Funcionalmente, el PBX realiza estas tareas principales:

- Establecer conexiones (circuitos) entre un set de equipos telefónicos de dos usuarios. Tome en cuenta que maquinas de fax, módems y otros dispositivos de comunicación pueden estar conectados al PBX (a pesar de que el PBX podría degradar la calidad de la señal para los módems). Por eso los sets telefónicos se los llama extensiones.
- Mantener las conexiones por el tiempo que el usuario las requiera.
- Proveer información hacia el departamento de cuentas (por ejemplo para la contabilidad y control de llamadas).

PABX: Conmutador Telefónico Automático.

Modem: Módem es un acrónimo que proviene de la palabra inglesa modem, (construida a partir de modulator/demodulator).Un módem es un modulador y desmodulador de datos que participa en una comunicación como ETCD[1] Su uso más común y conocido es en transmisiones de datos por vía

[1]ETCD: Equipo terminal del circuito de datos. Un ETCD es todo dispositivo que participa en la comunicación entre dos dispositivos pero que no es receptor final ni emisor original de los datos que forman parte de esa comunicación.

telefónica. Como los computadores procesan datos de forma digital y las líneas telefónicas de la red básica sólo transmiten datos de forma analógica, los módem lo que hacen es transformar mediante diferentes técnicas las señales digitales en analógicas y viceversa. Hay distintos tipos de módem en función de la velocidad de transmisión que alcancen: 14.400, 28.800, 36.600 y 56.600 bps son las velocidades más comunes actualmente. Otro módem menos conocido es el chip DSP que tenemos todos los que usamos una tarjeta de sonido; este chip se encarga de modular y desmodular las señales que le llegan en forma digital y transformarlas en señales analógicas que tal vez después de unos filtros lleguen a los altavoces de nuestro Computador. Existen, además, módems DSL (Digital Subscriber Line), que utilizan el espectro no-audible del cable telefónico, y permiten alcanzar velocidades mucho mayores que las de los módems analógicos o tradicionales. También poseen otras cualidades, como la posibilidad de establecer una comunicación telefónica por voz al mismo tiempo que se envían y reciben datos.

Switchoard: (intercambiador telefónico) es un dispositivo usado para conectar manualmente un grupo de teléfonos de uno a otro o hacia una conexión externa. El usuario de un switchboard es conocido como un operador. Los Switchboards entraron en uso poco después de la invención del teléfono en 1876. Los poblados pequeños tenían típicamente un operador de intercambiador telefónico instalado en la casa del operador, de esta manera esta persona podía contestar llamadas las 24 horas del día. Con el paso del tiempo se han ido reemplazando estos dispositivos por alternativas automatizadas como el TSPS (Traffic Service Position System) un sistema inventado por los laboratorios Bell para reemplazar a los tradicionales SwitchBoards alambricos.

2.3. Elementos del Sistema/Software.

En la configuración de un Centro de Llamadas (i.e. Call Center) se pueden distinguir claramente los componentes siguientes:

1. Central Telefónica (PBX, Private Branch Exchange). Central de conmutación de llamadas telefónicas.

2. Servidor CTI. Es un middleware[4] que hace las funciones de controlador principal de todos los componentes hardware y software del Call Center. Es este servidor el que, por ejemplo, imparte órdenes para el envío de información solicitada, o almacena y estructura la información para los diferentes reportes de operación que se requiera.

3. Servidores de Bases de Datos. Repositorios de la información de los clientes de una organización y un conjunto adicional de información que la institución genera.

[4]MiddleWare - Hardware/Software intermedio entre dos sistemas análogo/digitales para su integración

4. Sistema Interactivo de Respuesta de Voz (IVR, Interactive Voice Response System). Conjunto de hardware y software que se encarga de la gestión de llamadas entrantes (inbound) a una organización. Es éste el sistema que permite y facilita la entrega de mensajes «hablados» a los llamantes de tal forma que éstos puedan acceder a la información residente en las bases de datos de las organizaciones. El IVR es el elemento al cual se le asignan los trabajos de suministro de información rutinaria y en el caso del AEIRNNR la información alimentada desde otras Bases de Datos (Ej. Notas y Asistencias).

Con base en lo anterior, se puede definir a un Call Center como una solución informática orientada a facilitar el flujo de información entre clientes y entidades tanto de negocios como laborales, suministrando información a los llamantes en un ambiente personalizado.

2.4. Elementos del Sistema/Hardware.

La implementación del Call Center, al ser una solución que combina diferentes tecnologías[5], Necesita crear el puente que vincule todas ellas para convertirlas en conjunto en la solución que se desea ofrecer. Hablando estrictamente del hardware (entiéndase por hardware a equipos electrónicos, pseudo-electrónicos y eléctricos específicos para cada tecnología) el desarrollo, construcción y puesta en marcha del Call Center tiene como principales elementos de hardware los siguientes:

1. Central PBX: Existen nuevos equipos que soportan las últimas tecnologías como:
   - VOIP[6]
   - Direct Dialing (DDD or DDI): Discado directo
   - Customised Abbreviated dialing (Speed Dialing): Discado Rápido customizable
   - voice mail: Correo de voz (casillero estilo maquina contestadora)
   - Follow-me : Redireccionamiento de llamadas
   - Call forwarding on busy : Redireccionamiento de llamadas en estado ocupado
   - Call transfer: Transferencia de llamadas
   - Music on hold: Música en espera

[5]Entre otras tecnologías, se menciona las principales: Telecomunicaciones, Bases de datos, Desarrollo de aplicaciones,Electrónica y control.

[6]VOIP(Voz sobre IP): El estándar VOIP fue definido por la ITU(International Telecommunication Union) en 1996, VoIP proporcionaría a las delegaciones de una misma empresa, comunicaciones gratuitas entre ellas. No solo entre sus delegaciones, sino entre proveedores, intermediarios y vendedores finales, las comunicaciones se podrían realizar de forma completa ente gratuita. Además, la red de comunicaciones de la empresa se vería enormemente simplificada, ya que no habría que cablear por duplicado la red, debido a que se aprovecharía la red de datos para voz.(VoIP www.recursosvoip.com)

- Night serviceLog: Servicio de registro nocturno de actividad del PBX

Entre las principales marcas conocidas de este tipo de equipos se tiene: Avaya (fue Lucent y AT&T), Siemens AG (incluido Rolm), NEC, Nortel, Toshiba, Fujitsu, Vodavi, Mitel, Ericsson, Panasonic. Existe también un proyecto OpenSource llamado Asterisk[7]

2. Modem con capacidad VoiceModem: Este equipo se comunicará con la central PBX para atender a las llamadas, el soporte VOICE es indispensable por cuanto a través de este dispositivo se va a inyectar la información codificada a formatos audibles desde otros formatos de intercambio de información, el trabajo lógico automatizado de este proceso es una de las piedras angulares de nuestro proyecto.

3. Host especializado de comunicación (Servidor CTI/IVR): Este host está representado por un computador o conjunto de computadores (dispuestos en topología de un clusters), el requerimiento que determinará el número de equipos a usar esta dada por la carga del sistema. Para el fin específico del área creemos que será conveniente usar un PC de arquitectura i386[8] o Compatible, con una velocidad de 500Mhz o superior, con interfaces de red ethernet a 10/100Mbps. Otro servicio importante que podrá ofrecer este Host es el de servicio de Base de Datos para dar soporte Básico al Call Center.

4. Computador Cliente de alimentación de información: Con este computador se podrá configurar y ajustar los parámetros del *Host especializado de comunicación* haciendo uso de la herramienta de configuración que será construida, se podrá agregar/modificar/borrar la información que el Call Center entrega a los usuarios. Cabe señalar que este computador no necesariamente deberá estar dedicado a esta tarea solamente, un equipo conectado a la misma red del Host será suficiente para actualizar la configuración del Call Center.

2.5. Aplicación del Software Libre como Marco de Desarrollo

El software libre(Abella 2003, véase: definición del software libre) o Free Software[4] es el software que, una vez obtenido, puede ser usado, copiado, modificado

[7]asterisk: http://www.asterisk.org, es un proyecto que tiene como finalidad suplantar a centralitas de llamadas para que operen sobre un PC o algún otro hardware con los elementos especializados, es una opción valida para complementar a nuestro proyecto. (asterisk www.asterisk.org)

[8]i386 Se refiere a la arquitectura Intel de procesadores 386 o superiores, el único requerimiento indispensable de la arquitectura a usar es de que posea slots PCI para alojar el modem voice, el sistema operativo GNU/Debian Linux soporta cerca de 30 arquitecturas.

[4]En el término free Software, tal como está definido por la Free Software Foundation, la palabra free en inglés es ambiguo y puede referirse tanto a la libertad (free speech, libertad de expresión) como a la gratitud (free beer, cerveza gratis). En español no existe tal ambigüedad, distinguiéndose claramente el software libre del software gratis.

y redistribuido libremente. El software libre suele estar disponible gratuitamente en Internet, o al precio de coste de la distribución a través de otros medios; sin embargo no es obligatorio que sea así y, aunque conserve su carácter de libre, puede ser vendido comercialmente.

Una de las muchas ventajas del software libre es de que pone a la disposición de desarrolladores una infinidad de recursos que van desde compiladores, interpretes, IDEs[11], herramientas de documentación de código, librerías, clases y programas ya desarrollados que se podrían usar como conectores para crear aplicaciones, todo esto sin tener que pagar por una licencia, gracias a las licencias OpenSource(Abella 2003, vease:licencia GPL), un desarrollador esta armado con todo lo necesario para ser productivo.

Existen disponibles algunas Bases de Datos OpenSource o con licencias similares que son factibles de usar en este proyecto (o cualquier proyecto), cada cual tiene sus pros y contras, pero al final existe alguna que se ajuste mejor a lo que necesitamos, por nombrar algunas del tipo relacionales tenemos: Postgresql, Mysql, Firebird, Sqlite todas ellas cumplen con el estándar SQL92 y migrar un sistema de una base de datos a otra será relativamente sencillo si se mantiene alineado a este estándar.

2.6. Metodologías para el desarrollo de Proyectos Informáticos

El proceso de construcción de un sistema informático (puesto que el conjunto de piezas de software representado por el Call Center es un sistema informático (Senn 1998)), se deberá ajustar a las metodologías de desarrollo modernas basadas en los procesos usados a lo largo de la evolución de la ingeniería de sistemas. Una metodología de desarrollo es la estructura y naturaleza de los pasos en un esfuerzo de desarrollo, esto nos ayuda a comprender con claridad el problema y a diseñar una solución que satisfaga las verdaderas necesidades de los usuarios principales (stakeholders) (Larman 2001, ver:stakeholders).

Antes de comenzar cualquier esfuerzo de programación y diseño de las soluciones, se necesita que los desarrolladores entiendan con claridad el problema.

Esto requiere que se analicen los requerimientos, una vez hecho esto, alguien tiene que convertir este análisis en diseño de una solución que satisfaga en toda integridad los requerimientos especificados. Luego de cumplir con estas fases, los programadores podrán comenzar a crear código efectivo, el cual deberá ser probado y depurado. Existen algunas metodologías que se han venido empleando para el desarrollo de sistemas, no existe en realidad un «método estándar» de hacer las cosas, y cada metodología presenta sus ventajas y desventajas.

Sin embargo, se debe aclarar que debido a la comodidad y afinidad con las metodologías modernas orientadas a objetos, se ha considerado apropiado tomar

[11] IDE: Integrated development enviroment: Ambiente de desarrollo Integrado, es un programa que integra algunas funcionalidades y herramientas para desarrolladores, es práctico y facilita al desarrollador cumplir con su trabajo, integra herramientas como clientes CVS, auto completado de sentencias, etc, dependen del lenguaje de programación que se use, existen diferentes IDEs dependiendo del lenguaje y también los hay multi-lenguaje.

en cuenta las siguientes:

Waterfall El método en cascada(Ariadnetraining 2001) es relativamente un antiguo método que ha tenido éxito desde hace algunos años, se podría decir que los programadores experimentados todavía hacen uso de este método, junto con otros híbridos modernos, a pesar de ser un modelo antiguo, a juicio nuestro creemos que este método se podría ajustar al modelo de análisis y desarrollo orientado a objetos que perseguimos. este método tiene como filosofía la siguiente: Se subdivide en fases que son: análisis, diseño, codificación y distribución .Estas fases se ejecutan una después de otra comenzando por el análisis.

Una de sus principales desventajas es que los diferentes desarrolladores no trabajan juntos, es decir, un analista en la fase de análisis no podría estar en contacto con un diseñador para compartir puntos de vista importantes puesto que el diseño aun no ha comenzado.

Rational Unified Process
(Proceso unificado de desarrollo) (Larman 2001, véase: UP), es una metodología que adopta como base el ciclo de vida adaptado con un conjunto de cortas iteraciones (similares a las del modelo en Espiral, pero con fases más estrictas), Ha sido madurada a lo largo de los últimos años y ha obtenido favorable acogida sobre todo gracias a los artefactos de análisis y desarrollo que implementa, la gran cantidad de ellos son parte del lenguaje UML Se subdivide también en fases predefinidas que son:

- Inception (principio)
- Elaboration (elaboración)
- Construction (Construcción, es una serie de mini - waterfalls).
- Transition (Transición).

A pesar de ser un estándar libre, existe una gran compañía que ofrece software completo capaz de seguir esta metodología desde el principio hasta el final llamado Rational Rose, sin embargo no todas sus herramientas son libres y para nuestro medio, podrían parecer muy costosas. Cabe señalar que para adoptar esta metodología no es estrictamente necesario usar el software de Rational, es más, creemos que no es ni siquiera necesario el uso de un computador, bien podría usarse simplemente papel y lápiz sin embargo esto puede afectar a la productividad y efectividad del equipo de desarrollo. Una de las principales desventajas del RUP es por desgracia también su ventaja al ser una metodología completa y extensa, se la usa generalmente en proyectos grandes (+18 meses) puesto que si se lleva de una manera estricta, nos asegura el éxito. Se pueden acortar ciertas partes de la metodología que no «encajan» en algunos proyectos.

GRAPPLE: Esta es una de las muchas metodologías *adaptadas del RUP*, que podrían encajar en proyectos más pequeños

(Schmuller 2002, ver GRAPPLE). Esta no es una férrea metodología, es un conjunto de ideas adaptables y flexibles, se podría decir que es un *patrón simplificado en un proceso de desarrollo*. Esta metodología da a un director de proyecto de desarrollo la libertad de adaptar sus propias ideas de manera creativa, esto con la finalidad de acomodar las herramientas y artefactos disponibles de RUP y UML en casi cualquier proyecto.

3. Materiales y Métodos

3.1. La elección de la Metodología de Desarrollo

Como se señaló en capítulos anteriores, existen muchas y diversas metodologías para el desarrollo de sistemas. La RUP es una de las más difundidas debido a las características que ya se mencionaron. Incluso se han venido desarrollando "clones" reducidos del RUP, entre los que se mencionó GRAPPLE.

La elección de la metodología de desarrollo para el presente proyecto, tomando en cuenta que el escoger una metodología de desarrollo es una elección que va de acuerdo a: la conveniencia o no de usar RUP, el dominio del negocio, y preferencias personales, se la ha definido de acuerdo a los siguientes puntos:

1. RUP ha sido descartado porque es una metodología para proyectos de más de un año y medio, también requiere un estricto apego a sus métodos y artefactos, requiere que se cumplan "etapas" y a pesar de que ofrece un estricto control de calidad, relentiza el desarrollo en proyectos de pequeña escala.

2. El método en Cascada, también ha sido descartado por cuanto está orientado a proyectos con etapas secuenciales, y en donde los requerimientos casi no cambian. Nuestro proyecto aborda diferentes puntos en el desarrollo de manera asíncrona, y los requerimientos serán adaptados al desarrollo de pruebas pequeñas conforme se vuelven más complejos los requerimientos.

3. El dominio del negocio, refiriéndose a la forma en la que se manejan los procesos en el AEIRNNR, y tomando en cuenta que nuestra solución es una forma de "automatización alternativa" de un proceso eminentemente manual, el cual no afecta el curso normal en el que se desenvuelven y no es un remplazo de la forma tradicional. Esto nos da la libertad de utilizar una metodología adaptada a pequeños proyectos con la finalidad de "ordenar", "estructurar" y "documentar"los requerimientos mas no la de tomarla como una guía estricta de desarrollo.

GRAPPLE, es la metodología que se ha elegido por las siguientes razones adicionales:

1. Metodos Estándares: Puesto que usa los artefactos del RUP que son entre otros: casos de uso, diagramas UML, iteraciones, etc.

2. Comunicativo: Fomenta la comunicación a todo momento entre los desarrolladores: analistas, diseñadores y programadores. Integra los cambios y los requerimientos "en linea".

3. Es un RUP Simplificado: Rescata un conjunto de ideas adaptables y flexibles.

4. Permite adaptar nuestra propias ideas: Permite a un líder de un grupo de desarrolladores adaptar ideas propias, es por esto que más a llamado a

nuestra atención, al existir diferentes alternativas para una misma solución, al usar el Software Libre como marco de desarrollo, entonces GRAPPLE permite adaptar cambios y mejoras, permite agregar y suprimir artefactos y herramientas UML a conveniencia.

La metodología de desarrollo de software, junto con el lenguaje de modelado UML fueron las herramientas que permitieron estructurar la identificación de los procesos académicos administrativos del AEIRNNR posibles de ser automatizados.

3.2. Estructura del Proceso de Desarrollo GRAPPLE

El proceso de desarrollo GRAPPLE esta conformado por las siguientes fases:

Recopilación de Necesidades: Que comprendió la visita al AEIRNNR, las entrevistas informales con los estudiantes, con el objetivo de determinar los requerimientos, el producto de esta fase está reflejada en los casos de uso y en los diagramas de apoyo. Contiene una comprensión general de lo que "Se quiere hacer".

Análisis: En el análisis se realizó una selección de requerimientos, prioridades, se determino el alcance del proyecto, se obtuvo un vocabulario de términos referentes al proyecto (listas de materias de estudiantes, un snapshot[1] de la Base de Datos de los Sistemas Colaborativos), se identificaron los sistemas cooperativos y se investigó las posibles formas de comunicarnos con ellos. Se aumentó la comprensión del problema, determinamos el API[2] y documentamos los diagramas de base de datos existentes de los sistemas cooperativos. Se hicieron realidad los casos de uso y diagramas de actividades del proceso de negocios de acuerdo a lo evidenciado en la actualidad. Se hizo también la depuración de los diagramas de clases y diagramas de secuencias de acuerdo al avance en el análisis.

Diseño: En esta etapa se construyó y depuró los diagramas de clases, de actividades y diagramas de secuencias. También se planteó dentro del diagrama de casos de uso el diagrama de componentes para dividir la lógica de administración del sistema con la lógica de interacción con el usuario llamante. En esta etapa se crearon aplicaciones para el control y administración de los sistemas así como de herramientas de apoyo para la gestión de datos. En esta etapa se inició la documentación de las características básicas de las herramientas construidas y de los procesos principales de implementación del sistema en el ambiente propuesto. (Instalación de paquetes de software de apoyo: Asterisk, Python, Módulos de Python, etc).

[1] Snapshot de la Base de Datos: Es el congelamiento de datos en un determinado momento de un sistema.

[2] Application Programming Interface: Uno de los principales propósitos de un API es el de describir como las aplicaciones de software y los desarrolladores se deben comunicar, como deben acceder a un set de (por lo general de terceras personas) funciones, por ejemplo una librería, sin necesidad de acceder directamente al código fuente o conocer de forma detallada como la librería realiza su trabajo.

Desarrollo: En esta etapa se desarrollaron las herramientas necesarias para el control del sistema, así como también los módulos que forman parte del todo como sistema. Esta etapa incluyó el diseño del software, el modelo de datos, la aplicación de patrones (Shalloway 2003), programación y codificación de los requerimientos, de acuerdo al análisis. También incluyó la codificación de fonetos audibles necesarios para la interpretación de los resultados de las consultas. Fue de relevante importancia el uso de un sistema de versionamiento de código fuente[3] para mantener las diferentes versiones del sistema, puesto que los requerimientos tienden a cambiar. Se realizaron las pruebas y ajustes necesarios y se creó un prototipo funcional del sistema.

Distribución: En este punto ofrecen todas las instrucciones necesarias y la descripción del hardware específico para el soporte de la solución de acuerdo al aspecto crítico del sistema y del ambiente en el que va a funcionar.

3.3. Diagramas UML

El lenguaje UML sirvió para la documentación requerimientos, el análisis y finalmente, basados en los diagramas de casos de uso, secuencia y actividades, diseñar los componentes de software necesarios para el sistema. Los diagramas que se han confeccionado cumplen con el estándar propuesto por la OMG(Larman 2001) y las especificaciones del lenguaje UML2. Los principales diagramas que se usaron fueron:

- Diagramas de Casos de Uso: Este diagrama se usó para graficar el estado de situación actual del sistema, el diagrama de casos de uso finales y definir los límites del sistema (boundaries), con estos diagramas que se encuentran en A.2, se captó la comprensión general de la solución y se sentaron las bases para la definición de los aspectos de interacción con otros sistemas y la definición de la funcionalidad completa del sistema. El detalle de éstos se diseño en los diagramas de secuencia y de actividades.

- Diagramas de Secuencias: Estos diagramas permitieron definir/verificar la interacción de los objetos/actores/entidades que forman parte de las entradas y salidas del sistema. Nos ayudó a la especificación de objetos, clases y mensajes de interacción entre ellos, se encuentran documentados en A.2.

- Diagramas de Actividades: Estos diagramas se usaron para detallar el flujo de interacción, los pasos que siguen cada una de las operaciones y resultados de un proceso. Nos permitieron afinar y optimizar los algoritmos para la creación de los diferentes componentes del sistema, se encuentran documentados en A.2.

- Diagramas de Componentes: Estos diagramas permitieron dividir lógicamente las operaciones en módulos para ser comprendidos de mejor manera en la etapa de programación.

[3] SVN: Subversion

Todos estos diagramas han sido el principal apoyo al momento de crear los componentes, sirviendo como un "mapa" en donde se detallan los puntos que deben ser cumplidos a nivel de ingeniería de programación para que sean adaptados al lenguaje. No se ciñen a un lenguaje de programación en particular, aunque fundamentalmente al momento de programar deberán ser respetados a cabalidad por los programadores, sí se centran y basan sobre todo en los paradigmas de programación, de los cuales se podría decir que se han usado: Programación Estructurada y Orientada a Objetos, dependiendo del caso particular y la conveniencia por parte del programador y la exigencia del esquema propuesto por los diagramas.

3.4. Herramientas de Modelado de Datos

El modelado de datos es un proceso necesario al momento de crear un sistema, puesto que permite "modelar" el mundo real en un ambiente virtual mesurable. La base de datos de estudiantes es la principal fuente de datos modelada para la creación de los componentes de software. Los diagramas de bases de datos permitieron la normalización (Kline 2001) de la información. Las principales herramientas usadas fueron:

- Diagramas E-R: Para definir las relaciones entre entidades de datos y sus relaciones (Kline 2001), Cardinalidades, Atributos, etc.
- Normalización: Primera, segunda y tercera forma normal, para definición de entidades de datos y su modelamiento.

Estas herramientas sirvieron para comprender la estructura de las bases de datos de los sistemas colaborativos, es de acuerdo a estas definiciones que es posible la interacción de los componentes con la información real del sistema. También sirvieron para la creación de sentencias de manipulación de datos usando el lenguaje SQL(Kline 2001).

3.5. Motores de Bases de Datos auxiliares

Se usó como motor de base de datos sqlite(SQLite database system) un motor de base de datos ligero, Open Source. El motor de base de datos junto con el resto de herramientas han sido integradas con el propósito de crear un prototipo capaz de funcionar en el AEIRNNR y que sirva de apoyo para realizar las pruebas pertinentes.

3.6. Lenguajes de Programación

Python como Lenguaje Multipropósito

Una vez determinados los requerimientos, analizados y proyectados en los diagramas UML, es necesaria la creación de los componentes de software, este proceso paulatino esta coordinado por la metodología de desarrollo, que tiene como

producto final la creación de los "programas" usando un lenguaje conveniente. El lenguaje principal para el desarrollo de las herramientas que se ha usado es el lenguaje Python(Ascher 2003), por las siguientes características:

- Interpretado: No se compila en el sentido estricto de la necesidad de compilación, permite la generación de prototipos rápidamente.
- Sintaxis limpia: Fácil de programar, y de leer, con tipos de datos de alto nivel.
- Paradigma Flexible: Desde el punto de vista del paradigma, Python puede ser usado como un lenguaje estructurado o como un lenguaje orientado a objetos.
- Libre: Es un Lenguaje libre por definición, no se requiere pagar ninguna licencia para hacer uso de éste. Esta liberado bajo la licencia OpenSource aprobada por la OSI.
- Potente: Porque contiene extensiones para cualquier tipo de tarea como: Acceso a Bases de datos, conectores hacia otros modelos de datos, módulos para el tratamiento de operaciones matemáticas complejas, bindings a otros lenguajes de programación, soporte para paradigmas de programación orientada al desarrollo web, etc.
- Multiplataforma: Python corre en plataformas tan diversas y diferentes como: Linux, Windows, Unix, MacOS, Amiga, plataformas embebidas como Palm OS e incluso Teléfonos celulares Nokia.

Bash y Asterisk Scripting

El lenguaje Bash es otro lenguaje interpretado con orientación hacia la manipulación de funciones típicas de un sistema operativo Linux/Unix. Es el intérprete de comandos por defecto en la mayoría de distribuciones de Linux y también puede ser usado como un lenguaje de programación.

El modelo para la definición de interacción con el sistema Asterisk esta comandado por un seudo lenguaje para el procesamiento de instrucciones entendibles por el sistema Asterisk, se basa principalmente en el uso de «ztanzas de instrucciones» llamadas extensiones(Van Meggelen 2005, ver dialplan). Es un sistema de ejecución de instrucciones Top-Down encapsuladas en contextos o "contexts". Este es un sistema eficiente para crear interacciones de cualquier tipo, también pueden formar parte de estas instrucciones las llamadas hacia otros lenguajes de programación.

3.7. Plan de Validación del Software y Pruebas

El plan de validación de software permitió evaluar las diversas características de hardware y demás logística necesaria para la implementación del Callcenter Automatizado. Una prueba del funcionamiento del Callcenter usando el software

desarrollado, usando datos reales levantados de los estudiantes del AEIRNNR. Dentro de este marco, se realizaron las primeras pruebas de funcionamiento del callcenter en un ambiente de características similares a las que se tienen en el AEIRNNR, usando los siguientes equipos:

1. Dos lineas telefónicas provenientes de una central PANASONIC.
2. Dos interfaces FXO modelo T100P clones de las fabricadas por le empresa Digium.
3. Computador Pentium III 700 Mhz Marca Dell con 4 ranuras de expansión PCI, disco duro de 10Gb, 128Mb de memoria RAM.
4. Central telefónica PANASONIC modelo KXTES824LA.
5. Un teléfono VoIP marca Soyo G668.
6. Un Softphone X-lite en un computador Dell portátil con acceso wifi.
7. Un Softphone Twinkle en un PowerBook G4/ Debian GNU/Linux

Las pruebas han sido orientadas a determinar:

- El funcionamiento adecuado del Callcenter con todas las características con las que ha sido diseñado.
- El acceso simultaneo de solicitudes para determinar si el hardware usado cumple con la carga aplicada.
- La satisfacción de los usuarios del sistema y su aceptación.

Adicional a las pruebas de laboratorio, se realizaron pruebas en el AEIRNNR usando un sistema prototipo, los resultados de dichas pruebas están documentadas en la seccion 4.3.

4. Resultados

Después de un análisis de la situación actual y un modelado de acuerdo a los requerimientos iniciales de la solución, se obtuvieron los resultados reflejados en los casos de uso y en los diagramas UML. Estos comprenden un diseño de *como se desea que funcione* el sistema. También se ha podido desarrollar un conjunto de aplicaciones, librerías, esquemas y demás elementos de software que permitieron solventar los requerimientos antes obtenidos (ver diagrama de componentes en el anexo A.2).

4.1. Identificación de los procesos académicos administrativos del AEIRNNR factibles de ser automatizados

Los procesos factibles de ser automatizados del AEIRNNR son los siguientes:

1. Consulta de notas.
2. Consulta de asistencias.
3. Consulta de novedades y anuncios para estudiantes.
4. Consulta de requisitos para matriculación.

Tradicionalmente las consultas de notas y asistencias se las realiza de la siguiente manera: Los profesores entregan las notas en la secretaría, la secretaria toma el registro de éstas notas y publica en una estafeta pública los resultados de notas y asistencias por materia en curso. Los estudiantes pueden revisar las estafetas públicamente y de ésta manera enterarse de sus notas y asistencias.

Para la consulta de anuncios y novedades existe por lo general una estafeta para la publicación de anuncios, también suele "pasarse la voz" de los diferentes tópicos que interesan al AEIRNNR. Los requerimientos de matriculación suelen publicarse en tiempos de matrículas en las estafetas y ventanales de las instalaciones del AEIRNNR, con la intención de dar a conocer los requerimientos para alumnos nuevos y para alumnos del AEIRNNR.

Entre los inconvenientes que se encontraron en estos procesos convencionales se pueden citar los siguientes:

- No existe privacidad al momento de consultar notas y asistencias, por cuanto las estafetas son públicas.
- No existe la disponibilidad de poder consultar a cualquier hora, puesto que la universidad tiene un horario de atención definido y no son *24 horas del día* por razones obvias.
- Existen estudiantes que no residen en la ciudad a quienes se les hace difícil tener que esperar por la publicación de notas.

Es mucho más simple consultar desde la comodidad del hogar, realizando una simple llamada telefónica, las 24 horas del día, los 365 días del año, es decir, el ideal es conseguir un sistema de servicios que satisfaga la premisa 24/7 o veinte y cuatro horas del día los siete días de la semana.

De éste básico acercamiento se determinaron los siguientes modelos de procesos que se adaptarían adecuadamente para cumplir con las tareas antes mencionadas. A continuación se documentan los casos de uso del sistema y en los anexos en A.2 se publican los diagramas UML del análisis preliminar.

4.2. Documentación de Requerimientos

En esta sección se definen las características del Callcenter y su correspondiente diseño, los Casos de Uso son la base documental de donde parte el análisis y el diseño de la solución.

4.2.1. Documentación de Casos de Uso

caso de uso: Consulta de notas

Actores: usuario *llamante*, *sistema académico del AEIRNN*

Caso de uso #: 1

Propósito: Obtener información acerca de notas de un estudiante del AEIRNNR.

Precondiciones: El callcenter debe estar en modo *listo*, en espera de llamadas entrantes. El callcenter no debe tener la entrada de linea ocupada.

Postcondiciones: El callcenter regresa al estado *listo*.

flujo normal de eventos

usuario llamante	callcenter
realizar llamada al Sistema	tocar mensaje de saludo inicial e instrucciones
ingreso opción "consulta de notas"	tocar mensaje de instrucciones para el "servicio de consulta de notas y asistencias" solicitar cédula del estudiante
ingreso de cédula	validación de cédula reproducir mensaje de espera
	consultar registros del estudiante en la Base de Datos del "Sistema Académico del AEIRNNR"
	codificar resultado a flujo audible y reproducir el flujo
	pasar a estado listo y volver a empezar

flujo alternativo

usuario llamante	callcenter
	La opción ingresada es inválida, volver a tocar mensaje de menú. reproducir mensaje de error estudiante inválido

Descripción del proceso El usuario realiza una llamada al PBX del AEIRNNR e ingresa mediante el teclado del teléfono la extensión reservada para consultas, la llamada entrante es identificada por el *Sistema*, el Sistema inicia el mensaje de saludo y de instrucciones de uso del Servicio, luego espera por algún ingreso de dígitos realizada por el llamante haciendo uso del pad telefónico.

El usuario deberá seleccionar la opción correspondiente a consulta de notas y asistencias, el sistema inicia el mensaje de instrucciones para el servicio de consulta de notas y asistencias solicitando el número de cédula del estudiante de quien se desea conocer sus notas y una clave personal de dicho estudiante. Luego de ser ingresados satisfactoriamente estos datos por parte del llamante, el Sistema inicia la búsqueda de la información de notas haciendo consultas a la base de datos del Sistema de Matriculación de Estudiantes, luego de obtener los datos requeridos, el Sistema codifica el texto en un flujo audible, una vez transformado el flujo inicia la reproducción del mismo. Finalizada la reproducción el Sistema inicia otra vez el proceso desde el mensaje de saludo.

caso de uso: Consulta de asistencias

Actores: usuario *llamante*, sistema *académico AEIRNNR*

Caso de uso #: 2

Propósito: Obtener datos de las asistencias de un estudiante del AEIRNNR.

Precondiciones: El Sistema debe estar en modo *listo*, en espera de llamadas entrantes, El sistema no debe tener la entrada de linea ocupada.

Postcondiciones: El sistema regresa al estado *listo*.

flujo normal de eventos

usuario llamante	callcenter
realizar llamada al Sistema	reproduce mensaje de saludo y lista opciones (menu)
ingresa opción çonsulta de asistencias"	reproduce mensaje de instrucciones para el "servicio de consulta de asistencias, solicitar cédula del estudiante"
ingresa cédula reproducir mensaje de espera	validación de cédula consultar registros de asis-tencia del estudiante en el "Sistema Acadé-mico del AEIRNNR" codificar resultado a flujo audible y reproducir el flujo pasar a modo listo y volver a empezar.

flujo alternativo

	La opción ingresada es inválida, volver a tocar mensaje de menú. reproducir mensaje de error estudiante inválido

Descripción del proceso El usuario realiza una llamada al PBX del AEIRNNR e ingresa mediante el teclado del teléfono la extensión reservada para consultas, la llamada entrante es identificada por el *Sistema*, el Sistema inicia el mensaje de saludo y de instrucciones de uso del Servicio, luego espera por algún ingreso de dígitos realizada por el llamante haciendo uso del pad telefónico.

El usuario deberá seleccionar la opción correspondiente a consulta de asistencias, el sistema inicia el mensaje de instrucciones para el servicio de consulta de notas solicitando el número de cédula del estudiante de quien se desea conocer sus notas. Luego de ser ingresado satisfactoriamente este dato por parte del llamante, el Sistema inicia la búsqueda de la información de asistencias haciendo consultas a la base de datos del Sistema de Matriculación de Estudiantes, luego de obtener los datos requeridos, el Sistema codifica el texto en un flujo audible, una vez transformado el flujo inicia la reproducción del mismo. Finalizada la reproducción el Sistema inicia otra vez el proceso desde el mensaje de saludo.

caso de uso: Consulta de requisitos de matriculacion

Caso de uso #: 3

Actores: usuario *llamante*, *sistema académico AEIRNNR*

Propósito: Escuchar los requisitos de matriculación.

Precondiciones: El Sistema debe estar en modo *listo*, en espera de llamadas entrantes, el sistema no debe tener la entrada de linea ocupada.

Postcondiciones: El sistema regresa al estado *listo*.

flujo normal de eventos

usuario llamante	sistema
realiza la llamada al Sistema	el sistema reproduce saludo y enumera opciones a elegir
el usuario ingresa opción "consulta de requisitos de matriculacion"	reproduce mensaje de instrucciones para "requisitos de matriculación". buscar los fonetos de requisitos de matriculación reproducir los fonetos de requisitos de matriculación. pasar a modo listo y volver a empezar.

flujo alternativo

	La opción ingresada es inválida, volver a tocar mensaje de menú.

Descripción del proceso: El usuario realiza una llamada al PBX del AEIRNNR e ingresa mediante el teclado del teléfono la extensión reservada para consultas, la llamada entrante es identificada por el *Sistema*, el Sistema inicia el mensaje de saludo y de instrucciones de uso del Servicio, luego espera por algún ingreso de dígitos realizada por el llamante haciendo uso del pad telefónico.

El usuario selecciona la opción de consulta de requisitos de matriculación, el sistema busca los últimos fonetos grabados de requisitos de matriculación y toca cada uno de los requisitos en forma cronológica.

caso de uso: Consulta de novedades y anuncios

Caso de uso #: 4

Actores: usuario *llamante*

Propósito: Escuchar la información de novedades y anuncios.

Precondiciones: El Sistema debe estar en modo *listo*, en espera de llamadas entrantes. El sistema no debe tener la entrada de linea ocupada.

Postcondiciones: El sistema regresa a modo *listo*.

Flujo: Flujo normal de eventos.

flujo normal de eventos

usuario llamante	sistema
el usuario realiza la llamada al Sistema el usuario ingresa opción "consulta de novedades y anuncios"	el sistema reproduce saludo y lista las opciones a elegir el sistema busca fonetos de anuncios y novedades. el sistema reproduce los fonetos en forma cronológica. pasar a modo listo y volver a empezar.

flujo alternativo

	La opción ingresada es inválida, volver a tocar mensaje de menú.

Descripción del proceso: El usuario realiza una llamada al PBX del AEIRNNR e ingresa mediante el teclado del teléfono la extensión reservada para consultas, la llamada entrante es identificada por el *Sistema*, el Sistema inicia el mensaje de saludo y de instrucciones de uso del Servicio, luego espera por algún ingreso de dígitos realizada por el llamante haciendo uso del pad telefónico.

El usuario selecciona la opción de consulta de novedades y anuncios, el sistema busca los últimos fonetos grabados de novedades y anuncios, y toca cada uno de los fonetos en forma cronológica.

---

caso de uso: Administrar Requisitos

Caso de uso #: 5

Actores: usuario *administrador*

Propósito: Agregar, borrar y revisar los requisitos de matriculación.

Precondiciones: El Sistema debe estar en la extensión "administrar requisitos de matriculacion".

Postcondiciones: El sistema regresa a modo listo.

Flujo: Flujo normal de eventos.

flujo normal de eventos

usuario administrador	sistema
llama a extensión de "Administrar Requisitos para matriculacion"	toca instrucciones de administrar requisitos
	obtiene la opción e ir a opción seleccionada
	pasar a modo listo y volver a empezar.

flujo alternativo

el usuario presiona 1 para grabar un nuevo requisito	El sistema inicia la grabación del requisito
	el sistema vuelve a tocar el requisito grabado
el usuario presiona 2	el sistema toca instrucciones para revisar requisitos.
el usuario presiona la tecla 9	el sistema abandona el proceso
el usuario presiona 3 para borrar el requisito	el sistema borra el requisito
	el sistema toca instrucciones para revisar requisitos.

Descripción del proceso: El Sistema inicia el mensaje de saludo y de instrucciones de uso de administración de requisitos de matriculación, luego espera por algún ingreso de dígitos realizada por el llamante haciendo uso del pad telefónico.

El usuario selecciona la opción de "administrar requisitos de matriculación", si el usuario presiona '1' el sistema graba el nuevo requisito de matriculación, si el usuario presiona '2', entonces el sistema busca los requisitos de matriculación grabados y los toca uno a uno, si durante la ejecución de los requisitos de matriculación el usuario presiona '3', el requisito que se estuvo tocando es borrado; si el usuario presiona '9' el sistema abandona este proceso y regresa al menú inmediato superior.

caso de uso: Administrar Anuncios

Caso de uso #: 6

Actores: usuario *administrador*

Propósito: Crear, borrar y revisar las novedades y anuncios.

Precondiciones: El Sistema debe estar en la extensión administración de anuncios.

Postcondiciones: El sistema regresa a modo listo.

Flujo: Flujo normal de eventos.

flujo normal de eventos

usuario administrador	sistema
llama a extensión de "Administrar anuncios"	toca instrucciones de administrar anuncios obtiene la opción e ir a opción seleccionada pasar a modo listo y volver a empezar.

flujo alternativo

El usuario presiona 1 opción "grabar un nuevo anuncio"	el sistema inicia la grabación de un nuevo anuncio El sistema toca instrucciones para revisar anuncios El sistema toca los anuncios
El usuario presiona 9	El sistema abandona el proceso
El usuario presiona la tecla 3	El sistema borra el anuncio el sistema toca instrucciones para revisar anuncios

Descripción del proceso: El Sistema inicia el mensaje de saludo y de instrucciones de uso de administración de anuncios y novedades, luego espera por algún ingreso de dígitos realizada por el llamante haciendo uso del pad telefónico.

El usuario selecciona la opción de "administrar anuncios y novedades", si el usuario presiona '1' el sistema graba el nuevo anuncio, si el usuario presiona '2', entonces el sistema busca los anuncios almacenados en el sistema y los toca uno a uno, si durante la ejecución de éstos fonetos el usuario presiona '3', el foneto que se estuvo tocando es borrado; si el usuario presiona '9' en cualquier momento, el sistema abandona este proceso y regresa al menú inmediato superior.

Luego de haber determinado las narrativas de los casos de uso anteriores, se determinó que era necesario disponer de un *ente independiente y autosuficiente* con la capacidad de contener:

1. Los datos personales del estudiante. No es necesario tener todos los datos, en el estricto sentido de la palabra, puesto que con el número de cédula y su nombre completo se puede identificar plenamente a un sujeto.

2. Las materias a las que el estudiante asiste en la actualidad. No son históricas.

3. Las notas del estudiante por materia en donde dichas notas sean de carácter: actual, inmediato y coherente.

4. Las asistencias del estudiante por cada materia a la que el estudiante asista.

5. El total de horas dictadas para cada materia en la que el estudiante asista.

Este ente estará representado como un objeto, con sus atributos, métodos y demás características de acuerdo al paradigma de programación orientado a objetos adaptado al lenguaje de programación principal (Ascher 2003, ver class y object oriented programming) los métodos son un conjunto de funciones que actúan sobre los atributos del estudiante para darle la funcionalidad al objeto. El objeto principal *datos_ estudiante* implementará su persistencia en la base de datos de apoyo del sistema, la estructura de dicha base de datos se documenta más adelante.

4.3. Resultados del Plan de Validación del Software

Las pruebas orientadas a verificar el acceso simultaneo al callcenter arrojaron los resultados del cuadro 1.

Resultados del test de concurrencia al callcenter

llamadas	*uso procesador*	*# procesesos*	*memoria usada*
1 llamada	99 % iddle	2	0.4 %
2 llamada	99 % iddle	2	0.4 %
3 llamada	99 % iddle	3	0.5 %
4 llamada	99 % iddle	4	0.5 %

Cuadro 1: Resultado del Test de Concurrencia de llamadas y la respuesta del callcenter

Los valores del cuadro 1 se refieren al estado de la memoria, procesos y procesador del equipo servidor en el momento en el que uno o mas usuarios hacían uso del callcenter. Estos resultados indican que un sistema "modesto" como el que se esta usando se comporta de manera muy cómoda frente a la carga de 1 - 4 llamadas concurrentes, puesto que solo se cuenta con dos interfaces de entrada al callcenter, se ha emulado la carga usando clientes VoIP.

Anexo a las pruebas de laboratorio, se realizaron pruebas usando un sistema prototipo puesto en funcionamiento en el AEIRNNR por el lapso de 10 días. Dichas pruebas incluyeron los siguientes aspectos:

- El levantamiento de una muestra de la información recopilada desde la Secretaría General del AEIRNNR que comprende a todos los estudiantes del AEIRNNR de la carrera de Sistemas excluyendo los siguientes sujetos:
  - Estudiantes del AEIRNNR de la Carrera de Sistemas que no poseen notas publicadas.
  - Estudiantes de los talleres de primero y segundo módulo.
  - Estudiantes de otras carreras que forman parte del AEIRNNR.

- El levantamiento de los datos personales de los estudiantes, rescatados de un padrón electoral.
- La creación de una herramienta de manipulación de datos llamada "alimentador.py", usando un ambiente gráfico desarrollado en Python con el toolkit GTK. Con el "alimentador.py" se hicieron los ingresos de datos de los estudiantes, notas y asistencias. El código fuente de este programa está publicado en el anexo A.3.5
- La puesta en funcionamiento del Callcenter Automatizado Prototipo, usando la central telefónica del AEIRNNR. Configurando el número de extensión de Consultas del Callcenter en la extensión 110.
- La recopilación de opiniones de los estudiantes mediante una encuesta de opinión acerca del servicio del Callcenter Automatizado, para la retroalimentación (feedback). El modelo de consulta ha sido incluida en el anexo A.4.

4.4. Documentación de herramientas desarrolladas y de utilidades de apoyo

Se desarrolló un conjunto de herramientas y utilidades con la finalidad de facilitar la implementación del IVR y del acceso a las bases de datos. Se documenta y explica los pormenores básicos para el funcionamiento de estas herramientas. También se explican los componentes de software necesarios para el desarrollo coordinado del Callcenter.

La base de datos de apoyo

La base de datos de apoyo que se usó sirvió como sandbox[4] para ejecutar pruebas y en el futuro podría servir como repositorio imagen auxiliar de la base de datos del sistema del AEIRNNR ha sido llamada *base de datos de apoyo* y su estructura está reflejada en la figura 1:

También se han generado las tablas (ver figura 2) para almacenar: estudiantes, materias, notas y asistencias, y se ha definido una vista (data view) que generará los datos específicos para ser consultados por el callcenter

La base de datos puede ser accedida directamente usando el programa sqlite, que es un programa de linea de comandos para interactuar con sqlite[5]. Para realizar consultas sobre la base de datos se inicia el programa sqlite de la siguiente manera:

```
patovala@ivr:~$ sqlite /var/lib/callcenter/dbestudiantes.db
SQLite versión 2.8.16
Enter ".help" for instructions
sqlite>
```

[4]Sandbox: es un ambiente de prueba (o virtual) que aisla el código no probado o en fase de pruebas del código en producción.

[5]sqlite: Una librería embebida para el uso de un sistema de base de datos llamado sqlite - www.sqlite.org

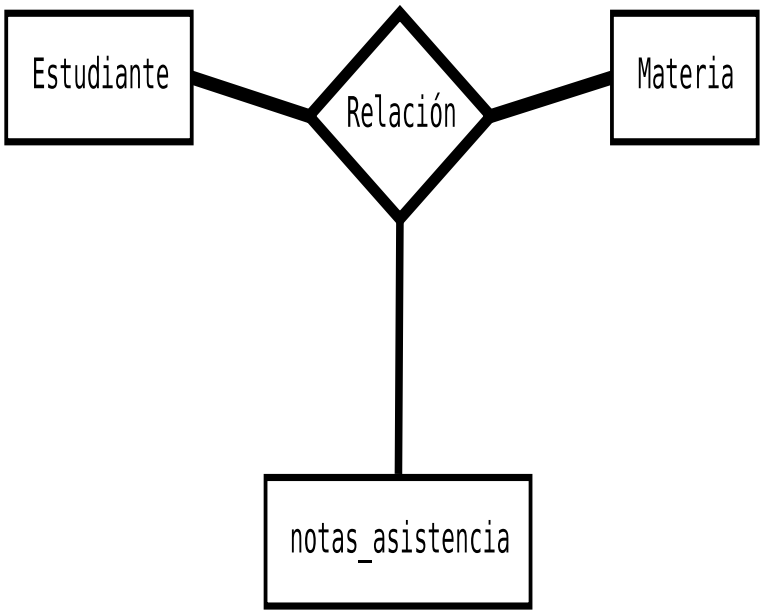


Diagrama 1: Entidad Relación

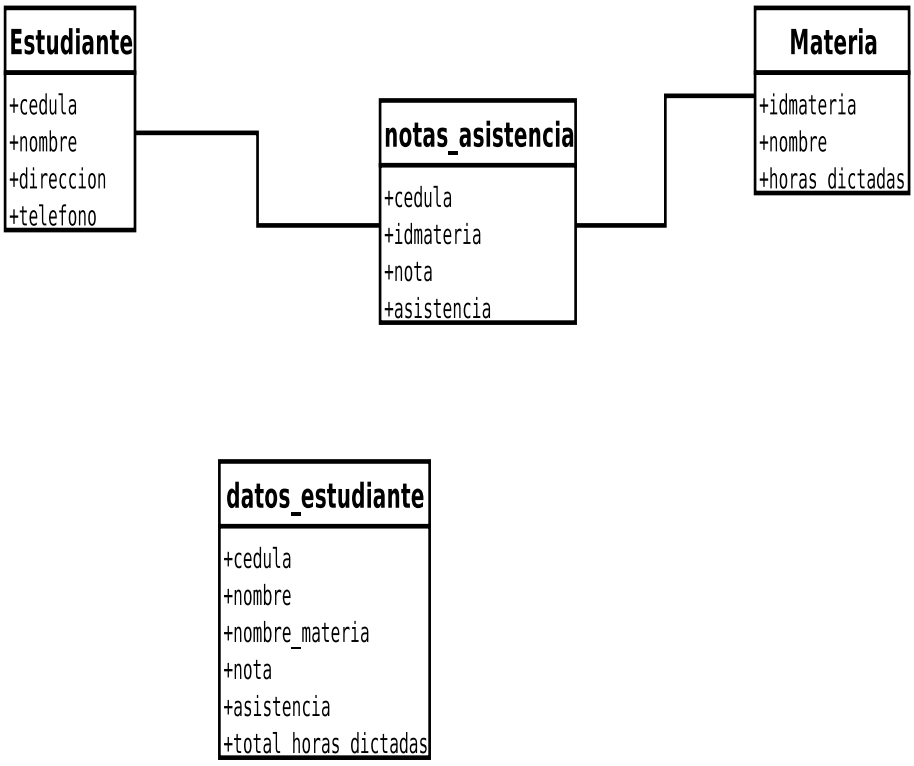


Diagrama 2: Tablas y Vistas

Se obtiene un shell para poder lanzar comandos SQL de consulta, no está por demás decir que se debe tener cuidado con las operaciones que se realizan en la base de datos porque se podrían producir pérdidas de datos. A pesar de que una de las ventajas de sqlite es el soporte a transacciones(Kline 2001)

La vista de datos del estudiante ha sido generada con el siguiente comando SQL:

```
create view datos_estudiante as
SELECT e.cedula,e.nombre,m.nombre,na.nota,na.asistencia,m.horas_dictadas
FROM estudiante e, materia m, notas_asistencias na
WHERE e.cedula=na.cedula
AND   m.idmateria=na.idmateria;
```

Con la base de datos de estudiantes disponible, se puede hacer uso de la librería *cclib* para el acceso a estos datos desde el callcenter. Es necesario hacer incapie en el hecho de que esta fuente de datos puede ser "cambiada" o "adaptada" a otros motores de bases de datos, siempre que se pueda generar una vista con los datos requeridos. Esta fuente de datos podría incluso estar en un servidor remoto.

4.4.1. La librería de acceso del callcenter *cclib*

La librería de acceso cclib ha sido desarrollada con la finalidad de aislar el acceso a los datos desde el callcenter, en el diagrama de clases (ver apéndice A.2) se puede observar el acoplamiento de esta librería y su interacción con el resto de objetos del sistema.

Se ha desarrollado también dentro de la librería una opción para *debug* (depuración) y para funcionamiento *standalone* (funcionar como un programa solo). El modo de funcionamiento de este programa es el siguiente:

```
patovala@ivr:~/callcenter$ ./cclib.py
No se ha especificado una cédula
./cclib.py PARAMETROS
PARAMETROS:  -d activa debug
             -c <cedula> : especifica la cedula a buscar
```

Desde la shell de acceso en el servidor IVR se tiene acceso a la librería y se ejecuta el programa cclib.py, los parámetros aceptables son -d para activar el debug y -c para especificar una cédula, con una cédula válida se puede hacer un test de conexión y de obtención de los datos correctos desde la base de datos.

```
patovala@ivr:~/callcenter$ ./cclib.py -c 1103418958
DEBUG ON
CEDULA ES:1103418958
Error buscando estudiante... no existe
```

El funcionamiento de la librería con los parametros correctos genera la siguiente información:

```
patovala@ivr:~/callcenter$ ./cclib.py -c 1109382000
DEBUG ON
CEDULA ES:1109382000
NOMBRE: juan martinez
CEDULA: 1109382000
NOTAS: {'living': 18.07, 'economia': 19.239999999999998, 'electronica':\\
 11.949999999999999, 'politica': 10.779999999999999, 'matematicas':\\
 18.620000000000001, 'fisica': 14.470000000000001, 'programacion 1':\\
 13.18, 'programacion 2': 19.559999999999999, 'ciencias sociales':\\
 11.279999999999999, 'metodos numericos': 14.1}
ASIST: {'living': 42, 'economia': 40, 'electronica': 33, 'politica': 27,\\
 'matematicas': 20, 'fisica': 56, 'programacion 1': 49, 'programacion 2': 60,\\
 'ciencias sociales': 34, 'metodos numericos': 48}
```

Configurando el layer de base de datos para *cclib* La librería *cclib* cuenta con un archivo de configuraciones llamado *cclib.ini*, ubicado en */etc/cclib.ini*. En éste archivo se especifica el layer de acceso a base de datos, se ha construído para nuestro propósito un layer de acceso a la base de datos *SQlite* que se encuentra dentro de la librería cclib. De acuerdo al diagrama de clases, el layer de acceso a la base de datos debe contener los siguientes métodos:

exec Ejecuta una consulta SQL en el motor, devuelve un tipo de dato lista con las tuplas del resultado.

run Ejecuta una consulta SQL que no retorna datos (ej. INSERT).

Para configurar la librería se debe editar el archivo cclib.ini, una configuración básica para el motor de base de datos sqlite tiene este aspecto:

```
[principal]
dbdriver = sqlite
dbname   = /var/lib/callcenter/dbestudiantes.db
streampath = /var/lib/callcenter/
```

En el archivo de configuración cclib.ini, la sección *[principal]* define las siguientes opciones:

- dbdriver es el driver de base de datos que se va a usar

- dbname el nombre de la base de datos

- streampath el path en donde se van a almacenar los fonetos

4.4.2. Funcionamiento del programa grabador de fonetos

Con la finalidad de crear fonetos audibles para el IVR se crearon definiciones de "lo que tiene que decir" el IVR al ser navegado por cada parte del menú interactivo, para ello se desarrollaron unos pequeños scripts en bash capaces de tomar texto plano y transformarlo a un stream audible en formato GSM(Van Meggelen 2005).

El script de creación de fonetos .gsm se llama *crear.sh* usa la utilidad sox[6] para realizar la transformación de un archivo wav generado por la utilidad text2wav[7] usando un rate de 8Khz con un canal mono y un volumen ajustado a 180

Para usar este script se debe tener un directorio con fonetos en archivos de texto *.txt* separados, cada archivo debe contener el texto que se desea "hablado". Para crear los fonetos *.gsm* se ejecuta el siguiente comando:

```
patovala@ivr:~$ cd path/a/fonetos
patovala@ivr:/path/a/fonetos$ ./crear.sh
patovala@ivr:/path/a/fonetos$ ls *.gsm
.
.
callcenter-noexiste-estudiante.gsm
callcenter-nomasdatos.gsm
callcenter-opciones-main.gsm
callcenter-opción-invalida.gsm
callcenter-punto.gsm
callcenter-requisito-borrado.gsm
callcenter-tieneasistencias.gsm
.
.
```

Ahí se tiene la lista de fonetos, ahora estos fonetos por convención deberán ser copiados a la ubicación por defecto de fonetos de nuestro callcenter en */usr/share /asterisk/sounds/callcenter/*, este path ha sido nombrado globalmente como *PATH-FONETOS*.

El proceso anterior nos ayudó mucho en la automatización de el proceso de grabado de fonetos audibles, sin embargo, debido a que el acento del interlocutor de dichos fonetos es un tanto "robótica" se ha visto la necesidad de crear una voz más real, es por eso que luego de haber determinado los fonetos necesarios para el sistema y de haber realizado las pruebas de acoplamiento semántico de lo que "dice" el IVR, se concluyó con un conjunto de frases en archivos de texto.

Con estas frases ya se puede comenzar a grabar fonetos más profesionales, se puede hacer uso de todo un estudio de grabación para manejar el audio adecuadamente, para evitar fenómenos típicos de la conversión análoga a digital[8]. Sin embargo una de las opciones más cómodas era la de usar el mismo sistema telefónico para realizar dichas grabaciones, puesto que la tecnología telefónica ha venido improvisando y lidiando con dichos problemas en la comunicación hasta el punto de obtener un medio de comunicación aceptable.

Se construyó un sistema de grabación que funciona de la siguiente manera:

1. El usuario llama a la extensión 205 del callcenter
2. El sistema le pide ingresar un número de identificación de la grabación
3. El sistema le indica al usuario en qué momento comenzar a hablar

[6] sox - Sound eXchange : traductor universal de samplers de sonido, es parte de algunas distribuciones en linux.

[7] text2wav es parte de la suite festival (www.festival.org), un sistema *text-to-speech* de transformación de texto a ondas audibles habladas

[8] Los típicos problemas de transformación análoga/digital como: saturación en la salida, eco, scratch, etc

4. Al terminar de hablar, el usuario presiona '#'

5. El sistema almacena la grabación en formato gsm en */tmp* del equipo.

Luego de tener estas grabaciones ya se puede renombrarlas de acuerdo a las convenciones adoptadas y ubicarlas en el PATHFONETOS del sistema. En el Apéndice A.1 se incluye una lista de todos los fonetos definidos para el callcenter.

4.4.3. La utilidad *revisarmensajes.py*

Esta es otra aplicación *standalone* que forma parte de la librería de utilidades *cclib*, sirve para la revisión de mensajes tanto de anuncios como de requisitos de matriculación. La lógica de este script es el resultado del caso de uso (ver A.2) en donde se detallan los pasos a seguir para la obtención de anuncios y requisitos de matriculación.

Esta utilidad funciona en un ambiente AGI(Van Meggelen 2005) que es un ambiente que nos permite interactuar desde asterisk con el mundo exterior. El script es llamado desde una extensión del *dialplan* en algunos *contextos* que forman parte de la lógica del IVR. El modo básico de uso es el siguiente:

```
exten => 2,n,AGI(revisarmensajes.py|anuncio|user) ; llamar al agi en modo usuario
```

Los argumentos que soporta esta utilidad son los siguientes:

patron Que puede ser anuncio o requisito, para que el IVR toque anuncios o requisitos respectivamente.

modo Que puede ser admin para modo administración, en donde el script nos presentará las opciones de eliminación de fonetos, y el modo user en donde el script solo funciona como un explorador.

4.4.4. La utilidad *ccconsulta.py*

La utilidad *ccconsulta.py* es otro script para el modo AGI del IVR que usa cclib para acceder a los datos de los estudiantes, este es el layer lógico de presentación. La principal función de esta utilidad de acuerdo a los casos de uso y los diagramas de actividades (ver A.2) es la de generar fonetos audibles a partir de la información obtenida desde cclib. El modo básico de uso es el siguiente:

```
.
exten => 1,n,AGI(ccconsultar.py|${CEDULA}|notas) ; llamar al agi a consultar notas
.
```

Los argumentos que soporta esta utilidad son los siguientes:

cedula La cédula del usuario del cual se desea consultar notas o asistencias.

consulta Puede ser notas o asistencias.

4.5. Construcción del Sistema Interactivo de Respuestas por Voz - IVR

Para el Sistema Interactivo de Respuestas por voz (IVR), componente fundamental del sistema, se usará un paquete de software llamado Asterisk. Junto con la interfaz de hardware propia para canales PSTN vamos a construir un sistema de respuestas por voz altamente configurable y que permitirá la interacción con los usuarios llamantes. A continuación se documenta los procedimientos y pasos para la puesta en marcha del IVR.

4.5.1. El Paquete Asterisk

Asterisk es un paquete de Software con licencia OpenSource que permite el desarrollo de sistemas informáticos para atacar requerimientos específicos. Provee una librería con funciones de telefonía completos, mediante el uso de patrones de dígitos (llamados extensiones), permitiendo un completo control de las capacidades de PBX, IVR y Callcenter así como el ruteo de llamadas de una manera práctica. Tal vez una de las características más valiosas de este juego de herramientas es el hecho de ser Abierto por naturaleza. Asterisk puede ser en el futuro mejorado por la comunidad de desarrolladores de todo el mundo. Esto le permite al sistema madurar de acuerdo a las necesidades de la industria. Se puede hacer uso del mismo respetando las licencias impuestas y contribuyendo con el desarrollo del producto. Por otro lado la misma naturaleza del paquete desemboca en el hecho de que existe un soporte[9] basado en la comunidad, y soporte técnico pagado ofrecido por la empresa desarrolladora del paquete (Digium.com). La curva de conocimiento requerida para desarrollar soluciones sobre este paquete es moderada e igual a la curva necesaria para aprender un nuevo lenguaje de programación, sin embargo una vez que se conocen los conceptos básicos, el resto viene más fácil con tiempo y práctica.

Conocimientos Requeridos y habilidades previas a usar asterisk Debido a la ilimitada flexibilidad que presenta Asterisk, el hecho de configurar un sistema requiere más de conceptos técnicos como: Conocimientos de Administración de GNU/Linux así como un básico conocimiento de Telefonía. En esta sección se abordan los conceptos más básicos y superficiales de estas tecnologías paralelamente a incursionar en los conceptos relevantes al diseño de Asterisk.

- Telefonía: Asterisk es un PBX y esto implica que mientras más conocimientos se tenga de telefonía, más fácil será aprenderlo. Términos como PSTN[10] y VoIP[11] deberán estar comprendidos.
- Instalación y Administración de GNU/Linux: Puesto que el proyecto Asterisk funciona bajo esta plataforma y en arquitecturas i386, es deseable

[9] La comunidad de desarrolladores y usuarios de Asterisk es una comunidad muy amigable y abierta a usuarios noveles, es el conocimiento puesto en manos de la comunidad

[10] PSTN: Public Switching Telephony Network - La red de telefonía pública Ej. Las redes de telefonía nacionales como Andinatel y Pacifictel

[11] VoIP: Voz sobre IP Protocolo de comunicación por medio de voz en una red TCP/IP

poseer conocimientos básicos de administración e instalación del Sistema Operativo GNU/Linux y particularmente Debian GNU/Linux.

Requerimientos mínimos de hardware impuestos por Asterisk El paquete Asterisk puede implicar usos exigidos del procesador debido al uso del CPU para realizar el procesamiento digital de señal (DSP - Digital Signal Processing). Por lo general dependiendo de la intensidad de la aplicación se podría decir que para la instalación de un sistema básico PBX se lo podría implementar usando un Pentium de 300Mhz con 256Mb de RAM. Es esencial para la construcción de nuestro sistema el uso de una interfaz PCI del kit de desarrollo ofrecida por Digium[12]. Un asunto importante referente a hardware a tomar en cuenta al momento de desarrollar y construir nuestro sistema es el de instalar el equipo x386 de tal manera de que no comparta IRQ[13] Es indispensable que las tarjetas de comunicación del tipo X100P vendidas por Digium no compartan ningún IRQ y tengan en uso un único IRQ por dispositivo, de esta manera se dedicar al CPU de manera más efectiva el control de recursos del sistema. En un sistema GNU/Linux, se puede ver el número de IRQs asignados a cada dispositivo del sistema en /proc/interrupts.

Requerimientos mínimos de Software Asterisk fue diseñado para funcionar en arquitecturas x386, sin embargo ha sido portado no oficialmente hacia otras plataformas como BSD y OSX, Digium ofrece soporte comercial únicamente para plataformas Linux en Arquitecturas x386.

El kernel de Linux deberá ser superior al kernel 2.4, al momento de escribir esto, el kernel oficial estable es el 2.6.12.

El único hardware necesario adicional a una interfaz de red ethernet es el de las interfaces de red provistas por Digium, no es necesario el uso de una tarjeta de sonido o similares. Si esta usando hardware de Digium o ztdummy, es necesaria la instalación del paquete zaptel.

Performance del paquete Asterisk Se debe prestar especial atención en el diseño de una efectiva configuración del Sistema Operativo. Los servicios y procesos necesarios para funcionar Asterisk no solo deberán estar optimizados y tunificados para que reciban la más alta prioridad por parte del Sistema Operativo, sino que en lo posible, deberán los recursos ser exclusividad del SO. Una razonable *performance* se puede obtener usando equipos PC modernos, se ha estimado que se puede tener 5 o más estaciones en un sistema GNU/Linux moderno compartiendo recursos con servidores web, bases de datos, etc, sin ningún problema. Ahora si se desea obtener un máximo de prestaciones en ambientes de alta disponibilidad (30 estaciones telefónicas o mas), se deberá especializar el equipo *únicamente* a tareas específicas de Asterisk.

[12] Digium: La empresa desarrolladora de Asterisk - ver www.digium.com

[13] IRQ: Interrupt Request - Peticiones de interrupción de hardware, revisar mas en (wikipedia enciclopedia web)

Instalando Asterisk en Debian Una de las razones por las que se ha escogido a Debian como distribución preferida para este proyecto se debe a su extenso juego de paquetes[14] y su extenso soporte por parte de la comunidad[15]. También existe documentación libre en internet acerca de esta maravillosa distribución, libros, manuales de referencia del Sistema Operativo, CookBooks y Recetarios de "How-tos"que nos ayudarán a configurar y programar de manera eficiente nuestro sistema.

Asterisk forma parte del juego de paquetes "dev"[16] desde hace algún tiempo. Para la instalación de dichos paquetes se debe proceder de la siguiente manera:

```
Como super usuario:
# apt-get update
# apt-get install asterisk asterisk-dev asterisk-config\
  asterisk-dev asterisk-h323
```

Actualizar la lista de paquetes disponibles (Se debe contar con un acceso a internet de banda ancha o similar). Se instalan los paquetes asterisk, asterisk-dev, asterisk-config y asterisk-h323

Afortunadamente el sistema de empaquetamiento de debian está bien diseñado, maneja las dependencias de paquetes de manera semi-automática, por lo general no se debería tener problemas al ejecutar los comandos apt antes mencionados.

También es necesario disponer de las fuentes de un kernel actualizado para poder compilar los drivers para los dispositivos FXO y similares, se necesita un kernel de versión mayor a la 2.4.x, para instalar las fuentes del kernel linux se procede de la siguiente manera:

```
# apt-cache search kernel source 2.6
...
kernel-source-2.6.8 - Linux kernel source for version 2.6.8
with Debian patches
linux-source-2.6.12 - Linux kernel source for version 2.6.12
with Debian patches
...

# apt-get install linux-source-2.6.12
```

Existen algunos paquetes con el código fuente del kernel disponibles para la distribución debian (stable). se escoge la versión más actualizada (recomendable) y se la instala usando apt.

[14] Debian Cuenta con alrededor de 9000 paquetes de software desde utilidades, herramientas de desarrollo, paquetes ofimaticos, juegos, etc, es decir un verdadero repositorio de software libre listo para ser usado y modificado

[15] Comunidades como Badopi.net, Bulma.net, la lista de discusión debian-es, son una fuente inagotable de soporte *gratuito*. formamos parte de esta comunidad desde hace 8 años, colaboramos con nuevos usuarios de linux y damos soporte en una verdadera comunidad.

[16] Los paquetes *dev* son paquetes de software con librerías y utilidades, bindings a lenguajes como c,c++,perl,python,php,etc para que desarrolladores hagan uso de ellos y no tener que "reinventar la rueda" un valor agregado al hecho de usar Software Libre.

También es necesario la instalación de los drivers adecuados para el manejo de interfaces de Telefonía Zaptel provistas por Digium que son las que se usará para este proyecto. Los drivers para la configuración de estas interfaces bajo linux usando kernel 2.6>, se pueden descargar del sitio de Asterisk.

Para compilar los drivers se procede de la siguiente manera:

```
# cd /usr/src

# export CVSROOT=:pserver:anoncvs@cvs.digium.com:/usr/cvsroot

# cvs login
password: anoncvs

# cvs checkout zaptel

# cd /usr/src/zaptel

# make clean ; make linux26; make install
```

Se cambia al directorio de fuentes de debian /usr/src. Nos conectamos al cvs anónimo de Digium y descargamos los drivers zaptel,luego cambiamos al directorio /usr/src/zaptel y procede a compilar los drivers, hay que tomar en cuenta que la compilación esta orientada a kernels mayores a 2.6.

Una vez que tengamos el módulo zaptel compilado en el sistema lo tendremos que activar y hacerlo disponible por el sistema:

```
# modprobe zaptel

# lsmod
Module Size Used By Not tainted
. .
. .
zaptel 18923 0 (unused)
```

En la configuración del modulo zaptel (/etc/zaptel.conf) se deberán activar las interfaces FX0 de las que vamos a disponer.

```
#
# Zaptel Configuration File
#
fxoks = 1
fxsks = 1
loadzone = ec
defaultzone = us
```

En el archivo de configuración zaptel.conf, especificamos el número de interfaces FXO. El parámetro fxoks=1 especifica al módulo wcfxs que trabaje con señales FXO y el parámetro fxs para señales FXS, la terminación ks indica que se use el protocolo de supervisión llamado *koolstart*, koolstart incorpora también características como *loopstart* y *groundstart* (ls y gs respectivamente). La idea de el protocolo koolstart es agregar inteligencia al circuito para saber que es lo que la otra parte de la linea esta haciendo. Se ha convertido en el estándar informal en Asterisk. El código del país (Ecuador) y la zona por defecto en caso de que no exista locales para ec.

Luego de haber configurado las interfaces zaptel, se deberá cargar el módulo wcfxs:

```
# modprobe wcfxs
Freshmaker version: 71
Freshmaker passed register test
Module 0: Installed -- AUTO FXS/DPO
Found a Wildcard TDM: Wildcard TDM100P REV E/F
```

Con el comando modprobe se activa el módulo, en Debian existe una manera de hacer estos cambios permanentes, en el manual de referencia de debian se expone el comando modconf que permite automatizar este proceso. El módulo al activarse presenta información útil de su test y diagnóstico al iniciarse.

Una vez cargado el controlador, se debe proceder a la configuración de los canales con el uso de ztcfg. EL comando ztcfg se usa para la configuración de las señales usadas por los interfaces físico FX. ztcfg usa las configuraciones de señales especificadas en zaptel.conf. Para ver la salida que el comando ztcfg genera, se debe agregar el parámetro *-vv* para provocar una salida más minuciosa.

Configuración de los canales de comunicación Los canales son conexiones lógicas a varias rutas de señal y transmisión que Asterisk puede usar para crear y conectar llamadas. Ellas pueden ser físicas (como un puerto análogo FXO), o basadas en software (como un canal IAX), dependiendo de su naturaleza. El *dialplan*[17], es el lugar en donde se definen las reglas que Asterisk sigue para determinar como queremos que Asterisk conecte los canales. Obviamente, antes de crear un dialplan, nosotros debemos determinar que clase de canales nosotros necesitamos y configurarlo para hacerlo usable por el sistema.

Los canales vienen de toda clase de formatos; circuitos físicos de telecomunicación (como FXO, FXS, PRI, BRI), basados en software, entidades conectadas en red (como SIP y IAX), y canales internos exclusivos de Asterisk para todo tipo de hechizos (como Agentes, Consola, Local y el exclusivo TDMoE). Asterisk trata a todos estos canales como puntos de conexión que se juntan en el dialplan.

[17] dialplan: Es un plan detallado que determina la manera en que será tratada los eventos que se produzcan en las interfaces del sistema Asterisk

Es importante recordar que estos canales podrían variar de acuerdo a la conectividad y su tecnología, Asterisk nos permite tratar a todos ellos casi de la misma manera.

Asterisk funciona como un PBX de manera flexible y poderosa debido mayormente a la forma en que maneja los canales. En muchos otros PBXs propietarios, los canales tenían maneras completamente diferentes de comunicación. Puertos Station, Puertos Trunk y Puertos IP son tan diferentes que se han necesitado años de experiencia para poderlos unir en una interface abstracta que permita tratarlos con relativa similitud.

Los términos FXO y FXS tienen sus orígenes en un antiguo servicio llamado "Foreing eXchange", este circuito tenía como propósito permitir a un teléfono análogo en una ubicación remota conectarse a un PBX en cualquier lugar. Un Circuito FX tiene dos terminales (El terminal STATION que es en donde el teléfono se encuentra y el terminal OFICE que es en donde se encuentra el PBX). Una tarjeta FXS es un dispositivo en donde se conecta una estación y para cumplir con su trabajo debe comportarse como una oficina central. De la misma manera, una FXO conectada a una Oficina Central significa que debe comportarse como un teléfono (un módem es un ejemplo clásico de un dispositivo FXO).

FXS Los canales Foreign eXchange Station (FXS) proveen la misma interface que una linea análoga tradicional que las compañías telefónicas ofrecen. A parte de otras cosas los canales FXS ofrecen:

- Tono de marcado
- Voltaje de timbrado
- DTMF (touch tone) detection
- Mensaje de espera
- Identificador de linea llamante (calling line ID)

Cuando usted configura en una tarjeta FXS Digium (como la TDM400P con dispositivos FXO instalados), usted necesita definir parámetros que son relevantes para dicho dispositivo FXS.

Lo más importante es que recuerde que una tarjeta FXS provee la señal de una oficina central (Central Office).

FXO Un dispositivo FXO es una tarjeta que se conecta a una oficina central. Un módem es un ejemplo clásico de una FXO (De hecho, si usted posee una FXO Digium como la X100P, es de hecho un módem) Un dispositivo FXO debe ser capaz de:

- Generar DTMF (touch tones).
- Detectar dial tone (tonos de marcado)

- Detectar timbrado (ringing)
- Detectar mensaje en espera
- Interpretar caller ID (Identificador de llamadas)
- Señales On Hook/Off Hook del otro lado así como flash

La diferencia principal entre los canales FXO y FXS es "signalling"(señales).

Definición de canales de entrada Para las interfaces FXO que se han instalado, necesitamos especificar los canales de entrada y las acciones que serán llevadas a cabo para manejar adecuadamente las llamadas entrantes de nuestro Sistema. El archivo de definiciones de canales FXO y FXS se encuentra en */etc/asterisk/zapata.conf*. Aquí vamos a especificar un canal de llamadas entrantes.

```
; <PV> definición de canales de entrada
[trunkgroups]
[channels]
; definiciones especificas del canal
; eliminación de ecos
; características de la linea: identificador de llamadas, llamada
; en espera
.
.
; definición de canales
context=entrantes ; llamadas entrantes van a 'entrantes' en extensions.conf
signalling=fxs_ks  ; Se usa señales FXS para canales FXO
channel => 1 ; PSTN esta conectado al canal 1
```

Ahora ya disponemos de un sistema listo para gestionar llamadas entrantes, estas llamadas serán manejadas mediante el extenso juego de herramientas del paquete asterisk, de nuestras aplicaciones de interacción con las bases de datos y demás herramientas necesarias para cumplir con nuestro propósito. La lógica que se use para la interacción con el llamante están especificadas en el archivo de configuración */etc/asterisk/extensions.conf* es aquí en donde se completa el diseño de nuestra aplicación y es en donde se programará la lógica de interacción con el usuario. Como punto principal para atender las llamadas entrantes se ha de definir un *context* llamado *entrantes* de acuerdo a la configuración de nuestros canales *zaptel*.

Especificación de las acciones a tomar para las llamadas entrantes Lo primero que gestionaremos es las llamadas entrantes, que rumbo tendrán, y cual será las acciones a tomar para cada situación específica, no podremos terminar todas las situaciones en este momento por cuanto el análisis y diseño específico de la solución proveerán del complemento, pero el esquema general se puede centrar en el requerimiento principal del sistema que es el de *obtener información puntual y general a partir de una llamada telefónica*. Es por eso que podríamos definir el funcionamiento general del sistema bajo el siguiente requerimiento:

> "El Sistema encargado del Servicio de Consultas para el AEIRNNR usando un CallCenter, deberá atender todas las llamadas que ingresen mediante la línea telefónica designada, deberá interactuar con el llamante y ser amigable con él ofreciendo el control de sus servicios para provecho del usuario llamante, esto deberá ser realizado de manera repetitiva e indefinida por todas y cada una de las peticiones realizadas a su Sistema de Entrada"

Para cumplir con este requerimiento general se ha diseñado el siguiente esquema (Diagrama de casos de uso)A.2, el cual documenta y modela este requerimiento, se ha diseñado el esquema pensando en el hecho de que el sistema se irá completando a medida que afinemos los requerimientos de los procesos de *negocios* del AEIRNNR y busquemos las mejores alternativas de diseño.

4.6. Creación de las partes iniciales del IVR

El IVR esta formado en su totalidad por la lógica que define la interacción entre el *usuario llamante* y el *callcenter*. De acuerdo a nuestro análisis y diseño, a continuación se documenta el conjunto de comandos que han servido para poder cumplir con nuestros propósitos.

4.6.1. Instalación de Prompts

Los prompts básicos son los fonetos que nos permitirán escuchar en otro idioma (en español) algunos mensajes que el sistema asterisk tiene por defecto, esto ha sido estandarizado para que el i18n[18] del paquete asterisk sufra el menor impacto.

En nuestro sistema Debian, los prompts se encuentran en */usr/share/asterisk /sounds*, se sugiere consultar el manual de referencia de Debian y el manual de asterisk (Van Meggelen 2005).

Breve introducción a la nomenclatura de Asterisk

Como ya se expuso anteriormente, el corazón del sistema asterisk es su archivo de configuraciones
ubicado en un sistema debian en: */etc/asterisk/extensions.conf*, este archivo es el lugar en donde se define el *dialplan*, el *dialplan* de acuerdo a nuestro análisis orientado a objetos, en nuestro sistema es un objeto que interactúa con el *usuario llamante* y atiende sus órdenes.

El *dialplan*, esta dividido en secciones llamadas *contexts*, los *contexts* son grupos de extensiones referenciadas por un nombre, para nuestro caso práctico vamos a definir en cada context un método o función a ser cumplida.

Un context se define de esta manera:

```
[nombreContext]
exten => <comando>
```

[18] i18n: Internationalization - Internacionalización estas siglas definen un estándar bajo el cual un paquete o especificación cumple con ciertos aspectos que lo hacen transportable a otros idiomas.

El final de un context es la declaración de un nuevo context. Para mayor información de como se gestionan los context, le sugerimos revisar la excelente documentación acerca del sistema asterisk(Van Meggelen 2005)

Construcción del grabador de voz

El grabador de voz del IVR es un pequeño programa interactivo que desarrollamos para grabar los diferentes mensajes, concretamente vamos a crear un grabador que le permita al administrador grabar nuevos mensajes para ser escuchados por el *usuario llamante*. También nos sirve como grabador de anuncios y de requisitos de matriculación.

Hemos definido el siguiente context para este propósito:

```
.
.
; el grabador
[grabador]
exten => s,1,Answer ; Contestar la llamada
exten => s,2,DigitTimeout,5 ; Set Digit Timeout to 5 seconds
exten => s,3,ResponseTimeout,10 ; Set Response Timeout to 10 seconds
exten => s,4,Set(LANGUAGE()=es) ; Setear el lenguaje a español
exten => s,5,Goto(1,1)

; Phrase Recording
exten => 1,1,Wait(1)
exten => 1,2,Read(PHRASEID|vm-enter-num-to-call)
exten => 1,3,Wait(1) ; give yourself 1 secs and wait for beep
exten => 1,4,Record(/tmp/${PHRASEID}:gsm)
exten => 1,5,Wait(2)
exten => 1,6,Playback(/tmp/${PHRASEID})
exten => 1,7,Wait(1)
exten => 1,8,Hangup
```

El grabador creado se lo puede usar de acuerdo a lo que se mencionó en 4.4.2.

4.6.2. Definición de los menús principales de la aplicación

Al contestar el IVR la llamada entrante, será atendido por el menú principal de la aplicación en donde se presentan las siguientes opciones:

- Opción 1 para consultar las notas
- Opción 2 para consultar asistencias
- Opción 3 para consultar requisitos de matriculación
- Opción 4 para consultar anuncios y novedades

El menú principal de la aplicación esta gobernado por el siguiente código

```
.
.
.
[astPBX]
; El inicio de la aplicación
exten => s,1,Answer
exten => s,n,Set(LANGUAGE()=es)
exten => s,n,Set(TIMEOUT(digit)=5)
exten => s,n,Set(TIMEOUT(response)=10)
; Background : toca algo hasta que se presione alguna tecla del dial
; callcenter : opciones 1 consultar notas, 2 consultar asistencias, 3 consultar
;              requisitos matriculación, 4 consultar anuncios
exten => s,n,Background(callcenter/callcenter-bienvenido)
exten => s,n(opciones),Background(callcenter/callcenter-opciones-main)

;Consulta de notas
exten => 1,1,NoOp(Extension: ${EXTEN} opción: Consultar notas)
exten => 1,n(ingreso),Read(CEDULA,callcenter/callcenter-ingresar-cedula,10,,,10)
exten => 1,n,NoOp(Debug: cedula es ${CEDULA})
exten => 1,n,AGI(ccconsultar.py|${CEDULA}|notas) ; llamar al agi a consultar notas
exten => 1,n,NoOp(Debug: cedula es ${CEDULA})
exten => 1,n,Hangup

;Consulta de asistencias
exten => 2,1,NoOp(Extension: ${EXTEN} opción: Consultar asistencias)
exten => 2,n(ingreso),Read(CEDULA,callcenter/callcenter-ingresar-cedula,10,,,10)
exten => 2,n,NoOp(Debug: cedula es ${CEDULA})
exten => 2,n,AGI(ccconsultar.py|${CEDULA}|asistencias) ; llamar al agi en modo
 usuario
exten => 2,n,Hangup

;Consulta de requisitos
exten => 3,1,NoOp(Extension: ${EXTEN} opción: Consultar requisitos)
exten => 3,n,Gosub(astPBXadmin-requisitos,3,1)
;Consulta de anuncios
exten => 4,1,NoOp(Extension: ${EXTEN} opción: Consultar anuncios)
exten => 4,n,Gosub(astPBXadmin-anuncios,3,1)

; salir
exten => 9,1,NoOp(adiós, se pulso 9)
exten => 9,n,Playback(callcenter/callcenter-adios)

; el timeout
exten => t,1,NoOp(timeout en astPBX, regresando al principio)
exten => t,2,Goto(s,opciones)

; el invalido
exten => i,1,Playback(callcenter/callcenter-opción-invalida)
exten => i,2,Goto(s,opciones)

; el hangup
exten => h,1,NoOp(chao far ha colgado)
exten => h,n,Hangup
.
.
```

Consulta de Notas

La opción de consulta de notas hace uso del script AGI *ccconsulta.py* (ver 4.4.4) para acceder a la base de datos de estudiantes y consultar las notas. La lógica de este script está documentada en los diagramas UML A.2 del apéndice.

La llamada desde el dialplan está constituída por el siguiente código fuente:

```
;Consulta de notas
exten => 1,1,NoOp(Extension: \${EXTEN} opcion: Consultar notas)
exten => 1,n(ingreso),Read(CEDULA,callcenter/callcenter-ingresar-cedula,10,,,10)
exten => 1,n,NoOp(Debug: cedula es \${CEDULA})
exten => 1,n,AGI(ccconsultar.py|\${CEDULA}|notas) ; llamar al agi a consultar
 notas
exten => 1,n,NoOp(Debug: cedula es \${CEDULA})
exten => 1,n,Hangup
```

En el tag *ingreso* del código anterior se obtiene el número de cédula ingresado por el usuario haciendo uso del pad telefónico, los impulsos generados son variaciones de voltaje del tipo DTMF (Van Meggelen 2005). En la linea exten =>1,n,AGI(ccconsultar.py|$CEDULA|notas) se realiza la llamada al script ccconsulta.py, éste script toma el control del callcenter desde ese punto.

Consulta de Asistencias

La consulta de asistencias hace uso igual de la utilidad *ccconsulta.py* para una mayor comprensión del funcionamiento de este script revisar la referencia: 4.4.4. Simplemente se cambia el parametro de consulta de *notas* a *asistencias* y se obtiene los datos de asistencias.

Menu Administrador del Sistema

El menu administrador del sistema nos permite administrar anuncios y novedades así como también administrar requisitos de matriculación. De esta manera el administrador del callcenter podrá grabar dichos anuncios en el sistema. Luego un usuario al llamar a la extensión del callcenter podrá tener acceso a dicha información. El menu de administración nos permite gestionar estos mensajes de forma cronológica. La construcción de este menu está definido por el siguiente código:

```
[astPBXadmin]
;Administrador del pbx
exten => s,1,NoOp(Administrador del callcenter)
exten => s,2,Set(TIMEOUT(digit)=5)
exten => s,3,Set(TIMEOUT(response)=10)
; mensaje: para administrar anuncios pulse 1, para requisitos de
;          matriculación pulse 2.
exten => s,4(restart),Background(callcenter/callcenter-admin-bienvenido)
exten => s,5,Wait(3)
exten => s,6,Goto(restart)
```

```
exten => 1,1,Goto(astPBXadmin-anuncios,s,1)
exten => 2,1,Goto(astPBXadmin-requisitos,s,1)
exten => 9,1,Hangup
exten => i,1,Goto(s,1)
```

Al seleccionar la opción 1 nos dirigiremos al administrador de anuncios, al presionar 2 nos dirigiremos al administrador de requisitos. Al presionar la opción 9 terminaremos nuestra sesión de administración

4.6.3. Creación de la base de datos de apoyo al sistema académico del AEIRNN

Nuestra concepción particular de la base de datos a usar para el sistema contempla la definición de la base de datos de acuerdo al análisis realizado en el literal 4.4, se definió que la mejor forma de obtener los datos era mediante una vista, puesto que la gran mayoría de base de datos soportan la creación de *vistas*, se podía *emular* una tabla similar para los distintos motores de bases de datos.

Es así que en nuestra base de datos sqlite usamos la siguiente definición de datos:

```
create table estudiante(
cedula INTEGER NOT NULL PRIMARY KEY,
nombre VARCHAR(150) NOT NULL,
direccion VARCHAR(50) DEFAULT 'desconocido',
telefono VARCHAR(15) DEFAULT '0991231829'
);

create table materia(
idmateria INTEGER NOT NULL PRIMARY KEY,
nombre VARCHAR(150) NOT NULL,
horas_dictadas INTEGER NOT NULL DEFAULT 100
);

create table notas_asistencias(
cedula INTEGER NOT NULL,
idmateria INTEGER NOT NULL,
nota DECIMAL NOT NULL,
asistencia INTEGER NOT NULL
);

create view datos_estudiante as
SELECT e.cedula,e.nombre,m.nombre,na.nota,na.asistencia,m.horas_dictadas
FROM estudiante e, materia m, notas_asistencias na
WHERE e.cedula=na.cedula
AND   m.idmateria=na.idmateria;
COMMIT;
```

Se ha creado un script que automatiza el proceso de creación de la base de datos, como referencia se lo ha puesto en el apéndice A.3.1 con el nombre de esquema.sql.

Con este script podremos recrear la base de datos en cualquier equipo usando sqlite y con pequeñas modificaciones podría ser adaptado a otros motores como MySql o Postgresql. El método más fácil de agregar este modelo a sqlite es de la siguiente manera:

```
patovala@powerpod:~/tmp$ sqlite mibase.sqlite < ~/callcenter/recursos/esquema.sql
```

Ya podremos ahora contar con la base de datos haciendo referencia desde nuestro sistema, para comodidad de paths, se ha definido una variable global *dbname* que podrá ser definida en *cclib.ini* para ubicar correctamente a la base de datos.

4.6.4. Detalle de los programas de acceso a los datos

Usando cclib - procesos internos El acceso a la librería *cclib* desde *cc-consulta.py* constituye el eslabón que une la parte del IVR con la parte de base de datos, el código fuente relevante lo detallamos a continuación, el resto del código y sus detalles pueden ser consultados en el apéndice A.3.2.

```
# definición de globales
PATHFONETOAST="/usr/share/asterisk/sounds/callcenter/"
CONSULTA="notas" # puede ser notas o asistencias
CONFIGFILE="/etc/cclib.ini" # el path al config de la base de datos
#CONFIGFILE="cclib.ini" # el path al config de la base de datos
CCLIBPATH="/usr/local/cclib" # el path a la libreria cclib
```

Aqui se han definido variables globales de acuerdo a nuestra estructura en el sistema de archivos del IVR, también incluyen variables por defecto, en Python una inicialización de variables asigna un "nombre" a un objeto.

```
def tocar(data):
    """Explicitamente toca un foneto"""
    # Send Asterisk a command
    sys.stdout.write("EXEC Playback %s%s\n" % (PATHFONETOAST,data))
    sys.stdout.flush()
```

Se ha definido un método encargado de tocar fonetos, este es un wrapper[19] para las funciones AGI que ofrece *asterisk*, se trata de una implementación de un esquema de "conversación" usando sdtin, sdtout y sdterr (la salida estándar, la entrada estándar y el error estándar) implementados por el sistema operativo. Al final de una escritura en el descriptor se debe hacer un flush del descriptor, para que los datos en caché sean asentados en el descriptor.

```
 # importar la libreria e inicializarla
sys.path.append(CCLIBPATH)
import cclib # la libreria de acceso a recursos de la BD
cclib.inicializar(CONFIGFILE)
```

```
# validar los datos, revisar si existe un estudiante con esta cédula
# buscar los datos del estudiante con cédula CEDULAR
try:
```

```
        de=cclib.buscarDatosEstudiante(CEDULA)
        tocar("callcenter-elestudiante")
        recibir()
        tocarNumero(CEDULA)
        recibir()
        tocar("callcenter-tiene%s" % CONSULTA)
        recibir()
.
.
.
except cclib.CCLibError:
        pdebug("ERROR!: no se encontraron datos")
        tocar("callcenter-noexiste-estudiante")
        sys.exit()
```

En esta parte del código se hace la importación de la librería cclib desde CCLIBPATH, se llama al método inicializar usando el CONFIGFILE que define las globales que usará el cclib. Luego del try, se hace uso de la librería cclib para buscar los datos del estudiante usando como parámetro CEDULA. La librería cclib define una excepcion que será capturada por este extracto de código si se produce algún error.

Construcción de la librería cclib La librería cclib nace como un requerimiento para cumplir con un proceso definido de acuerdo a nuestro análisis y su documentación (ver A.2). Los puntos más reelevantes son la creación de las clases encargadas de manipular los datos y de clases como tipos de datos en sí.

```
class BuscadorEstudiantes:
    "un objeto buscador de estudiantes"
    def __init__(self):
        global db
        self.db=db

    def buscar(self,cedula):
        "buscar en la base de datos por el estudiante en concreto"

        r=self.db.exe("SELECT * FROM estudiante WHERE cedula="+cedula)
.
.
.
class DatosEstudiante:
    "datos basicos de un estudiante - estructura necesaria"
    #c: cedula, n:nombre
    def __init__(self,c,n):
        "Datos de estudiante, parametros: cedula,nombre"
        self.cedula=c
        self.nombre=n
        self.materia_y_notas={}
        self.materia_y_asistencias={}
.
.
.
```

Python tiene muchas formas de definir clases y crear objetos, se ha usado la forma primitiva. De acuerdo al diseño, se tiene un buscador de estudiantes, representado por la clase buscadorEstudiante. Una instancia de esta clase tiene el poder de buscar y devolver datos de estudiantes, los cuales son tipos de datos representados en la clase DatosEstudiantes. Podemos notar que existe llamadas al objeto self.db que es nuestro wrapper a la base de datos. Este objeto acepta sentencias SQL estándar.

El código fuente completo de la librería la puede consultar en el apéndice A.3.3

Construcción de un layer de acceso a sqlite El layer de acceso a sqlite se encuentra dentro de *./lib* del directorio principal de la librería cclib. El script principal de acceso a los datos se llama dbsqlite.py, tiene estos puntos reelevantes:

```
class DB:
    dbname="/var/tmp/callcenter.db"
    resultado=[]
    def __init__(self,dbname=None):
        if dbname is None: raise(DBException)
        self.dbname=dbname
        self.conn = sqlite.connect(dbname)
        self.cursor = self.conn.cursor()

    def exe(self,s):
        self.cursor.execute(s)
        self.resultado = self.cursor.fetchall()
        return self.resultado

    def run(self,s):
        out = self.cursor.execute(s)

        self.conn.commit()
#        print "DEBUG DB.run:",out
    def cerrar(self):
        self.cursor.commit()
```

En el archivo dbsqlite.py se debe definir una clase llamada "DB" con los métodos globales *exe* y *run* para consultas SQL hacie el motor, el método *exe* retorna los datos haciendo un "fetch" de todos los resultados de la consulta. El resultado es un tipo de dato *list*, para mayor información de las características de los tipos de datos del lenguaje python le sugerimos revisar (Ascher 2003).

El código completo de la librería de acceso a sqlite la puede encontrar en el apéndice A.3.4.

5. Analisis de factibilidad

5.1. Introducción

El presente análisis de factibilidad tiene la intención de explorar desde diferentes puntos de vista la posibilidad de implementación del Sistema de Consultas Automatizado usando un Callcenter, sistema a ser utilizado como una posible solución a la consulta de información y diferentes aspectos de caracter académico personal de todos los estudiantes del AEIRNNR.

Una tecnología de uso de sistemas de interacción usando una llamada desde un teléfono convencional, la utilización de componentes de Software Libre, el análisis y diseño de la solución de acuerdo a estándares, procedimientos y metodologías de creación de Software, hacen que sea una solución válida a ser implementada en el AEIRNNR. El presente análisis de factibilidad persigue acumular la suficiente información de juicio para establecer el costo/beneficio, aspectos de carácter técnico como: características de hardware, y aceptación social referentes a la implementación de esta solución en el AEIRNNR.

Generalmente, suele pensarse que el diseño de la configuración de una solución informática, solo depende del volumen de información a procesar, y que una vez cuantificada esta variable el responsable del proyecto decide cuanto equipo comprar. Sin lugar a dudas, esto es un grave error, dado que el volumen de la información, como parámetro, incide en la evaluación de las velocidades de cada dispositivo y en la forma de almacenamiento de los datos, pero no determina el diseño de la solución informática, que necesariamente estará condicionada a las políticas que encuadran al proyecto.

Como se ha visto, la solución informática debe contemplar los aspectos político/sociales que la condicionan, los volúmenes, las velocidades y capacidad de los dispositivos, y finalmente los aspectos técnicos referidos al hardware y software.

Desde los inicios del presente trabajo se ha desentrañado mediante el análisis y recopilación de requerimientos el "modelo del negocio" que en resumen se refiere a la implementación de un Sistema de Consultas Automatizado. Asumido el hecho de que el análisis y recopilación de requerimientos forman parte de este trabajo, queda sobre entendido este modelo, el cual deberá ser tomado en cuenta para un Plan Estratégico del AEIRNNR con los objetivos planteados en dicho plan.

5.2. Factibilidad Técnica

El límite de clientes concurrentes que podría soportar el callcenter en mención está determinado por el número de interfaces conectadas al PBX del AEIRNN. En una configuración normal de PBX se podría tener de 2 a 4 lineas en configuraciones PBX con centrales pequeñas.

El número de lineas que se podrá destinar para el callcenter se puede estimar tomando en cuenta los siguientes factores:

1. El número de estudiantes del Área

2. El número de unidades temáticas y productos acreditables que conforman cada módulo según el SAMOT

3. Los horarios diurnos y vespertinos

4. La frecuencia de actualización de notas para los estudiantes

5. El tiempo que toma realizar una consulta.(1 minuto aprox)

Se ha estimado que el tiempo promedio que toma realizar una consulta es de cerca de 1 minuto por estudiante. En un ambiente de 300 estudiantes, se estarían despachando solicitudes usando dos interfaces en un promedio de dos horas y media, al agregar una nueva extensión al callcenter se podría bajar ésta carga a cerca de una hora y media. Todo esto es muy subjetivo y se está suponiendo que *todos* los estudiantes realicen consultas *a una misma hora*, pero es una idea que podría optimizar el servicio prestado por el callcenter.

Determinación del hardware mínimo Para poder estimar la carga máxima que podría soportar el Callcenter, se ha recurrido a investigaciones de escenarios parecidos de personas usando equipos similares en otros países, de donde se ha rescatado el siguiente extracto traducido al español:

```
Dimensionando un sistema Asterisk

Las preguntas típicas realizadas en la lista de discusión
asterisk-user son:

Que tan veloz/grande debería ser mi máquina para servir mis necesidades?
Cuantas llamadas simultaneas puede manejar Asterisk?
Cuantas llamadas concurrentes?

Aqui encontrará una colección de respuestas promulgadas en asterisk-users:

Nota: Cuando agregue a esta lista, por favor proveer CPU y Uso de Memoria,
no promedio de carga. Estas son métricas mucho mas concretas y tienen mucho
más relevancia cuando se compara diferente hardwared y sistemas operativos.

Las entradas estan sorteadas por tamaño del CPU.
    * para CPUs viejos y lentos revise las recomendaciones de hardware de
      Asterisk
    * Gumstix-based SIP-to-IAX proxy server pueden routear cerca de 40 llamadas
      concurrentes (sin transcoding, Gumstix 400XM, voicemail en Tarjetas CF)
    * Pentium 133 MHz, 16 megs de ram: Manejan hasta 3 llamadas concurrentes
      con el protocolo SIP antes de que la calidad se degrade, no hay información
      acerca del transcoding
    * Linksys NSLU2 aka "slug" (Dispositivo NAS por $79): Maneja cerca de 4
      llamadas SIP con g711 o gsm (pero no g729) en su procesador IXP400
    * Pentium 1, 166mhz, 32meg ram: Satisfactoriamente arranca 4 llamadas
      SIP con codec g711
    * Pentium II 233/64 RAM: 2xBRI (=4 ISDN channels) 20 dispositivos SIP.
```

Este es un extracto del contenido de la página web http://www.voip-info.org/wiki-Asterisk+dimensioning del sitio *voip-info* sitio que recolecta y documenta

Comparativa de hardware mínimo

	Opción 1 línea	*Opción 2-4 líneas*	*Opción >4 líneas*
Procesador	Pentium II	Pentium III	Pentium 4
Memoria	128Mb Ram	256Mb Ram	512Mb o más
Almacenam.	20Gb	20 Gb	20Gb o más
Interfaces FXO	TDM400P	TDM400P	Adtran (TA) 750s

Cuadro 2: Comparativa de hardware mínumo

Configuraciones de Equipos

	Opción 1 linea	*Opción 2-4 lineas*	*Opción >4 lineas*
Procesador	AMD Turion 1.8Gz	Intel Core2 duo 1.8Gz	Intel Core2 duo 1.8Gz
Memoria	128Mb Ram	256Mb Ram	512Mb o más
Almacenam.	20Gb	20 Gb	20Gb o más
Interfaces FXO	TDM400P	TDM400P	Adtran (TA) 750s
Precio Ref	$1000	$1200	$4000 o más

Cuadro 3: Configuraciones de Equipos por lineas telefónicas

información gratuita acerca de aspectos relacionados con Asterisk. En este extracto se puede apreciar una lista de sistemas funcionando en escenarios similares al determinado en el presente trabajo, en donde se detallan características de hardware que usan en sus soluciones tanto de PBX, Callcenter e IVRs. También es importante revisar la fórmula *Erlang*[20]

Comparativa de hardware: El hardware mínimo que se ha determinado será adecuado, en función al número de líneas telefónicas sugeridas para la solución se resume en el cuadro 2 :

En vista de que el hardware expuesto 2 ya no se fabrica y por consiguiente, ya no se encuentra disponible en el mercado de venta de equipos de computación, se ha agregado una configuración de equipos por cargas de líneas telefónicas usando equipos de bajo costo disponibles en el mercado (cuadro 3).

El hardware necesario para la interface entre el PSTN y el Computador (es decir, las interfaces FXO) se puede adquirir en el mercado local, las interfaces FXO pueden ser adquiridas de los fabricantes en http://www.digium.com, o importadas mediante alguna empresa de importación.

Debido a otros factores como la disponibilidad de equipos en el mercado, puesto que computadores Pentium III ya no se encuentra en el mercado con garantía y disponibilidad inmediata, como recomendación se ha puesto a consideración el hardware óptimo estimado:

1. Computador de última generación, puede ser el computador *de facto* que se comercializa en el mercado de PCs de escritorio, se recomienda procesadores AMD por su bajo costo y sus altas prestaciones, lo importante es que el mainboard cuente con suficientes ranuras PCI para poner interfaces FXO y

[20]Fórmula Erlang : Para calcular la capacidad planeada, es un modelo estadístico de complejidad media. Se adapta convenientemente para el cálculo de capacidad en servidores VoIP.

que sea de una marca reconocida como ASUS, o Intel, con más de 512Mb de memoria RAM, el espacio en disco esta directamente asociado con la cantidad de fonetos que se vayan a almacenar, se estima que con un disco de 60Gb alcanzará. Todo esto cuesta aproximadamente US$700.

2. Interfaces FXO: se pueden usar las que vende digium (www.digium.com) existen "combos" de 2 y 4 módulos FXO, e incluso bancos completos para conectar >12 lineas telefónicas. El kit que ofrece Digium para desarrolladores es suficiente para dos líneas y su costo es de US$200-US$300.

El límite de servicio es virtual!!!!, puesto que siempre se podrá extender el hardware y las lineas telefónicas que serían nuestras variables decisivas.

5.2.1. Selección de las tecnologías del software

Selección de software libre Como se expuso en el literal 2.5, el software libre es una revolución informática que dispersa el desarrollo de soluciones basado en colaboración de millones de desarrolladores por Internet. Los puntos más relevantes que a nuestro juicio superan opciones comerciales son las siguientes:

1. Al ser el software libre un producto de libre distribución, es más asequible para desarrolladores en países como el nuestro en donde la tecnología es relativamente costosa. El tener acceso a todo el software necesario para la implementación de nuestros objetivos implica costos relativamente bajos.

2. El software libre encuentra un hogar en los países "en vías de desarrollo" en los cuales el costo del software no libre es muchas veces *prohibitivo*

3. Se vuelve mucho más rentable para un desarrollador de software libre hacer "carrera" usando éste paradigma porque recibe un apoyo directo de toda la comunidad a nivel mundial y el desarrollador recibe un pago justo por su trabajo y esfuerzo.

4. Al tener a disponibilidad el código fuente, se lo puede corregir, modificar o personalizar. Esto es mucho más seguro que usar software cerrado que no admite modificaciones, que puede ser vulnerado y que tiene un lento proceso de actualización de seguridades.

Selección de Debian GNU/Linux A pesar de que existen otros sistemas operativos libres como: freebsd, hurd (no terminado aún) y otros no muy famosos sistemas operativos, se ha elegido GNU/Linux por la madurez que ha obtenido en los últimos años, porque el kernel tiene mejor soporte para el hard-ware necesario para Asterisk. Particularmente se ha usado Debian por ser un linux verdaderamente libre, por no tener una empresa con fines de lucro detrás de su desarrollo, por ser un linux para usuarios con conocimientos especializados y para desarrolladores. Una vez que las soluciones han sido implementadas, se adopta una modalidad de actualización perpetua, pues que la tecnología avanza a pasos agigantados y los conceptos informáticos evolucionan, es necesario tener

un sólido sistema de actualizaciones, corrección de bugs, mejoras, etc. Debian y su contrato social(Manual de referencia Debian www.debian.org/doc) garantizan que Debian será siempre libre. Para un mejor entendimiento de todo lo que esto implica, sugerimos revisar el contrato social[21].

Selección de asterisk Los sistemas de telefonía comerciales son parte de un grande y multinacional negocio monopólico que implica los avances tecnológicos en materia de telecomunicaciones y la necesidad del ser humano de mantenerse comunicado. Desde el siglo pasado se han venido observando monopolios de inmensas empresas de comunicaciones como AT&T y derivadas, han existido guerras antimonopólicas para poder "liberar" la tecnología. Siguiendo éste camino se encuentran empresas proveedoras de hardware de telecomunicaciones como Cisco, Alcatel, etc. Quienes han propuesto protocolos de comunicación propietarios, esto ha permitido de alguna manera el nacimiento de nuevos monopolios. Con el desarrollo de estándares de comunicación como SIP[22] y otros esfuerzos de la comunidad de software libre, nace el proyecto Asterisk(asterisk www.asterisk.org), las bondades ya se las ha enunciado en 4.5.1, sin embargo las principales razones para usarlo son:

1. Incursionar en nuevas tecnologías relacionadas con nuestra carrera y que se encuentran a nuestra disposición gracias al software libre.

2. El costo de incursionar en ésta tecnología es ínfima y no se compara a los costos de soluciones privadas como Panasonic, Cisco, Alcatel, en donde soluciones similares están por sobre los US$30000.

3. El haber incursionado en este nuevo tipo de soluciones abre un espectro inmenso de nuevas posibilidades de trabajo y gracias a ello permite ofrecer nuevas herramientas con posibilidades de apertura en nuevos mercados de servicios y mejora nuestro nivel de vida.

4. La satisfacción personal de poder construir con nuestras propias mentes soluciones profesionales complejas y de alto nivel.

Selección de Python Python al igual que Perl, Bash, Php y Tcl son lenguajes de programación interpretados, no necesitan de compilación[23], sin embargo se ha optado por Python para este proyecto frente a Perl, Bash, Php y Tcl por las siguientes razones:

1. Tipado dinámico: Python no requiere una declaración explicita de tipos de datos y un tipo de dato puede cambiar dinámicamente.

[21] Debian contrato social: http://www.debian.org/social_contract.es.html

[22] (Session Initiation Protocol)(Van Meggelen 2005) un protocolo abierto de comunicación que hace posible tecnologías como VoIP, esta declarado en el RFC3261, ver : http://www.ietf.org/rfc/rfc3261.txt

[23] aunque en Python si se realiza una compilación a código intermediario al estilo vm de Java

2. Tipado fuerte: Reduce la necesidad de tener varios operadores (como en Perl) para trabajar con diferentes tipos de datos.

3. Que sea multiparadigma, esta posibilidad permite elegir la programación funcional u orientada a objetos, según conveniencias.

4. Lenguaje Interpretado, evita la necesidad de compilación aunque exista de por medio una pre-compilación a código máquina por parte de python. Existen herramientas para producir binarios y optimizar la velocidad dependiendo de la plataforma.

5. Las variables se liberan automáticamente mediante un proceso inteligente de recolección al estilo "garbage collector" de Java.

6. El juego de librerías de Python es extenso e incluye bindings para bases de datos e incluso para asterisk. Perl y Php son una buena elección en este punto, Perl tiene una amplia librería llamada CPAN que goza de muy buena fama, lo mismo sucede con Php.

7. El soporte orientado a objetos de Python es nativo, mientras que el de Perl se basa en namespaces que resultan un tanto incómodos, Php tiene un buen soporte para programación orientada a objetos.

8. A pesar de que Perl estuvo mucho antes en los ambientes shell, Python está surgiendo como un interesante candidato a reemplazar Perl en ambientes de sistemas operativos, Php resulta tedioso de instalar y usar en ambientes shell.

9. La facilidad de uso de Python es uno de sus principales fuertes, puesto que Perl resulta demasiado abstracto, y medianamente igual Php

10. Otro destacado en los ambientes de consola es Tcl, sin embargo su juego de utilidades limitan demasiado su campo.

Se ha en Python frente a opciones comerciales como Java, Visual C++, Visual Basic, etc, por razones muy obvias, una de las principales es de que estos ambientes son muy "pesados" y requieren equipos poderosos para su funcionamiento, mientras que Python funciona en muchísimas arquitecturas
(Ascher 2003).

Python es un lenguaje relativamente nuevo, que integra la las mejores características de lenguajes como: Perl (expresiones regulares), Java(administración de memoria y maquinas virtuales), Lisp(manejo de listas y tipos de datos avanzados), SmallTalk(Orientación a objetos). Python se encuentra en constante evolución, los nuevas improvisaciones de éste lenguaje incluyen frameworks para desarrollo rápido, desarrollo para el web, y muchísimas herramientas como IDEs, editores, debugers, etc.

Selección de tecnología VoIP Se presagia que la tecnología VoIP (voz sobre Ip), pronto se volverá tan popular que reemplazara a la vieja tecnología telefónica PSTN con más de 100 años de existencia. VoIP y Herramientas como Asterisk serán el futuro de las telecomunicaciones en un mundo no muy distante, ya el nacimiento de soluciones como la que se ha planteado aquí y de muchas otras que se pueden derivar de éste trabajo implican una revolución tecnológica que llevará al viejo y obsoleto aparato telefónico a nuevos horizontes. Era de esperarse que la tecnología de comunicación en tiempo real alcance Internet, VoIP era un paso obligado a las distintas revoluciones como la súper autopista de la comunicación o la revolución tecnológica que auguran los expertos. Es en Internet en donde se va a jugar nuevas batallas por el control de las comunicaciones, la tecnología VoIP junto con Asterisk son nuestros aliados en un país atrasado tecnológicamente y con necesidades imperantes de comunicarse.

5.3. Factibilidad Operacional

Se ha determinado la factibilidad operacional del presente proyecto, enmarcados en los siguientes aspectos:

Complejidad: Tanto la interface de administración como la de consultas del callcenter está basado en llamadas telefónicas, desde este punto, existe un menú explicativo que se encarga de encaminar al llamante paso a paso para conseguir la información que desea. Un sistema simple y efectivo que reduce al máximo la posibilidad de fallo, está basado en una lógica con recuperación de errores en el caso de que el usuario haga un ingreso erroneo, usando un bucle inteligente que analiza la entrada del usuario y delega el trabajo de consultas a los subsistemas que se han desarrollado, los cuales han sido testeados y su comportamiento es mutable dependiendo del ambiente en el que se están ejecutando. Esta granulidad ha permitido depurar eficientemente los scripts y sumarlos hasta conseguir la solución esperada.

Resistencia al cambio: No existe cambio en el sentido estricto, simplemente, se ha optimizado la forma en la que los estudiantes consultan sus notas, como ya se expuso al inicio del presente trabajo, existen otras posibilidades que se podrían adaptar para consultar la misma información, sin embargo, la innovación tecnológica, la practicidad, la exploración de otras opciones desde el punto de vista académico y económico, juegan un papel importante al momento de decidir por la solución de uso del Callcenter Automatizado.

Adaptabilidad: Desde la perspectiva del profesor, que es quien en primera instancia genera la información referente a las notas y asistencias de los estudiantes, se hace imprescindible el uso de un sistema académico que facilite el ingreso de la información que luego será explorada por el Callcenter, el proceso de alimentación de datos de estudiantes al callcenter deberá ser un proceso transparente. La adaptabilidad al sistema, de los interesados en la información de notas y asistencias no es un requerimiento directo de este proyecto, puesto que, al disponer de una base de datos centralizada de

Costo de llamadas

llamada	*costo US$*
llamada telefónica local	$0.01
llamada telefónica interprovincial	$0.03
llamada telefónica regional	$0.025
llamada telefónica celular	$0.15

Cuadro 4: costo de llamadas entrantes

notas y asistencias por parte del AEIRNNR la cual es alimentada por los profesores de las diferentes asignaturas, el Callcenter automatizado toma los datos de esta fuente. Sin embargo, para fines de pruebas se desarrollará una herramienta gráfica que permita inyectar información a la base de datos de manera práctica.

Obsolencia: Obsolencia de la solución, se refiere al hecho de que, todos los sistemas tienen un ciclo de vida, como ya se analizó en los sustentos teóricos del presente proyecto, la tecnología telefónica tiene más de 100 años de uso y vigencia, con muy escasas modificaciones, las cuales no han variado el comportamiento esencial de dicho equipo. Es muy probable que la tecnología telefónica tal como se la conoce, nos siga acompañando por muchos años más, ya se han presagiado importantes cambios, pero ninguno tiene una esencia significativa como para que tilde de obsoleto a la presente solución. De hecho, la tecnología telefónica en un futuro muy cercano se potenciará de nuevas cualidades y alcances que la mantendrán en auge e innovación continua. Al ser la presente solución modular y adaptable, no será muy complicado adaptar la modularidad de ciertas partes para que colaboren con nuevas fuentes de datos, los estándares y la filosofía abierta del Software Libre permitirán dar mantenimiento, actualización y adaptación del presente trabajo a nuevas soluciones, ayudando a la perpetuidad del sistema.

5.4. Factibilidad Económica

Análisis de costos referentes al uso del Callcenter Automatizado: La solución y su respectivo impacto económico, dependen de la configuración final de lineas, de concurrencia de llamadas, y de otros aspectos analizados anteriormente. Se ha hecho un análisis referente al costo de uso tanto para el usuario final como para el AEIRNNR, y el costo referentes a la implementación del presente proyecto.

El costo por uso del sistema se reduce a las siguientes cantidades[24]:

Llamada telefónica local: Una llamada telefónica local tiene un costo de $0.01 por cada minuto más impuestos.

Llamada interprovincial Una llamada interprovincial tiene un costo de $0.03 por un minuto.

[24]Los costos han sido consultados a la empresa Pacifictel de la ciudad de Loja.

Llamada regional Una llamada regional tiene un costo de $0.025

Llamada celular El costo por llamar desde un teléfono celular va desde los $0.1 dólares por minuto, dependiendo de la operadora celular.

Análisis de costos de implementación: De acuerdo al análisis técnico del hardware que se ha determinado que podría funcionar y suplir las necesidades del AEIRNNR, tomando en cuenta la infraestructura telefónica tanto de la central telefónica del AEIRNNR como de las extensiones disponibles, se han determinado los siguientes valores necesarios para la implementación del Callcenter Automatizado usando *Dos Lineas telefónicas*:

2 lineas telefónicas del proveedor de telefonía Local Pacifictel a un costo de $150 cada una, total $300.

1 computador de última generación con una interface FXO para dos lineas de entrada a un costo de $1200

1 costo de implementación, referente a la adaptación del software $700

Esto hace un total de $2200, no se ha tomado en cuenta el costo de creación de la solución que implica el desarrollo de la presente tesis.

Análisis de beneficios: En vista de que el producto final está orientado a satisfacer la necesidad de un usuario de consultas, esta solución se podría catalogar como una solución de "servicio" y por tanto intangible desde el punto de vista económico, sin embargo se ha recopilado algunas de las razones que otorgan un beneficio y valor agregado tanto al AEIRNNR como a la Universidad Nacional de Loja.

1. La calidad de la información
2. Información puntual
3. Mejor imagen de la organización, ayuda a atraer mas clientes
4. La experiencia obtenida del desarrollo
5. Beneficios indirectos que afectan al desempeño administrativo (comodidad al consultar las notas, liberar de trabajo a los empleados)

Como punto final respecto a la inversión económica estimada, se expone un resumen de costos (cuadro 5)para la implementación de la solución óptima usando un equipo en fucnión al número de líneas telefónicas e interfaces.

Los callcenters automatizados para consultas diversas está proliferando hasta volverse una herramienta válida e indispensable para diversas soluciones, como se lo detalla en la sección 6.1, la adaptación social de estas soluciones se da de una manera transparente y evolutiva como resultado del avance tecnológico natural de la sociedad. Es por esto que se cree válida la adopción de esta herramienta por parte de los estudiantes y personas interesadas en la información que el callcenter proveerá.

Resumen del Presupuesto por Implementación

Opción 1	*Descripción*	*Costo*
Computador	AMD Turion X2 1.8Ghz	$700
Memoria	128Mb Ram	
Disco Duro	20Gb	
Interfaz FXO	TDM400P	$300
Lineas Telef	1 linea	$150
Costo Implem.		$700
Total		$1850
Opción 2	*Descripción*	*Costo*
Computador	Intel Core2 Duo 1.6Ghz	$900
Memoria	256Mb Ram	
Disco Duro	20Gb	
Interfaz FXO	TDM400P	$300
Lineas Telef	2 lineas	$300
Costo Implem.		$700
Total		$2200
Opción 3	*Descripción*	*Costo*
Computador	Intel Core2 Duo 1.8Ghz	$1000
Memoria	512Mb Ram	
Disco Duro	80Gb	
Interfaz FXO	Adtran (TA)750s 24 puertos	$2436
Lineas Telef	4 lineas	$600
Costo Implem.		$700
Total		$4736

Cuadro 5: Resumen del Presupuesto para implementación

6. Discusión

Luego de haber terminado satisfactoriamente el análisis, diseño de la solución y haber probado los componentes de software que se desarrollaron para cumplir con los objetivos, se realizará un análisis crítico de los puntos más relevantes del presente estudio.

6.1. Análisis de la efectividad de la solución

Hemos realizado una investigación de soluciones para "realizar consultas usando callcenters" y poderlas comparar con nuestra solución, exponemos algunas de las principales soluciones comerciales que de alguna manera tienen una similitud con la nuestra.

6.1.1. Los proveedores de telefonía celular

Los proveedores de telefonía celular y los servicios que ellos ofrecen como la telefonía prepagada que implica la compra de tarjetas con "saldo" constituyen un ejemplo práctico del uso de consultas usando un callcenter o IVRs.

Gracias a estas tecnologías los proveedores de telefonía han podido masificar el uso del celular y crear verdaderas multinacionales empresas multimillonarias. La implementación de éste tipo de soluciones superan los cientos de miles de dólares y hace algunos años la tecnología parecía *asunto de magia*. Nuestras soluciones ponen este tipo de tecnologías al alcance de nuestras manos y para el aprovechamiento eficiente en nuevas ideas.

6.1.2. Los proveedores de llamadas de larga distancia

Se crearon en países desarrollados los sistemas de llamadas asistidas por una operadora automática que solicitaba el ingreso de un PIN numérico que se vende dentro de una tarjeta "de raspar" y luego ofrecía tiempo para llamadas a larga distancia, éste es otro ejemplo de una solución práctica usando consultas telefónicas y un callcenter. El costo promedio de estas soluciones van desde los US$40000 a US$100000.

6.1.3. Bancas electrónicas

Otro interesante producto de la tecnología con la que hemos desarrollado nuestro proyecto es el de *Banca electrónica* un servicio ofrecido por los bancos para la consulta de saldos, movimientos, estados de cuenta, etc. En donde se le ofrece al usuario con la comodidad de una llamada telefónica todo el estatus de su cuenta e incluso realizar movimientos y otros servicios avanzados. El costo de estas soluciones de acuerdo al mercado local se estima en alrededor de US$90000.

7. Conclusiones

- El callcenter automatizado resultado del análisis y diseño, y la solución de los requerimientos han dado como fruto una solución informática que se adapta plenamente.
- Se identificaron cuatro procesos académicos factibles de ser automatizados: Consulta de Notas, Consulta de Asistencias, Consulta de Novedades y anuncios para estudiantes, Consulta de Requisitos de Matriculación
- Las características del Call Center y su diseño, para satisfacer las necesidades de acceso automático a la información, se han desarrollado en el lenguaje UML, siguiendo los procedimientos formales de creación de software. Cumpliendo además con los estándares y usando el proceso de desarrollo Grapple. Se generaron los resultados del análisis del proceso de negocios del sistema mediante casos de uso. Durante el diseño se ha tomado en cuenta la posible integración del software resultante con otros sistemas informáticos existentes.
- Se estableció la factibilidad técnica, económica y social de acuerdo al análisis y diseño de la solución desarrollada, encontrándose que es una solución válida y factible.
- Los componentes de software necesarios para el funcionamiento coordinado del Call Center están conformes con la determinación de procesos, el análisis y diseño del callcenter, y de un proceso de ingeniería de software.
- Se crearon herramientas necesarias para integrar todos los puntos del callcenter y convertirlo en una solución eficiente. El software fue analizado en profundidad y el proceso de desarrollo se documentó en sus partes más relevantes.

8. Recomendaciones

Sobre la base de los resultados obtenidos en el presente trabajo, se considera pertienente plantear las siguientes recomendaciones:

- Considerando que en el presente proyecto se ha diseñado y evaluado la factibilidad de usar un Sistema de Consultas Utilizando un Callcenter Automatizado, el siguiente corresponde a la formulación de un *plan de implementación* de la solución, el mismo que comprende la preparación del presupuesto, la gestión del financiamiento, adquisión de equipos, la instalación y la puesta en funcionamiento, aplicando el software desarrollado.

- El motor de hardware central de la solución da como resultado un completo servidor que, a parte de los protocolos e interfaces que se han usado en éste proyecto, puede soportar otros protocolos propios del "ambiente VoIP", se podría tener como base el presente trabajo para la creación de nuevas soluciones que abarcan los siguientes aspectos:

  - VoIP implementación de protocolos libres de codificación de voz en tiempo real.
  - PBX extendidos usando redes ethernet y VoIP.
  - Conectividad en modo cluster para ambientes críticos, es decir, unir varios servidores como los que se ha diseñado en el presente proyecto, para trabajar en ambientes con cientos de llamadas concurrentes.
  - Creación de empresas telefónicas alternativas usando VoIP y redes públicas wireless

- La implementación del Callcenter Automatizado en el AEIRNNR constituye un "Proyecto Piloto" previa a la extención del uso en *TODA* la Universidad Nacional de Loja. La adaptación de la presente solución a un nuevo escenario como lo es la Universidad en su totalidad y sus respectivas Áreas Académico Admnistrativas, Carreras y Programas de formación dan pie a la formulación de nuevos proyectos.

Referencias

[Larman 2001] Larman Craig, Addison Wesley, Applying UML and Patterns, An Introduction to Object Oriented Analisys SECOND EDITION

[Schmuller 2002] Schmuller Joseph, PRENTICE HALL, Aprendiendo UML en 24 Horas

[Ariadnetraining 2001] Ariadne Supporting, 2001, UML Applied www.ariadnetraining.com.uk

[Shalloway 2003] Shalloway, SOFTWARE PATTERNS SERIES Design Patterns Explained,

[Senn 1998] Senn James A., Mc Graw Hill - SEGUNDA EDICION Análisis y Diseño de Sistemas de Información,

[wikipedia enciclopedia web] WikiPedia : http://www.wikipedia.org

[Manual de referencia Debian www.debian.org/doc] GNU/Debian : http://www.debian.org

[PBX Info www.pbxinfo.org] PBXInfo: todo acerca de PBX http://www.pbxinfo.org

[SQLite database system] SQLite library self-contained embeddable engine: http://www.sqlite.org

[asterisk www.asterisk.org] Digium y Asterisk web site: http://www.asterisk.org

[Agile Modeling www.agilemodeling.com] Agile Modeling, http://www.agilemodeling.com

[visualparadigm www.visualparadigm.com] Visual Paradigm resources http://www.visual-paradigm.com

[magickdraw resources] MagicDraw, Recursos y manuales de operacion http://www.magicdraw.com

[Abella 2003] Libro Blanco de Software Libre en España, 2003, http://www.libroblanco.com

[Kline 2001] David Kline with Kevin Kline, 2001, O'Reilly, SQL in a Nutshell

[Ascher 2003] David Ascher - Mark Lutz, 2003, O'Reilly, Learning Python Second Edition

[Van Meggelen 2005] Jim Van Meggelen - Jared Smith - Leif Madsen, 2005, O'Reilly, Asterisk: the future of Telephony

[VoIP www.recursosvoip.com] Recursos Voz sobre ip: http://www.recursosvoip.com

A. Apéndice

A.1. Lista de frases de fonetos

callcenter-adios.txt adios, que tenga un lindo dia

callcenter-admin-bienvenido.txt Acaba de ingresar al administrador del callcenter sea usted bienvenido. Para ingresar nuevos anuncios, presione uno. Para ingresar nuevos requisitos de matriculacion, presione 2. Para salir presione 9.

callcenter-anuncio-borrado.txt el anuncio ha sido borrado

callcenter-bienvenido.txt Bienvenido al callcenter del Area de Energia Industria y Recursos naturales no Renovables

callcenter-elestudiante.txt El estudiante con cedula

callcenter-ingresar-cedula.txt Por favor ingrese su numero de cedula

callcenter-instr-adminanuncios.txt para agregar un nuevo anuncios presione 1 para revisar y borrar anuncios presione 2 para escuchar los anuncios presione 3

callcenter-instr-adminrequisitos.txt administracion de requisitos. Para agregar un nuevo requisito presione uno, para borrar un requisito presione 2, para escuchar todos los requisitos presione 3. Para salir presione 9.

callcenter-instr-revisarmensajes.txt A continuacion se van a tocar los mensajes uno a uno, despues del beep puede presionar 3 para borrar el mensaje o 9 para salir

callcenter-noexiste-estudiante.txt El estudiante no existe

callcenter-nomasdatos.txt Ya no existen mas datos

callcenter-opciones-main.txt Para consultar notas, presione 1. Para consultar asistencias, presione 2. Para consultar los requisitos de matriculacion, presione 3. Para consultar anuncios, presione 4. Para salir del callcenter presione 9.

callcenter-opcion-invalida.txt Opcion invalida

callcenter-punto.txt punto

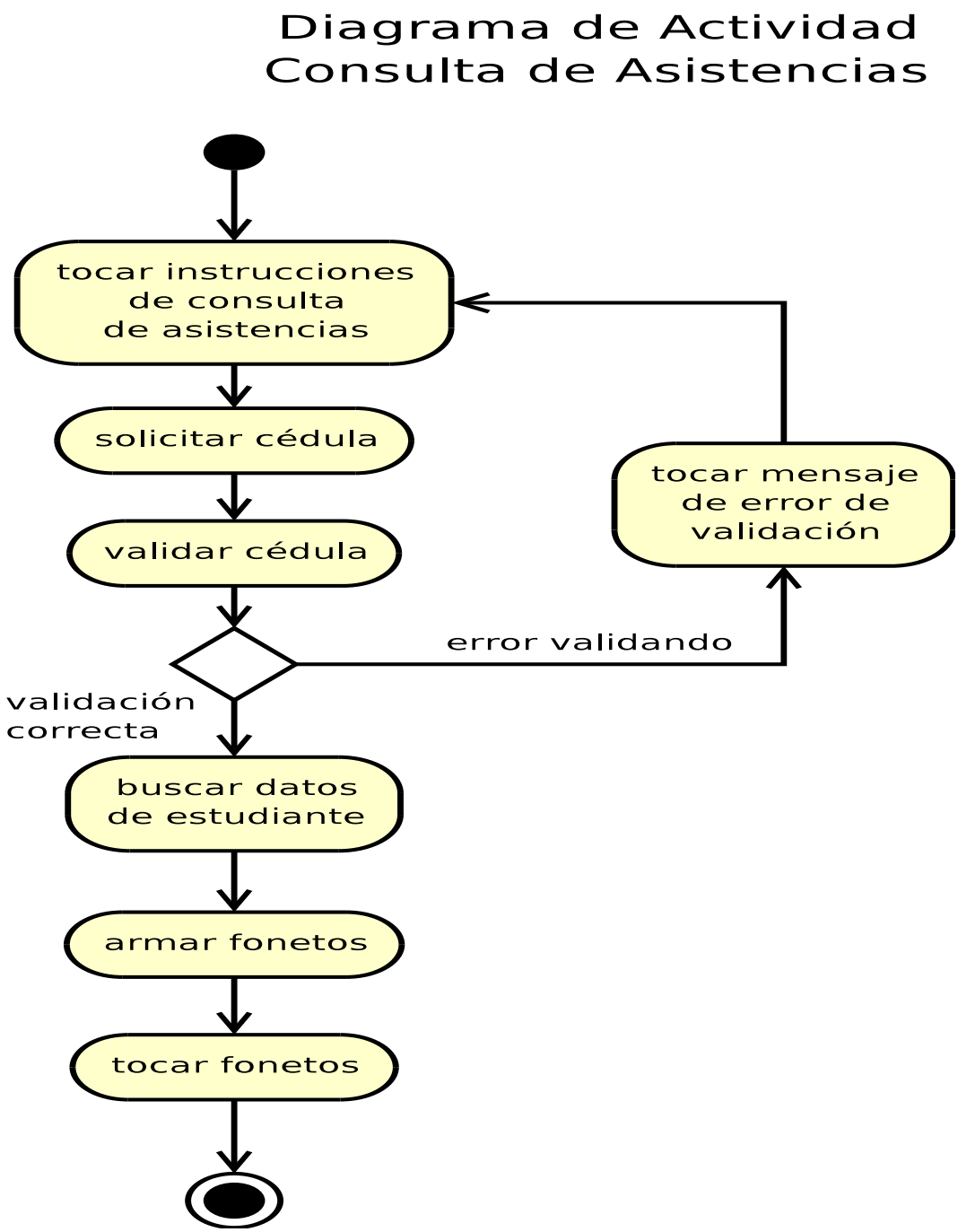
callcenter-requisito-borrado.txt el requisito ha sido borrado

callcenter-tieneasistencias.txt Tiene las siguientes asistencias

callcenter-tienenotas.txt tiene las siguientes notas

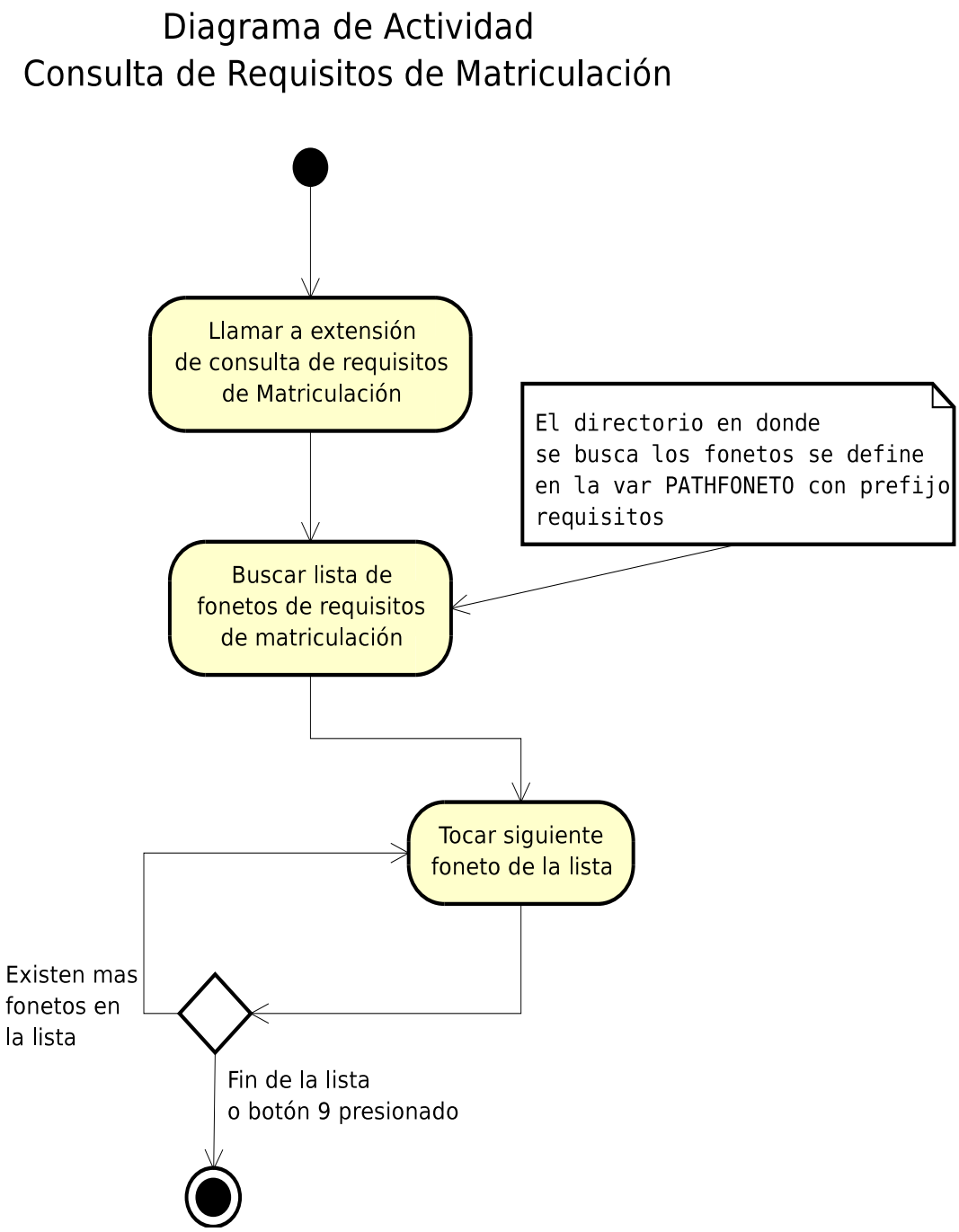
callcenter-volver-grabar.txt para grabar otro anuncio presione 1, para salir presione 9

A.2. Documentación de diagramas UML



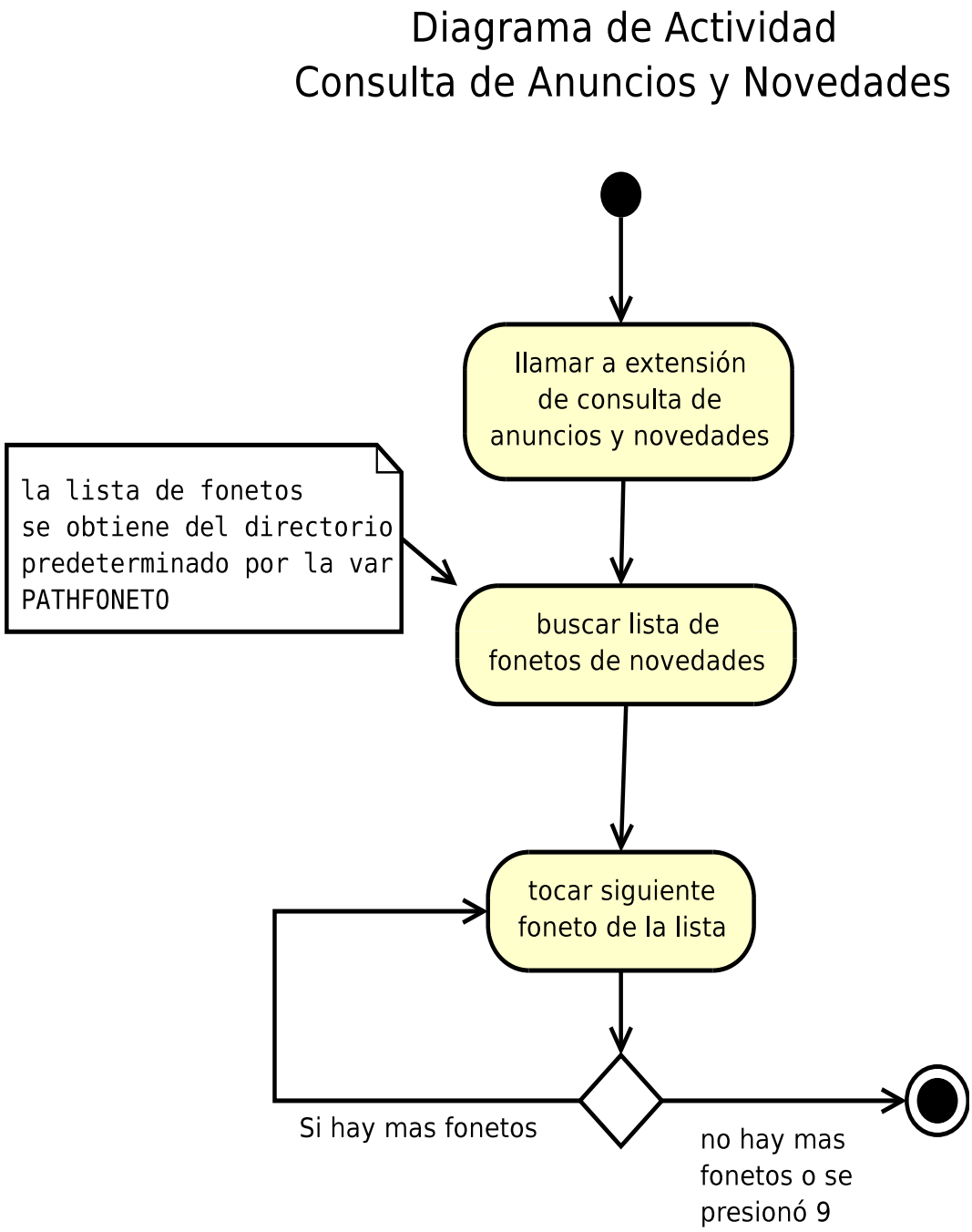
[a]

[a] Diagrama de Actividad - Consulta de Asistencias



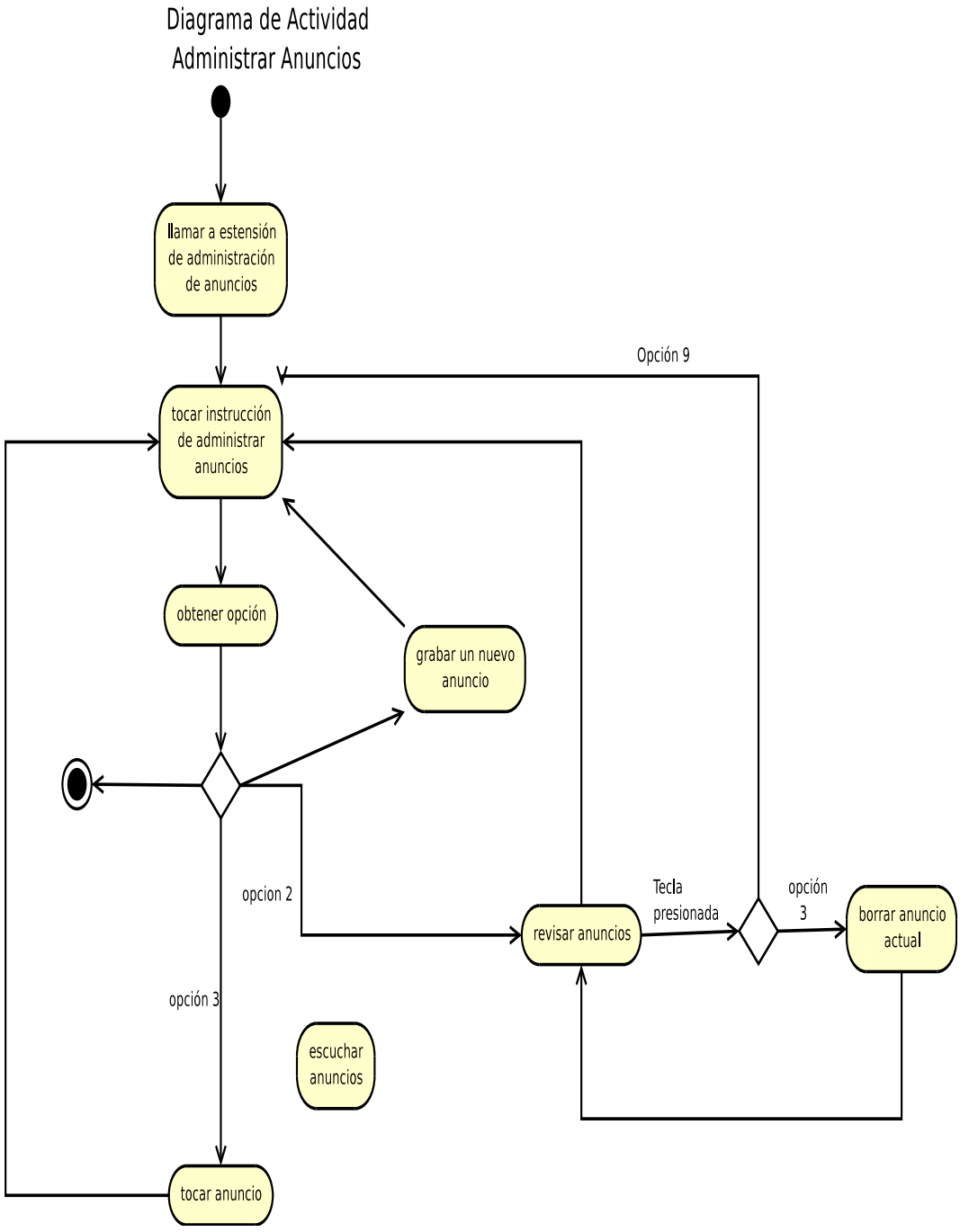
a

[a] Diagrama de Actividades - Consulta de requisitos de matriculación



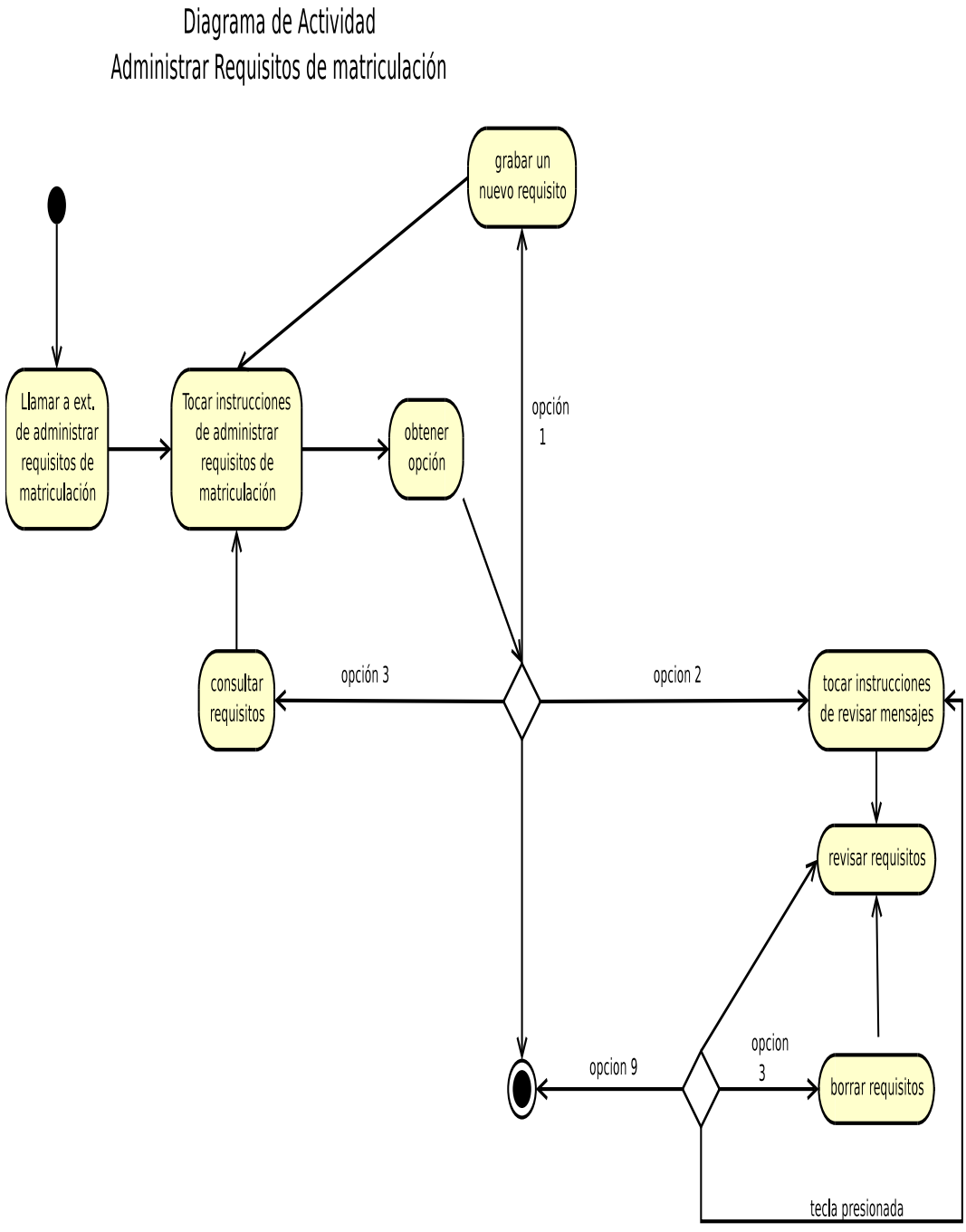
a

[a] Diagrama de Actividad - Consulta de anuncios y novedades



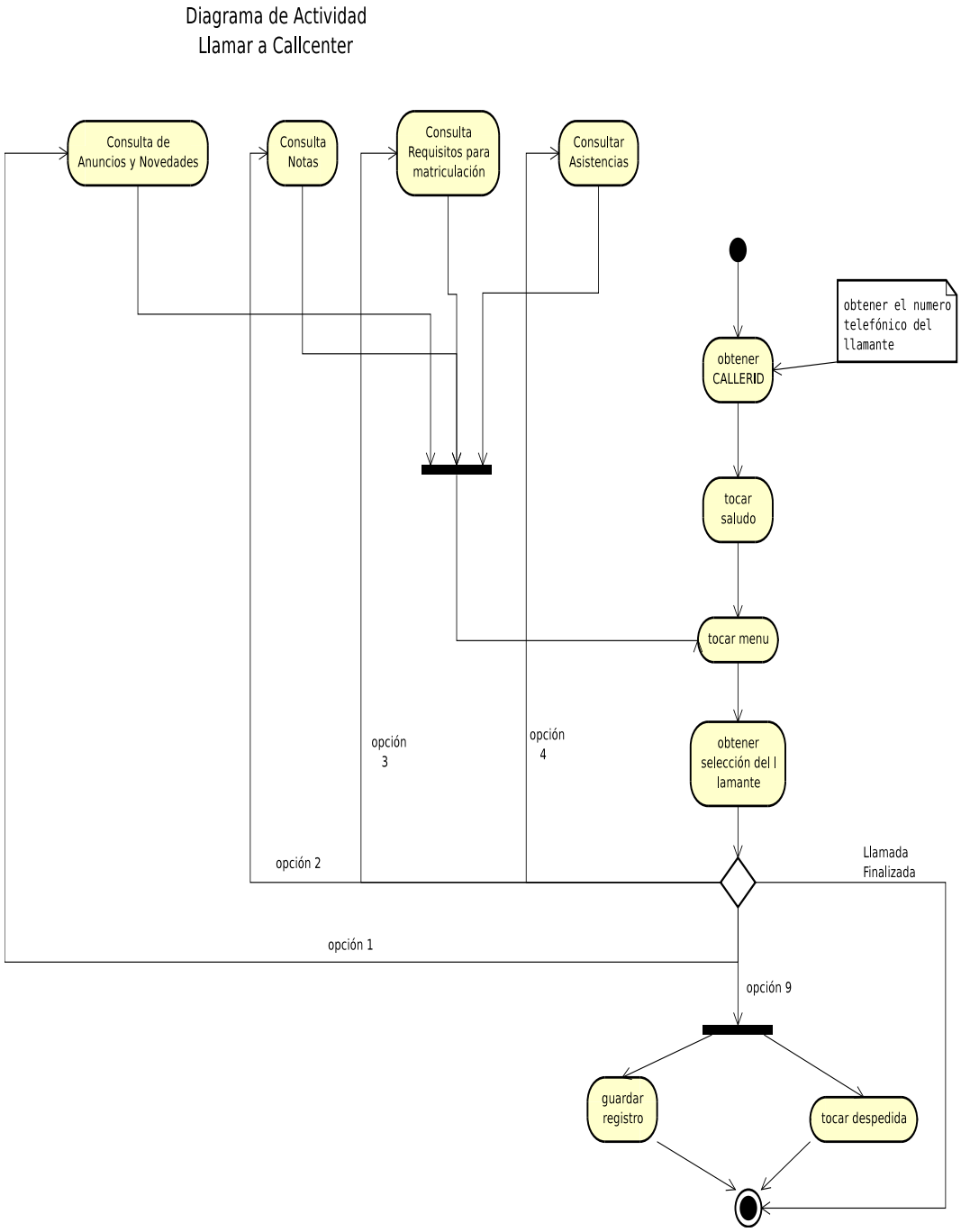
a

[a] Diagrama de Actividad - administración de anuncios



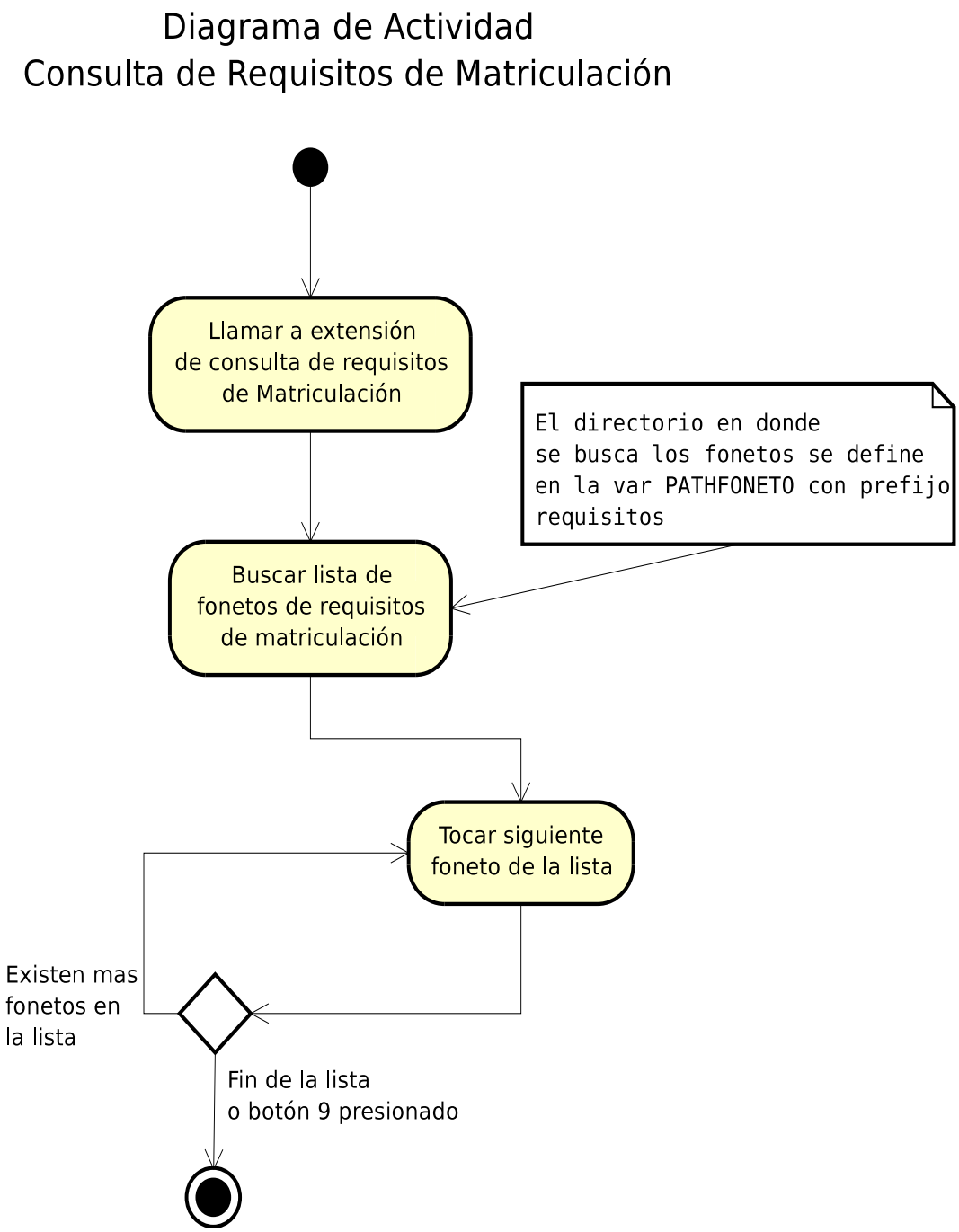
a

[a] Diagrama de actividad - Administración de requisitos de matriculación



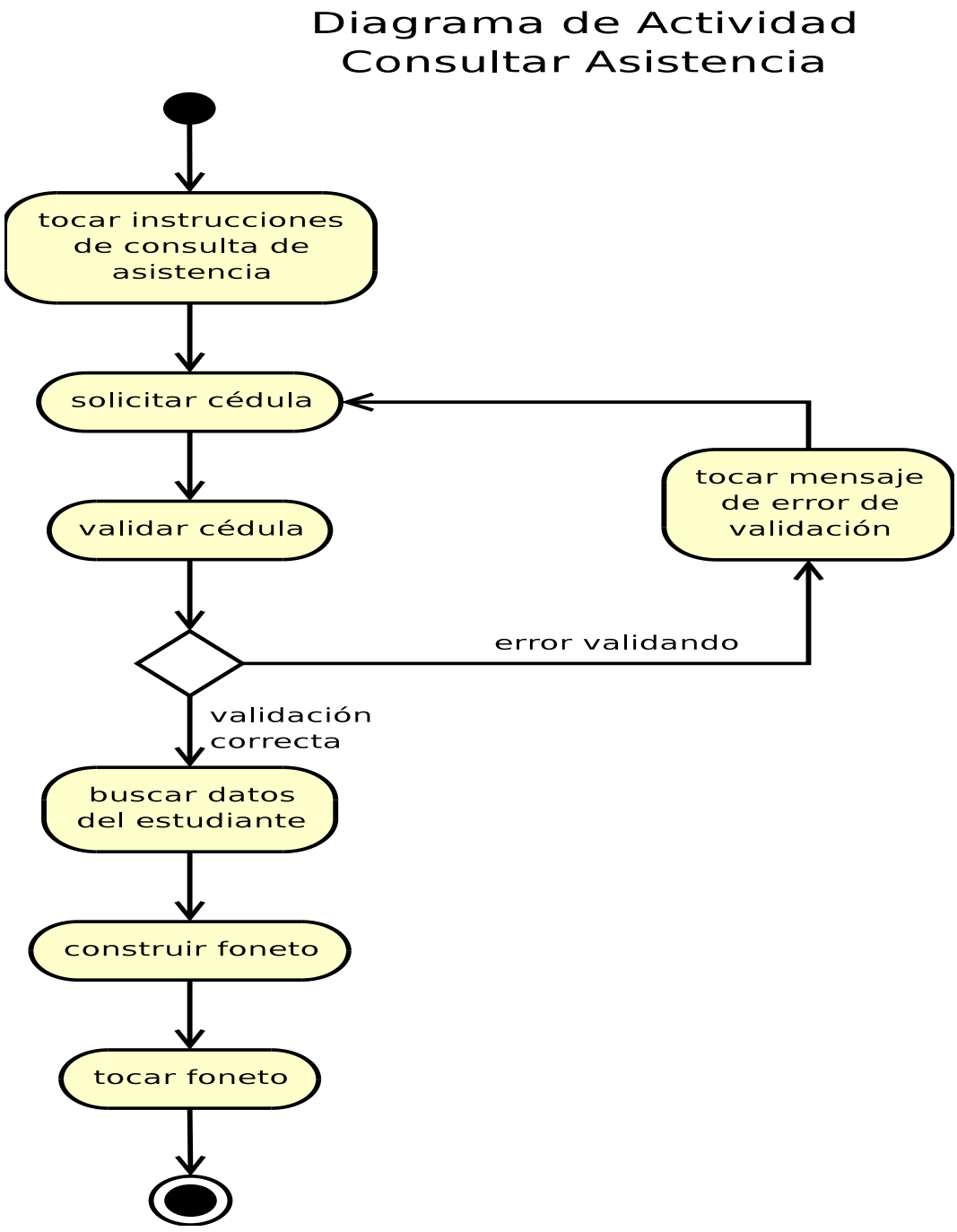
a

[a] Actividad - Llamar a callcenter



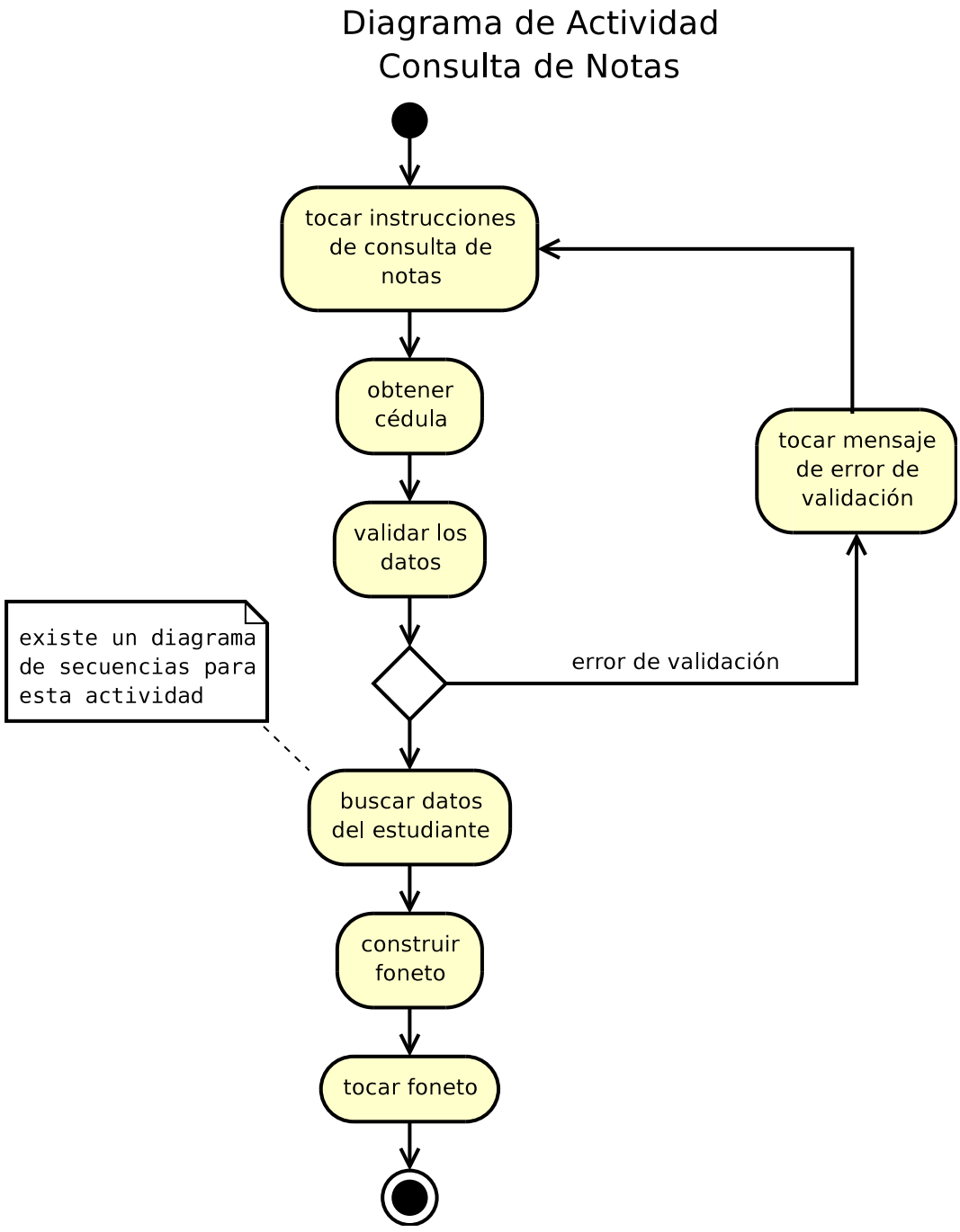
a

[a] Actividad - Consultar requisitos para matriculación



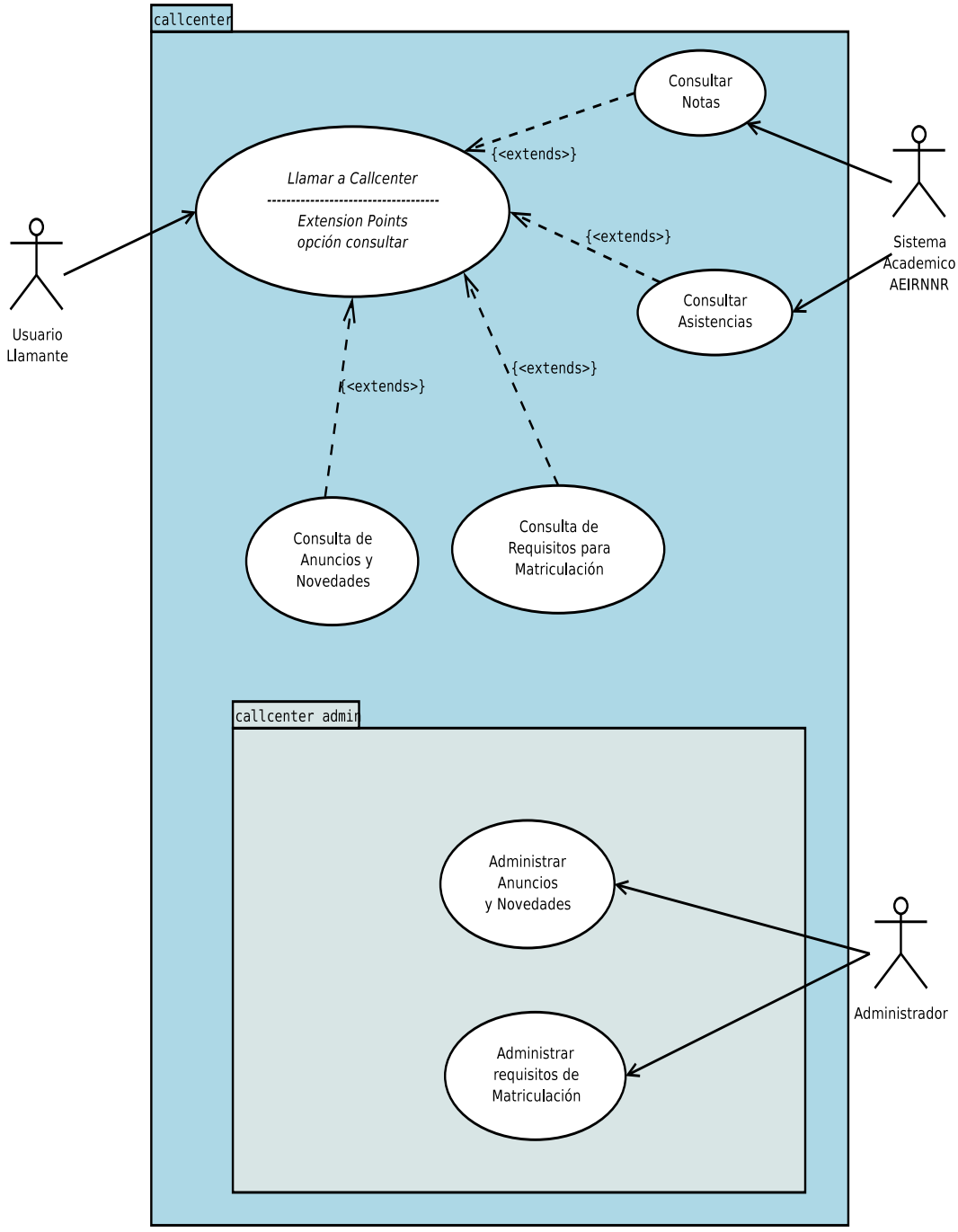
a

[a] Diagrama de Actividad - Consultar Asistencia



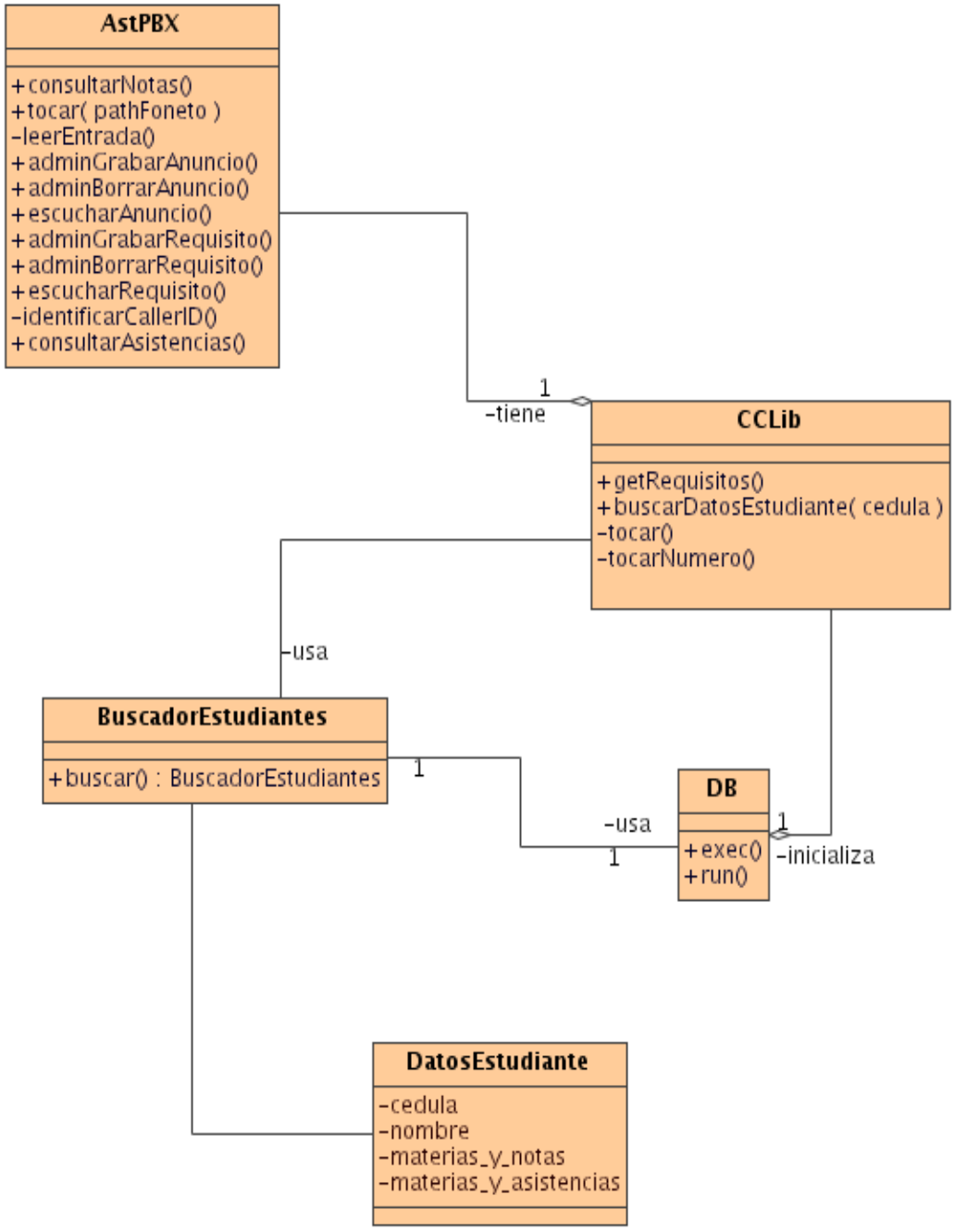
a

[a] Diagrama de Actividad - Consulta de notas



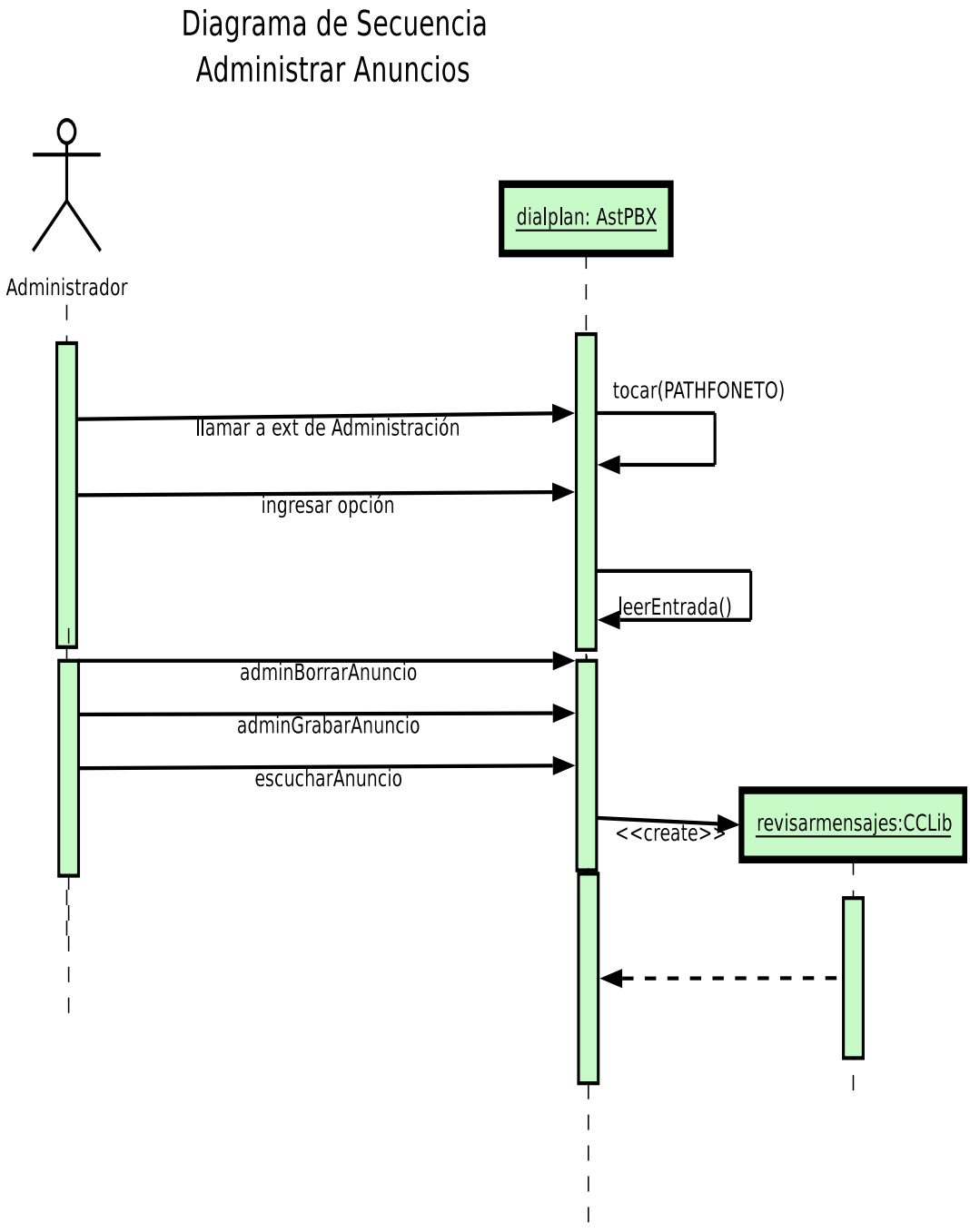
a

[a] Diagrama de casos de uso principal



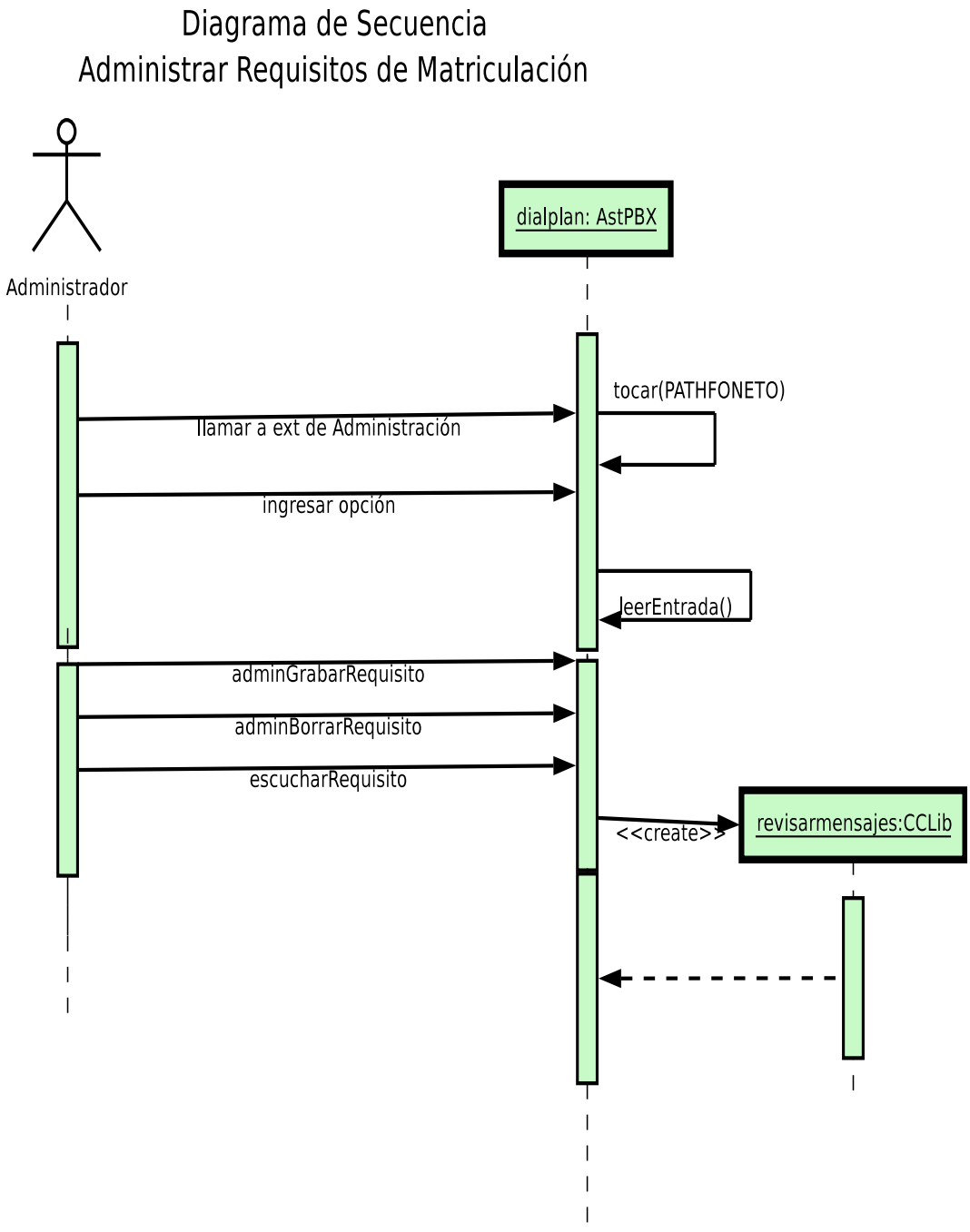
a

[a] Diagrama de Clases Principales



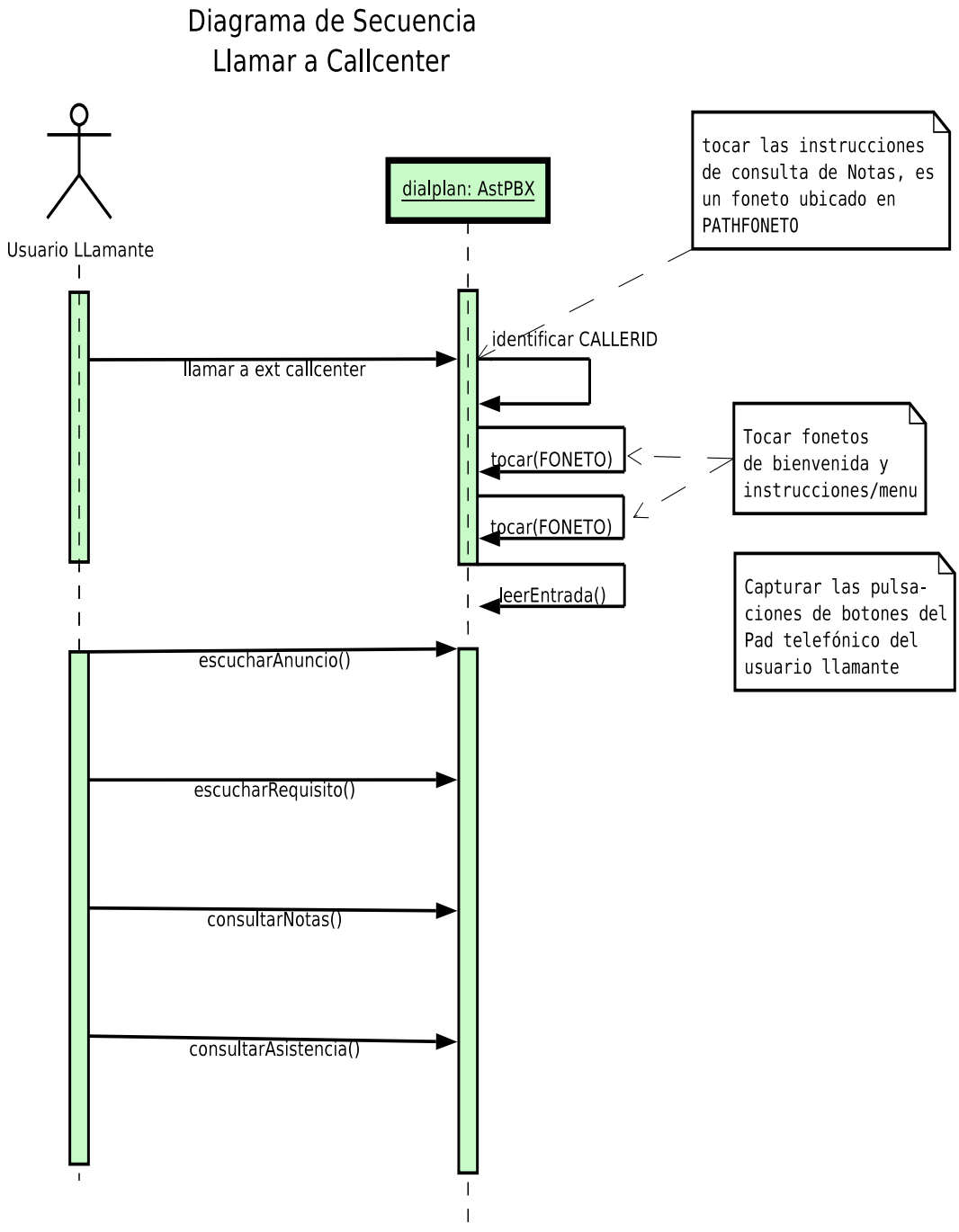
a

[a] Diagrama de Secuencia Administrar Anuncios



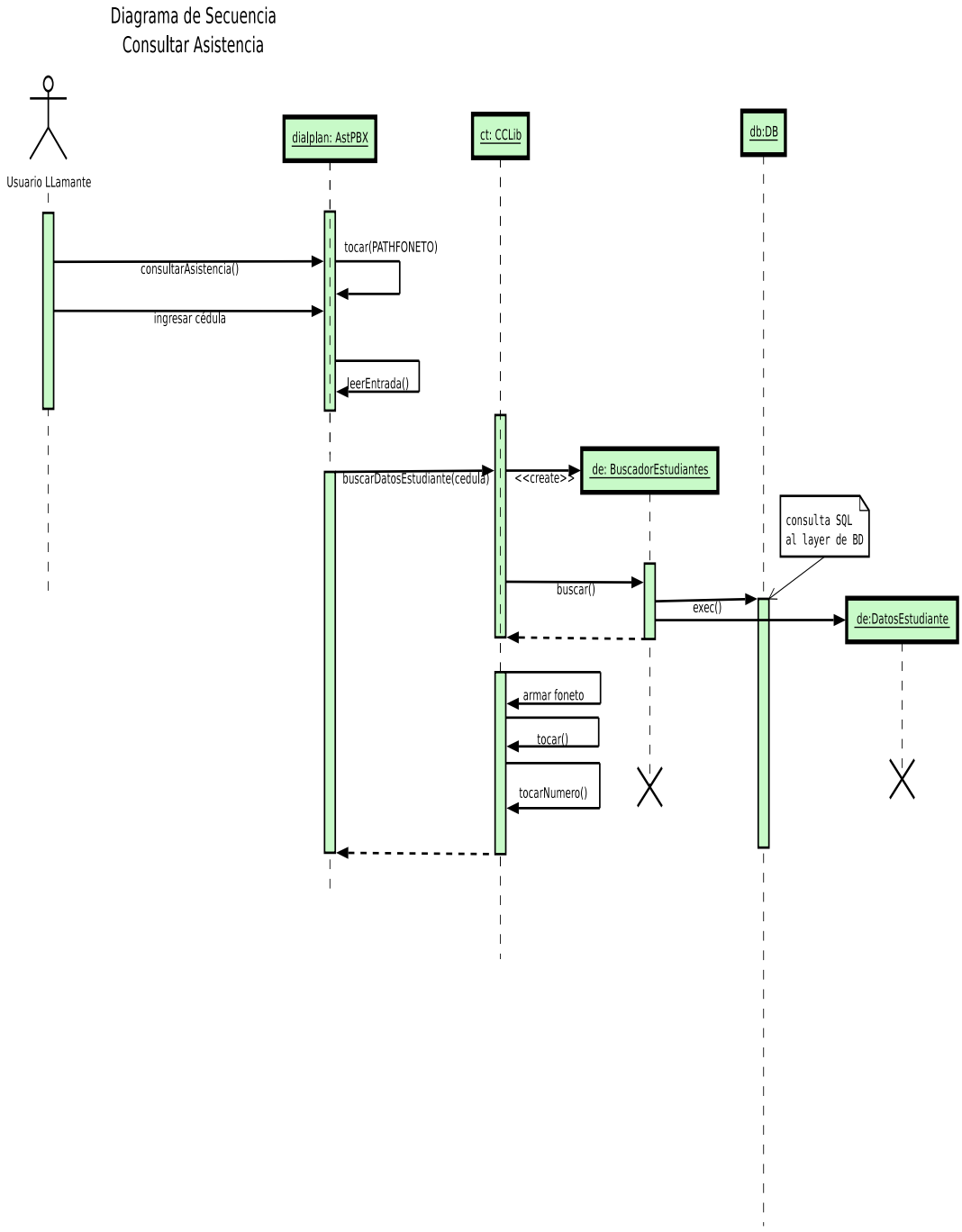
a

[a] Diagrama de Secuencia - Administrar Requisitos



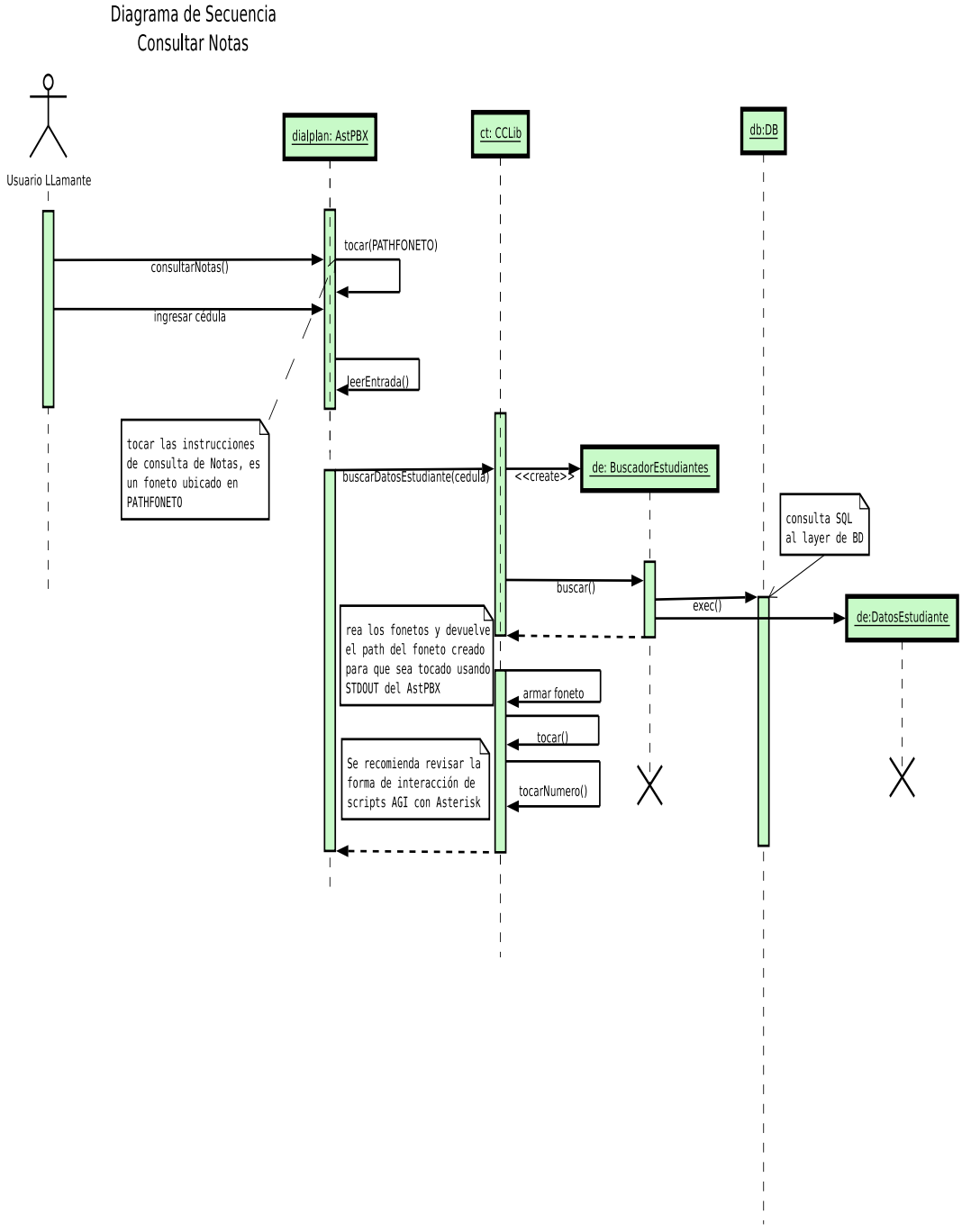
a

[a] Diagrama de Secuencia - llamar a callcenter



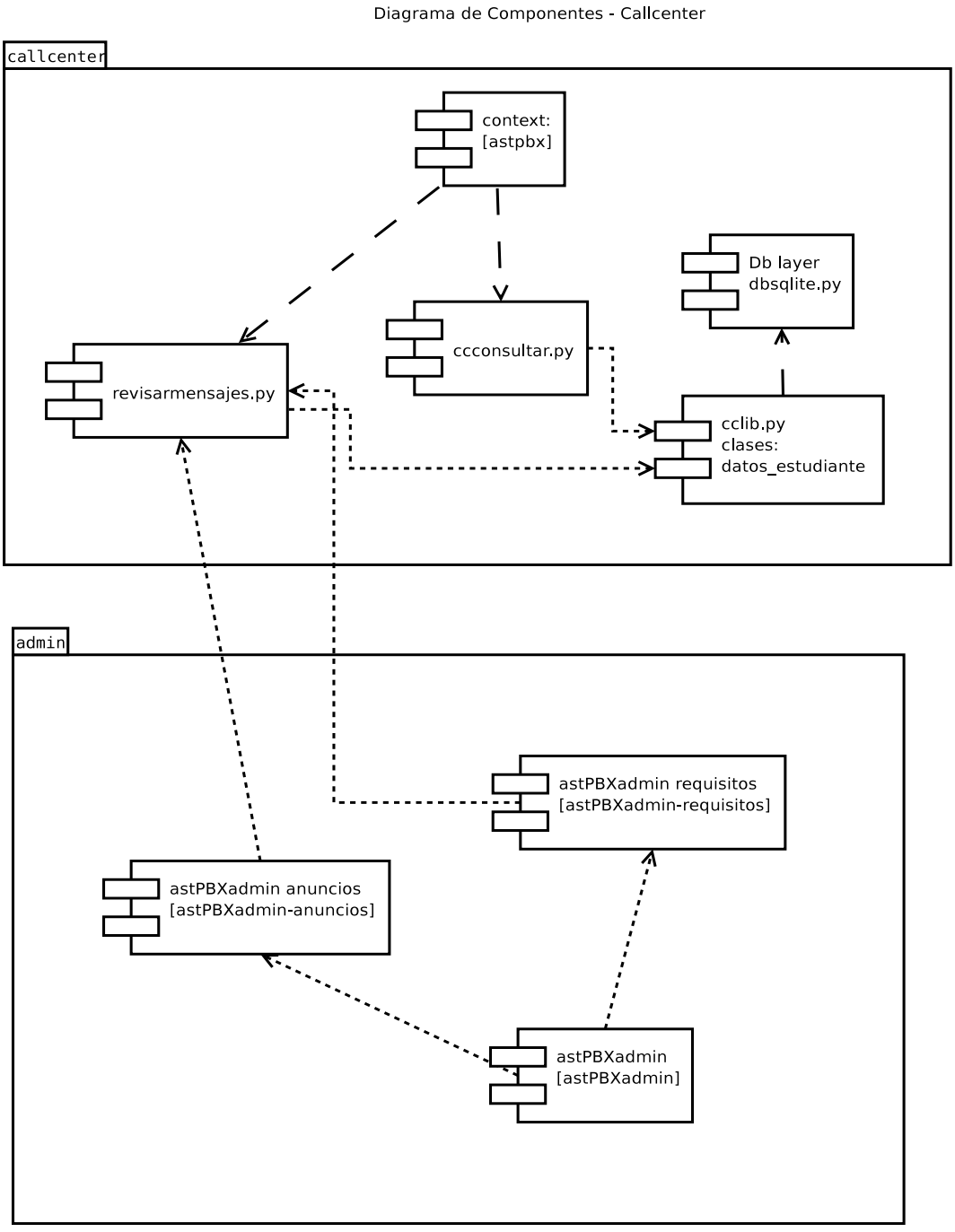
a

[a] Diagrama de Secuencia - Consultar Anuncios



a

[a] Diagrama de Secuencia - Consultar Notas



a

[a] Diagrama de Componentes

A.3. Código fuente de los programas desarrollados

A.3.1. esquema.sql

```
BEGIN␣TRANSACTION;
create␣table␣estudiante(
cedula␣INTEGER␣NOT␣NULL␣PRIMARY␣KEY,
nombre␣VARCHAR(150)␣NOT␣NULL,
direccion␣VARCHAR(50)␣DEFAULT␣'desconocido',
telefono␣VARCHAR(15)␣DEFAULT␣'0991231829'
);

create␣table␣materia(
idmateria␣INTEGER␣NOT␣NULL␣PRIMARY␣KEY,
nombre␣VARCHAR(150)␣NOT␣NULL,
horas_dictadas␣INTEGER␣NOT␣NULL␣DEFAULT␣100
);

create␣table␣notas_asistencias(
cedula␣INTEGER␣NOT␣NULL,
idmateria␣INTEGER␣NOT␣NULL,
nota␣DECIMAL␣NOT␣NULL,
asistencia␣INTEGER␣NOT␣NULL
);

create␣view␣datos_estudiante␣as
SELECT␣e.cedula,e.nombre,m.nombre,na.nota,na.asistencia,m.horas_dictadas
FROM␣estudiante␣e,␣materia␣m,␣notas_asistencias␣na
WHERE␣e.cedula=na.cedula
AND␣␣␣␣m.idmateria=na.idmateria;
COMMIT;
```

A.3.2. ccconsulta.py

```
 #!/usr/bin/python
# -*- coding: utf-8 -*-
#
# $URL: $Url$
# $Id: $Id$
# $LastChangedDate: $LastChangedDate$

# para el vi: scp programacion/python/callcenter/ccconsultar.py
# root@voip:/var/lib/asterisk/agi-bin/

"""
Este programa sigue las especificaciones del diagrama de secuencia y de actividad
 para el caso de uso consultar notas y consultar asistencias.
"""
import sys
import os
import re
import time

# definicion de globales
PATHFONETOAST="/usr/share/asterisk/sounds/callcenter/"
```

```
CONSULTA="notas" # puede ser notas o asistencias
CONFIGFILE="/etc/cclib.ini" # el path al config de la base de datos
#CONFIGFILE="cclib.ini" # el path al config de la base de datos
CCLIBPATH="/usr/local/cclib" # el path a la libreria cclib

def pdebug(data):
    """Escribir en el debug del sistema"""
    sys.stderr.write("DEBUG %s\n"%data)
    sys.stderr.flush()

def tocar(data):
    """Explicitamente toca un foneto"""
    # Send Asterisk a command
    sys.stdout.write("EXEC Playback %s%s\n" % (PATHFONETOAST,data))
    sys.stdout.flush()

def tocarNumero(params):
    """Explicitamente toca un numero de dos en dos"""
    # por restricciones del IVR tenemos que dividir los numeros grandes
    # lo vamos a hacer en pares para que el asterisk no se indigeste
    enteros,decimales=None,None

    # 1. Dividir el numero antes y despues de la coma
    if "." in params:
enteros,decimales = params.split(".")
    else:
enteros=params

    # 2. Partir enteros y decimales en grupos de 2 o individuales si es 0
    le=[]
    for i in xrange(0,len(enteros),2):
     if enteros[i]=='0':
le.append(enteros[i])
le.append(enteros[i+1])
else:
le.append(enteros[i:i+2])
    pdebug("esto es le:"+str(le))

    for cifra in le:
pdebug("SAY NUMBER %s \"\"\n" % str(cifra))
sys.stdout.write("SAY NUMBER %s \"\"\n" % str(cifra))
sys.stdout.flush()
time.sleep(1)

    if decimales:
ld=[]
for i in xrange(0,len(decimales),2):
     if decimales[i]=='0':
ld.append(decimales[i])
ld.append(decimales[i+1])
else:
ld.append(decimales[i:i+2])

pdebug("tocar punto\n")
tocar("callcenter-punto")
for cifra in ld:
```

```
     pdebug("SAY NUMBER %s \"\"\n" % str(cifra))
sys.stdout.write("SAY NUMBER %s \"\"\n" % str(cifra))
sys.stdout.flush()


def checkresult (params):
"""Evalua los resultados de una solicitud """
params = params.rstrip()
if re.search('^200', params):
result = re.search('result=(\d+)',params)

if (not result):
sys.stderr.write("ERROR ('%s')\n" % params)
sys.stderr.flush()
return -1
else:
# el primer grupo del re compilado
result = result.group(1)
pdebug("Result: %s Params:%s" % (result,params))
return result
else:
sys.stderr.write("FAIL (unexpected result '%s')\n" % params)
sys.stderr.flush()
return -2
pdebug("final de checkresult")

def leerIntro():
"""Capturar toda entrada inicial para dejar libre el stdin"""
env = {}
while 1:
line = sys.stdin.readline().strip()

if line == '':
break

key,data = line.split(':')
if key[:4] <> 'agi_':
#eliminar lo que no tenga agi_
sys.stderr.write("algo paso!\n")
sys.stderr.flush()
continue

key = key.strip()
data = data.strip()

if key <> '':
# rellenamos nuestro env
env[key] = data

pdebug("AGI SALIDA: \n")
for key in env.keys():
pdebug("-- %s = %s\n" % (key,env[key]))


def recibir():
    "recibir lo que haya en el stdin"
```

```
    res=sys.stdin.readline().strip()
    res = checkresult(res)
    return res

def pedir():
"""pedir data del asterisk"""
timelimit=5000
digitcount=1
prompt="you-sound-cute"
pdebug("GET DATA %s %d %d\n" % (prompt,timelimit,digitcount))
sys.stdout.write("GET DATA %s %d %d\n" % (prompt,timelimit,digitcount))
result = recibir()
pdebug("getnumber %s\n" % result)

if result:
return result
    else:
return -1


# inicio del script

try:
    CEDULA = sys.argv[1]
    CONSULTA = sys.argv[2]

except IndexError:
    pdebug("ERROR!: faltan argumentos")
    pdebug("script ncedula <notas/asistencias>")
    sys.exit(1)

# Leer e ignorar AGI pre info
leerIntro()

# importar la libreria e inicializarla
sys.path.append(CCLIBPATH)
import cclib # la libreria de acceso a recursos de la BD
cclib.inicializar(CONFIGFILE)


# validar los datos, revisar si existe un estudiante con esta cedula
# buscar los datos del estudiante con cedula CEDULA
try:
        de=cclib.buscarDatosEstudiante(CEDULA)
        tocar("callcenter-elestudiante")
        recibir()
        tocarNumero(CEDULA)
        recibir()
        tocar("callcenter-tiene%s" % CONSULTA)
        recibir()

        if CONSULTA=="notas":
            resultados = de.getNotas()
        else:
            resultados = de.getAsistencias()
```

```
        for k in resultados:
            pdebug("callcenter-materias-%s" % k.replace(" ","_"))
            tocar("callcenter-materias-%s" % k.replace(" ","_"))
            recibir()
            pdebug("nota es %.2f" % resultados[k])
            if type(resultados[k]) is float:
    tocarNumero("%.2f"%resultados[k])
    else:
         tocarNumero("%d"%resultados[k])
            recibir()
            time.sleep(1)


except cclib.CCLibError:
        pdebug("ERROR!: no se encontraron datos")
        tocar("callcenter-noexiste-estudiante")
        sys.exit()

tocar("callcenter-nomasdatos")
recibir()
sys.exit()
```

A.3.3. cclib.py

```
 #!/usr/bin/python
# -*- coding: utf-8 -*-
#
# $URL: $Url$
# $Id: $Id$
# $Autor: $Author$
# $LastChangedDate: $LastChangedDate$
"""
Programa para generar listas de fonetos de los resultados de una
consulta a una BD de estudiantes
"""
import sys
import ConfigParser
import string

# Definicion de excepciones
class CCLibError(Exception): pass

# Definicion de constantes base
#config=None
global config
global db
global debug
debug = 0

# Definicion de clases
class BuscadorEstudiantes:
    "un objeto buscador de estudiantes"
    def __init__(self):
        global db
        self.db=db
```

```
    def buscar(self,cedula):
        "buscar en la base de datos por el estudiante en concreto"

        r=self.db.exe("SELECT * FROM estudiante WHERE cedula="+cedula)

        if len(r) is not 0:
            # crear un datosEstudiante
            de = DatosEstudiante(r[0][0],r[0][1])
            # ponerle las notas de este estudiante y sus asistencias
            r=self.db.exe("SELECT * FROM datos_estudiante WHERE cedula="+cedula)
            #print ("DEBUG SELECT:",r)

            if len(r) is not 0:
                for d in r:
                    de.addMateriaNota(d[2],d[3])
                    de.addMateriaAsistencia(d[2],d[4])
            return de

        return None

class DatosEstudiante:
    "datos basicos de un estudiante - estructura necesaria"
    #c: cedula, n:nombre
    def __init__(self,c,n):
        "Datos de estudiante, parametros: cedula,nombre"
        self.cedula=c
        self.nombre=n
        self.materia_y_notas={}
        self.materia_y_asistencias={}

    def addMateriaNota(self,materia,nota):
        self.materia_y_notas[materia]=nota;

    def addMateriaAsistencia(self,materia,asistencia):
        self.materia_y_asistencias[materia]=asistencia

    def getNota(self,materia):
        return self.materia_y_notas[materia]

    def getAsistencia(self,materia):
        return self.materia_y_asistencias[materia]

    def mostrar(self,format=1):
        "presenta este estudiante formateado"
        if format is 1:
            print "NOMBRE:",self.nombre
            print "CEDULA:",self.cedula
            print "NOTAS:",self.materia_y_notas
            print "ASIST:",self.materia_y_asistencias
        else:
            print self.nombre,":",self.cedula
            print "ASISTENCIAS"
            for k in self.materia_y_asistencias.keys():
                print "'%s',%d" % (k,self.materia_y_asistencias[k])
            print "NOTAS"
```

```
            for k in self.materia_y_notas.keys():
                print "'%s',%d" % (k,self.materia_y_notas[k])

    def getAsistencias(self):
        "obtener una lista de asistencias por materia"
        return self.materia_y_asistencias

    def getNotas(self):
        "obtener una lista de asistencias por materia"
        return self.materia_y_notas

class MyConfig:
    "Configurador y parser de configs"
    _ConfigDefault = {
        "dbdriver" : "sqlite",
        "dbname" : "/var/lib/estudiantes",
        "dominio"      : "pupilabox.net.ec"
    }

    dicc = {}

    def __getattr__(self,id):
        "devolver del diccionario como si fueran atributos "
        try:
            return self.dicc[id]
        except KeyError:
            raise AttributeError, id

    def __init__(self,configFile="cclib.ini"):
        "cargarse la ini de entrada"
        self.configFile = configFile
        cp = ConfigParser.SafeConfigParser()
        cp.read(self.configFile)

        # guardar en un diccionario
        for sec in cp.sections( ):
            name = string.lower(sec)
            for opt in cp.options(sec):
                self.dicc[opt] = string.strip(cp.get(sec,opt))

    def __len__(self):
        return len(self.dicc)

# no hay class ccLib porque en python el script base hace de
# contenedor tipo clase jeje
def usage():
    "Imprime ayuda de este programa"
    print sys.argv[0],"PARAMETROS"
    print "PARAMETROS:  -d activa debug "
    print "             -c <cedula> : especifica la cedula a buscar"



def buscarDatosEstudiante(cedula):
    "Busca un estudiante con cedula y retorna notas o asistencia de
acuerdo a consulta"
```

```
    be = BuscadorEstudiantes()
    de = be.buscar(cedula)
    if de is not None:
        # aqui tenemos que retornar una lista con los datos puros
        # del estudiante, se me ocurre una lista q comience con
        # nombre_estudiante
        # materia ... nota ... asistencia
        # materia ... nota ... asistencia
        # update: retornamos de para que otro programa lo manipule
        # de.mostrar(0)
        #return "texto"
        return de
    else:
        raise CCLibError

def crearFoneto(texto):
    pass

def dprint(texto):
    "imprime si estamos en modo debug"
    global debug
    if debug:
        print texto


def inicializar(configFile="cclib.ini"):
    global debug
    global db
    debug = False
    config = MyConfig(configFile)
    db=None


    if config.dbdriver == "sqlite":
        dprint ("Importando Driver sqlite con dbname "+config.dbname)
        from lib.dbsqlite import DB
        from lib.dbsqlite import DBException
        db = DB(config.dbname)
    else:
        print "no existe un driver a la bd especificado"
        sys.exit(1)


if __name__ == '__main__':
    # buscar los parámetros por defecto
    cedula=None
    try:
        for o in range(0,len(sys.argv)):
            if sys.argv[o] == "-d":
                debug=True
                dprint ("DEBUG ON")
            if sys.argv[o] == "-c":
                cedula = sys.argv[o+1]
                dprint ("CEDULA ES:"+cedula)
        if cedula is None:
```

```
            print "No se ha especificado una cédula"
            raise IndexError
    except IndexError:
        usage()
        sys.exit()

    # inicializar el ambiente por defecto
    inicializar()
    # cedula="1109382005"
    dprint("Buscando un estudiante con cedula "+cedula)

    try:
        de=buscarDatosEstudiante(cedula)
        de.mostrar()

    except CCLibError:
        print "Error buscando estudiante... no existe"
        sys.exit()
```

A.3.4. dbsqlite.py

```
 # -*- coding: utf-8 -*-
# programa: layer de BD para SQLITE
# $URL: $Url$
# $Id: dbsqlite.py $


# Conectarse al sqlite
import sqlite
class DBException(Exception): pass


class DB:
        dbname="/var/tmp/ringspeaker.db"
        resultado=[]
        def __init__(self,dbname=None):
              if dbname is None: raise(DBException)
                  self.dbname=dbname
                  self.conn = sqlite.connect(dbname)
                  self.cursor = self.conn.cursor()

        def exe(self,s):
                self.cursor.execute(s)
                self.resultado = self.cursor.fetchall()
                return self.resultado

        def run(self,s):
                out = self.cursor.execute(s)

                self.conn.commit()
                #print "DEBUG DB.run:",out

        def cerrar(self):
                self.cursor.commit()
```

A.3.5. alimentador.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#
# $LastChangedDate: $LastChangedDate$
"""
Programa alimentador grafico de la base de datos del callcenter.
"""

import pygtk
pygtk.require('2.0')
import gtk

from sys import argv
from sys import path
from gazpacho.loader.loader import ObjectBuilder

path.append("/usr/local/cclib")
import cclib


class estudiantes(object):
    "separador de logica, ver algÃºn patron (facade es)"
    frm = None
    def __init__(self,wt):
        self.frm = wt.get_widget("estudiantes")
        self.wt=wt
        # crear el formulario de notas
        self.notas = notas(self.wt)
        self.nestudiante = nestudiante(self.wt,self)
        wt.signal_autoconnect(self)
        self.obtener_datos();

    #---------- utilidades del GUI -------#
    def obtener_datos(self):
        treeEstudiantes=self.wt.get_widget("treeEstudiantes")
        temodel = treeEstudiantes.get_model()
        temodel.clear()
        #cclib.inicializar()
        lestudiantes=cclib.listarDatosEstudiantes()
        #temodel.set_sort_column_id(2,gtk.SORT_ASCENDING)
        count=0
        for l in lestudiantes:
            count+=1
            temodel.append([count,l[0],l[1]])

    #---------- eventos del GUI ----------#
    def show(self):
        "mostrar el widget"
        self.frm.show()

    def on_botonAgregarEstudiante_clicked(self,button):
        self.nestudiante.show()
```

```
    def on_listaEstudiantes_dobleclick(self,tree,data,col):
        "ojo con esta forma esoterica de obtener datos del tree"
        model = tree.get_model()
        iter = model.get_iter(data)
        cedula=model.get_value(iter,1)
        nombre=model.get_value(iter,2)

        self.notas.show(cedula,nombre)

    def on_botonEstudiantesCerrar_clicked(self,button):
        gtk.main_quit()

    def on_estudiantes_destroy(self,window):
        gtk.main_quit()


class notas(object):
    "controlador de la ventana de notas de un estudiante"
    frm=None
    cedula=None
    nombre=None

    def __init__(self,wt):
        self.frm = wt.get_widget("notas")
        self.wt=wt
        self.wt.signal_autoconnect(self)

        materias=cclib.listarMateriasDisponibles()

        comboMaterias = wt.get_widget("comboMaterias")

        #comboMaterias = gtk.combo_box_new_text()
        ltor = gtk.ListStore(str,str)
        #ltor=comboMaterias.get_model()

        for l in materias:
            ltor.append([l[1],l[0]])

        comboMaterias.set_model(ltor)


    def show(self,cedula,nombre):
        self.cedula = cedula
        self.nombre = nombre
        labelNombreEstudiante = self.wt.get_widget("labelNombreEstudiante")
        labelCedulaEstudiante = self.wt.get_widget("labelCedulaEstudiante")
        labelNombreEstudiante.set_text(nombre)
        labelCedulaEstudiante.set_text(cedula)
        self.obtener_datos()
        self.frm.show()

    #-------- metodos de control ------------#
    def obtener_datos(self):
        "obtener los datos del gallo este"
        self.de = cclib.buscarDatosEstudiante(self.cedula)
```

```
        # tabularlos! uuuf
        treeNotasAsistencias=self.wt.get_widget("treeNotasAsistencias")
        tmodel = treeNotasAsistencias.get_model()
        tmodel.clear()
        lista_notas = self.de.getNotas()
        lista_asistencias = self.de.getAsistencias()
        lista_materias = self.de.getNotas().keys()

        for materia in lista_materias:
            tmodel.append([materia,lista_notas[materia],
                           lista_asistencias[materia]])


    #-------- metodos de presentacion -------#
    def on_botonAgregarNota_clicked(self,boton):
        "agregar la nota al de"
        # recuperar otra vez el combo
        comboMaterias = self.wt.get_widget("comboMaterias")
        ltor = comboMaterias.get_model()
        active =comboMaterias.get_active()
        nota = self.wt.get_widget("txtNota").get_text()
        asistencia = self.wt.get_widget("txtAsistencia").get_text()
        if active==None:
            print "no se selecciono nada.. saliendo"
            return

        # agregar el valor en de y en el tree
        self.de.addMateriaNota(ltor[active][0],nota)
        self.de.addMateriaAsistencia(ltor[active][0],asistencia)
        self.de.grabar()
        self.obtener_datos()
        #self.de.mostrar()

    def on_botonEditarNota_clicked(self,boton):
        print "No hay edicion porque se edita cuando se pone otra mat."

    def on_botonBorrarNota_clicked(self,boton):
        "borrarlo de todo definitivamente?? o esperar a cerrar?"
        # ver en donde esta el puntero en el tree
        treeNotasAsistencias=self.wt.get_widget("treeNotasAsistencias")
        tselect = treeNotasAsistencias.get_selection()
        store,iter = tselect.get_selected()
        if iter is not None:
tmodel = treeNotasAsistencias.get_model()
materia_a_borrar = tmodel.get_value(iter,0)
self.de.borrarMateriaNotaAsistencia(materia_a_borrar)
self.de.grabar()
self.obtener_datos()

    def on_botonNotasCerrar_clicked(self,boton):
        print "presionado ",boton
        self.frm.hide()

    def on_seleccionarRowNotas(self,data):
        print "seleccionado",data
```

```
class nestudiante(object):
    frm=None
    de=None

    def __init__(self,wt,parent):
        self.frm = wt.get_widget("nestudiante")
        self.wt=wt
        self.txtNuevoNombre=wt.get_widget("txtNuevoNombre")
        self.txtNuevoCedula=wt.get_widget("txtNuevoCedula")
        self.txtNuevoDireccion=wt.get_widget("txtNuevoDireccion")
        self.txtNuevoTelefono=wt.get_widget("txtNuevoTelefono")
        self.parent=parent #backref al formulario
        self.wt.signal_autoconnect(self)

    def limpiar_datos(self):
"limpiar todos los campos del formulario"
self.txtNuevoNombre.set_text("")
self.txtNuevoCedula.set_text("")
self.txtNuevoDireccion.set_text("desconocido")
self.txtNuevoTelefono.set_text("099665231")

    #--- metodos de presentacion -----#
    def show(self):
        "mostrar el widget"
        self.limpiar_datos()
        self.frm.show()

    def on_botonAECerrar_clicked(self,boton):
        self.frm.hide()

    def on_botonAEGrabar_clicked(self,boton):
"grabar al tipod"
nombre = self.txtNuevoNombre.get_text()
cedula = self.txtNuevoCedula.get_text()
direccion = self.txtNuevoDireccion.get_text()
telefono = self.txtNuevoTelefono.get_text()
de = cclib.DatosEstudiante(cedula,nombre)
de.grabar()
self.parent.obtener_datos()

def main():
    "este es el mainpoint"
    #wt.signal_autoconnect(globals())
    # crear primero alguien que vea en la bd
    cclib.inicializar()
    wt = ObjectBuilder(cclib.config.gui)
    # crear la ventana de estudiantes
    e = estudiantes(wt)
    e.show()
    # entrar al loop principal
    gtk.main()

if __name__ == '__main__':
    main()
```

A.4. Consulta de opinión del prototipo implementado en el AEIRNNR

A.5. Consulta de opinión del prototipo implementado en el AEIRNNR

A.6. Manual del Programador