UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

Título: "Interfaz de software para la conexión de aplicaciones externas al módulo del *Call Center* del *Elastix* (**ICAEC**)"

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores: José Carlos Pérez Zamora

Lester González López

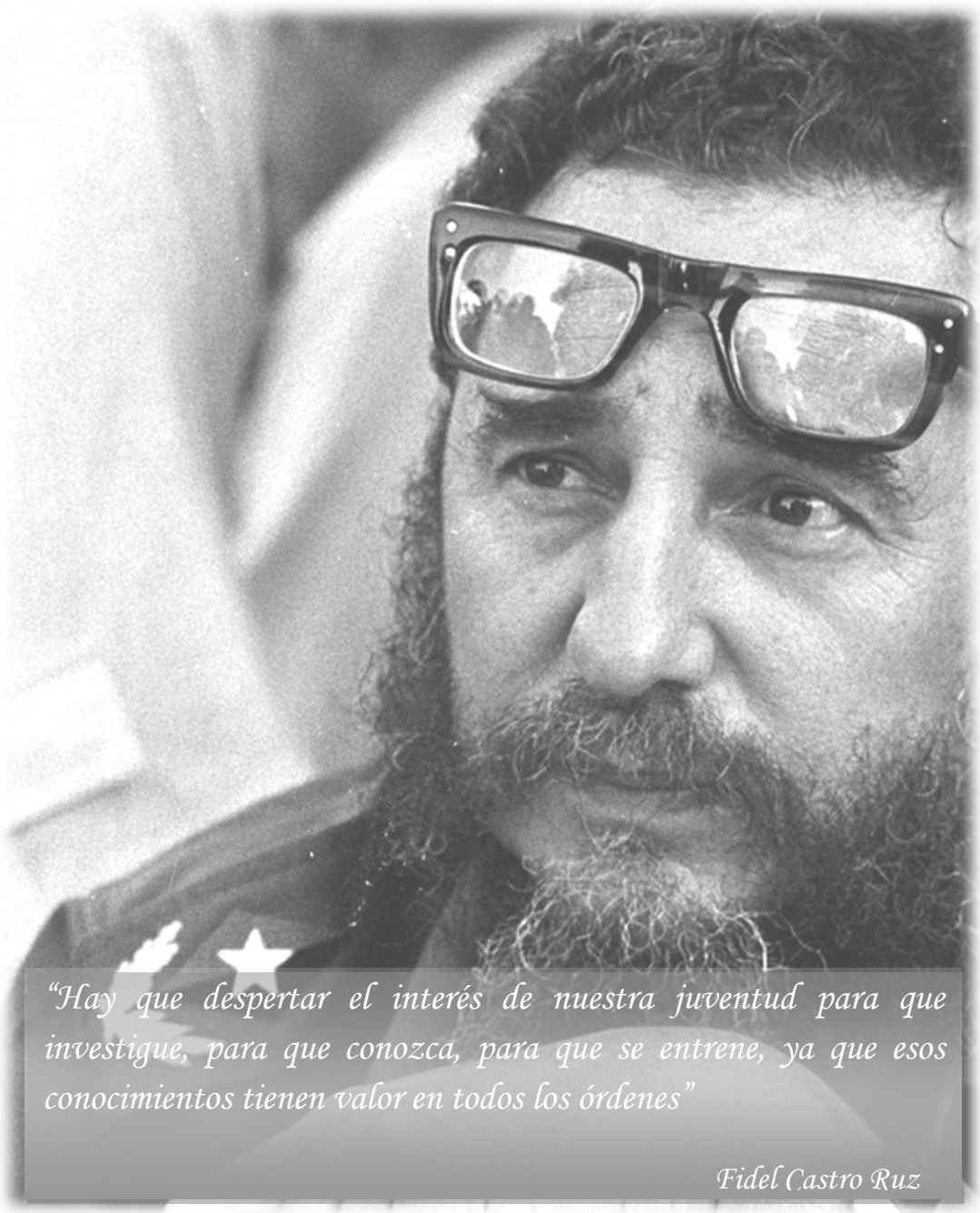
Tutores: Ing. Rainer Segura Peña

Ing. Lianet Salazar Labrada

Ing. Gerardo Rodríguez Fernández

"La Habana, Junio de 2013"

"Año 55 de la Revolución"



"Hay que despertar el interés de nuestra juventud para que investigue, para que conozca, para que se entrene, ya que esos conocimientos tienen valor en todos los órdenes"

Fidel Castro Ruz

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos al centro de Telemática de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____ _____

José Carlos Pérez Zamora

Firma Autor

Lester González López

Firma Autor

_____ _____

Ing. Rainer Segura Peña

Firma Tutor

Ing. Gerardo Rodríguez Fernández

Firma Tutor

_____

Ing. Lianet Salazar Labrada

Firma Tutor

DEDICATORIA

“A mis padres y hermanos, a la Revolución y especialmente a Fidel”

Lester

“A mis padres, hermanos y a la Revolución”

José Carlos

AGRADECIMIENTOS

“A mis familiares, en especial a mis padres, hermanos, abuelos, novia y mis tíos Nidia y Vito, que sin ellos no hubiese llegado a ser lo que soy ahora”.

“A mis tutores, que siempre estuvieron ahí cuando tenía dudas, cuando el camino se tornaba difícil, por ser perfeccionistas con cada detalle y hacerme dar lo mejor de mí en cada momento”.

“A mi compañero de tesis, que marchando unidos pudimos llegar a este resultado”.

“A mis amigos del Soho: José Carlos, Roexcy y Willian, quienes demostraron ser buenos amigos y juntos tratamos de solucionar los problemas del mundo”.

“A mis compañeros de cada brigada: 9101, 9201, 6301, 6401, 2402 y 2502”.

“A mis amigos y compañeros de trabajo de la FEU de la antigua facultad 9, de la facultad 6, de la facultad 2 y del Consejo de la FEU UCI”.

“A Juan Daniel, la profe Ailec y mi tío Vito, que me ofrecieron su apoyo incondicional para el desarrollo del trabajo de diploma”.

“A todo aquel que contribuyó con mi formación, en especial a los profesores de esta casa de altos estudios”.

“A la FEU, que me quitó tiempo pero me enseñó a aprender y abrirme camino en la vida, por lo que ahora ganaré mucho más”.

“A Fidel Castro Ruz y a la Revolución, que me brindó la posibilidad de estudiar en esta casa de altos estudios y ser hoy Ingeniero en Ciencias Informáticas”.

Lester

“A mis padres Virgen y Jorge, a mis hermanos Sandra y Jorgito, a mis tías María del Pilar y Leonor, que me apoyaron y mantuvieron durante estos cinco años de estudios y también se sacrificaron por mí”.

“Al profesor Rainier Segura, que sin su guía no hubiese llegado a este momento”.

“A mis tutores, que brindaron una ayuda incondicional”.

“A mi compañero de tesis y amigo Lester, que gracias a su empeño e inteligencia alcanzamos este resultado”.

“A mis amigos que siempre han sido un apoyo importante y necesario, en especial a Willian, Lester, Roexcy y Enier, que estuvieron presente en los buenos y malos momentos, y que hoy son como hermanos”.

“A la UJC y a la FEU que fomentaron en mí valores de compromiso y lealtad, me instruyeron y educaron en el trabajo con las personas”.

“A la Revolución que me permitió venir del lejano oriente a formarme como Ingeniero en Ciencias Informáticas”.

José Carlos

RESUMEN

En estos tiempos, la telefonía IP[1] y las comunicaciones unificadas se han convertido en un eslabón principal para el desarrollo tecnológico de la sociedad. En este contexto, *Elastix* es una opción de plataforma integral para este tipo de implementaciones, que tiene su propio conjunto de utilidades y permite la creación de módulos de terceros para hacer de este un paquete de software muy utilizado en la telefonía de código abierto. Este sistema nos permite contar con una central telefónica de inmensas prestaciones e integra tecnologías de fax, telefonía, mensajería instantánea, correo electrónico y de colaboración. Uno de los módulos que integra en su aplicación web es el de *Call Center*, encargado de gestionar las llamadas de una típica central telefónica. Actualmente en el Centro Telemática (TLM) de la Universidad de las Ciencias Informáticas (UCI) se trabaja con una solución basada en *Elastix*, al que se le instala el módulo de *Call Center*. Varias son las aplicaciones que en el centro se desarrollan y necesitan utilizar las funcionalidades del módulo de *Call Center* del *Elastix*. Estos sistemas requieren implementar una conexión que les permita hacer uso de una o varias funcionalidades, teniendo que conocer los protocolos necesarios para establecer dicha conexión, los permisos, nombres de usuarios, contraseñas y otros parámetros del servidor de telefonía que deberían ser transparentes para aplicaciones externas, además de que los usuarios tienen que tener dominio sobre el protocolo existente para establecer dicha conexión, para entonces poder implementarlo en el software que desarrollan.

PALABRAS CLAVE

Elastix, módulo de *Call Center,* servicio web

[1] Protocolo de Internet

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las tecnologías han llegado a ocupar un lugar importante en la vida de las personas, haciéndolas cada vez más dependientes de las mismas y logrando que sean el centro de interés para el desarrollo en diferentes áreas del conocimiento y de la ciencia. Una de las ramas que más ha revolucionado al mundo es la de las Telecomunicaciones, que ha traído grandes transformaciones técnicas y sociales, llegando a influir en la manera de vivir y actuar de la población.

Se han logrado vencer las barreras y limitaciones que existían años atrás respecto a las comunicaciones, logrando establecerlas de diferentes formas y entre dos o más objetivos en cualquier lugar que se encuentren. Uno de los servicios que goza de aceptación es la telefonía, que desde su surgimiento ha tenido un gran impacto en la sociedad debido a que logra acortar las distancias para la comunicación, sin tener la necesidad de trasladarse para transmitir o recibir una información. El auge de esta tecnología trajo consigo que el mundo comenzara a demandar una serie de servicios asociados a estos dispositivos, tales como fax, mensajería instantánea y correo electrónico, observándose la necesidad de crear un software que integre todas estas prestaciones en un mismo sistema, dentro de los cuales se encuentra el *Elastix*.

Elastix, es una distribución de software libre de servidor de comunicaciones unificadas (1). Esta tecnología va a la vanguardia en cuanto a la integración de varios servicios, tales como correo de voz, correo electrónico, fax, soporte para VoIP[2], mensajería instantánea y video-llamadas en su versión 2.0. Implementa una parte de su funcionalidad sobre el programa *Asterisk*, que es un software de centralita[3] con capacidad para voz sobre IP, el cual es instalado y a través de una aplicación web brinda la posibilidad de administrar una planta telefónica y otras aplicaciones de una forma intuitiva, agradable y profesional. Esta planta telefónica desempeña un rol importante, indistintamente de sus altos precios y el

[2] Término usado en telefonía IP para definir los servicios que se usan al transmitir voz usando el protocolo IP.

[3] Aparato que conecta una o varias líneas telefónicas con diversos teléfonos instalados en los locales de una misma entidad.

soporte que necesitan. Por ello, es recomendable utilizar las que operan sobre software libre, por la gran demanda mundial que tienen y la cantidad de funcionalidades que brindan. *Elastix* también posee varios módulos que pueden ser cargados desde la misma interfaz web y uno de ellos es el de *Call Center* (Centro de Llamadas). Este módulo constituye la unidad funcional de la empresa diseñada para manejar grandes volúmenes de llamadas telefónicas entrantes y salientes, entre agentes y clientes.

Cuba, dentro de su amplio programa de informatización de la sociedad cubana, ha priorizado optar por el uso de las telecomunicaciones y demás tecnologías basadas en software libre. Una de las instituciones que ha contribuido con el país en cumplir con esta política ha sido la Universidad de las Ciencias Informáticas (UCI), como centro de excelencia en la formación de profesionales competentes en la rama de la informática y en el desarrollo de software. Dicha institución cuenta con una estructura especializada para la actividad productiva, dentro de la cual se encuentra el centro de telemática (TLM), encargado de trabajar en el desarrollo de aplicaciones relacionadas con las telecomunicaciones. Esta entidad también ofrece un servicio de montaje de *Call Center*, cuya aplicación de alternativa libre es *Elastix*, diseñado primordialmente para entornos corporativos, basado en soluciones de telefonía y con muchas funcionalidades que pueden ser gestionadas a través de múltiples aplicaciones.

Varias son las aplicaciones informáticas que se desarrollan, las cuales requieren implementar una conexión al módulo de *Call Center* del *Elastix* para hacer uso de las diferentes funcionalidades que este tiene. Dicha conexión es realizada a través del *Elastix Call Center Protocol* (ECCP), teniendo que ser implementado este protocolo en cada aplicación cliente, lo cual constituye mayor trabajo y redundancia de código para el equipo de desarrollo. Además, de esta manera cada aplicación abre una conexión al *Call Center* cada vez que hace una petición, lo cual provoca que el servidor del *Elastix* pierda rendimiento, pues tiene que encargarse de gestionar las sesiones para las aplicaciones clientes. También resulta necesario tener un control de los usuarios que utilizan dichas funcionalidades para el desarrollo de sus aplicaciones, además de que lo hagan a través de un usuario y contraseña que le sea asignado, pues actualmente para establecer la conexión tienen que conocer los protocolos necesarios, los permisos, nombres de usuarios, contraseñas y otros parámetros del servidor de telefonía que deben ser transparentes para aplicaciones externas.

En base a lo expresado, la presente investigación se propone encontrar una solución a la situación planteada, por lo que se identifica como **problema a resolver**: ¿Cómo evitar la conexión directa de aplicaciones externas al módulo de *Call Center* del *Elastix*?

Teniendo en cuenta el problema planteado se determinó que el **objeto de estudio** de esta investigación sea el proceso de conexión de aplicaciones externas al módulo de *Call Center* del *Elastix*.

Con el fin de dar respuesta al problema se define como **objetivo general**: desarrollar una interfaz de software que evite la conexión directa de aplicaciones externas al módulo de *Call Center* del *Elastix.* El **campo de acción** de la investigación está enmarcado en el proceso de conexión de aplicaciones externas al módulo de *Call Center* del *Elastix* mediante una interfaz de software*.*

El objetivo general se desglosa en los siguientes **objetivos específicos**:

- Realizar un estudio del objeto de la investigación, así como de las herramientas, tecnologías, y metodología de desarrollo para la elaboración del software.
- Definir las funcionalidades que debe tener la interfaz de software.
- Desarrollar las funcionalidades definidas.
- Realizar pruebas para verificar el correcto funcionamiento de la interfaz desarrollada.

La **idea a defender** es desarrollar una interfaz de software para la conexión de aplicaciones externas al módulo de *Call Center* del *Elastix*, para que estas hagan uso de sus funcionalidades.

Para dar cumplimiento a los objetivos se realizaron esencialmente las siguientes **tareas**:

1. Caracterizar las diferentes vías de conexión existentes al módulo de *Call Center* del *Elastix*.
2. Definir las funcionalidades que necesita el cliente y los demás requisitos que debe cumplir la interfaz de software.
3. Seleccionar las herramientas, metodologías y tecnologías de trabajo a utilizar para el diseño e implementación del software.
4. Identificar el diseño de clases para facilitar la implementación.

5. Verificar el correcto funcionamiento de los requisitos a través del diseño de los casos de pruebas.

Para apoyar el desarrollo de la investigación se emplean los siguientes métodos científicos:

- **Métodos teóricos:**

Analítico – Sintético: este método ayuda a analizar y comprender la base teórica, así como toda la documentación referente al objeto de estudio, facilitando la obtención de los elementos más importantes relacionados con el mismo. Resulta determinante para definir el camino a seguir en cada paso de la investigación a partir de la exploración de la realidad.

Inductivo – Deductivo: permite que a partir del estudio de los hechos aislados se pueda arribar a proposiciones generales y que a partir de lo general se pueda inferir casos particulares, teniendo como punto de partida un razonamiento lógico. Se pone de manifiesto cuando se selecciona la tecnología a utilizar en el flujo de trabajo de implementación del producto, después de un estudio de las existentes.

Modelación: se evidencia con la creación de reproducciones simplificadas del entorno, como son los modelos y diagramas representados, permite la reproducción de la realidad y facilita estudiar nuevas relaciones y cualidades del objeto de estudio.

- **Métodos empíricos:**

Entrevista: consiste en establecer una conversación planificada con una o varias personas para la obtención de información necesaria para la investigación. Su uso constituye una vía para el conocimiento cualitativo de los fenómenos o sobre características personales del entrevistado que pueden influir en determinados aspectos de la conducta humana, por lo que es recomendable una buena comunicación. Se realizaron varias entrevistas para definir las principales funcionalidades del *Call Center* que se usarán a través de la interfaz creada, además de otras que se realizaron con personas especializadas en el trabajo con *Asterisk* y *Elastix*.

El trabajo de diploma se divide en 3 capítulos, los cuales están estructurados de la siguiente forma:

Capítulo 1. "**Fundamentación Teórica**": en este capítulo se describen los principales conceptos a tratar, tales como *Elastix*, módulo de *Call Center*, interfaz de software, aplicaciones externas y *Asterisk* como software que proporciona las funcionalidades de una planta telefónica. Se define la metodología de software, herramientas y tecnologías a utilizar para el desarrollo de la aplicación.

Capítulo 2. "**Características del Sistema, Exploración y Planificación**": en este capítulo se caracteriza el sistema, se confeccionan las Historias de Usuarios (HU), que son de interés para la primera entrega del producto, así como una propuesta del prototipo no funcional de la aplicación. A partir de la metodología seleccionada para el desarrollo de la aplicación, también se describen las fases por la que está compuesta la misma.

Capítulo 3. "**Diseño**, **Implementación y Prueba**": en este capítulo se define la arquitectura del sistema y los patrones que se utilizan, se realiza el modelo físico de la base de datos y el de despliegue. Se definen las tareas de la ingeniería asociadas a las diferentes HU. Se implementan cada una de las funcionalidades hasta lograr el producto completo y se realizan las respectivas pruebas unitarias y de aceptación para validar el correcto funcionamiento de la interfaz.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se profundizan los principales conceptos relacionados con el desarrollo de la interfaz de software que permite a aplicaciones externas hacer uso de las funcionalidades del módulo de *Call Center* del *Elastix,* para entender mejor la investigación. Se realiza un estudio sobre cómo se efectúa este acceso en la actualidad, tanto en el escenario internacional como el realizado por el propio centro de Telemática y se exponen las principales características de las tecnologías, herramientas y metodología empleada.

1.2 Términos asociados al objeto de estudio

Para entender el problema a resolver es necesario explicar algunos términos relacionados con el objeto y campo de acción de la investigación. Estos términos se muestran a continuación:

1.2.1 Elastix

Elastix como distribución libre de Servidor de Comunicaciones Unificadas integra varias herramientas disponibles para centrales telefónicas e implementa gran parte de sus funcionalidades sobre el software *Asterisk*, a través de una interfaz gráfica muy sencilla y fácil de usar. Este sistema fue creado y actualmente mantenido por la compañía ecuatoriana *Palosanto Solutions* y sus características son variadas, permitiendo integrar en un solo paquete la mayoría de las funcionalidades de VoIP, PBX[4], Fax, mensajería instantánea y correo electrónico, así como añadir su propio conjunto de utilidades y creación de módulos.

Algunas de las características con las que cuenta *Elastix* son (2):

- Soporte para video. Se puede usar video-llamadas con *Elastix* en su versión 2.0.

[4] *Private Branch Exchange*: Central Secundaria Privada

- Soporte para virtualización. Es posible correr múltiples máquinas virtuales de *Elastix* sobre la misma caja.
- Interfaz web para el usuario.
- "Fax a correo" para faxes entrantes. También se puede enviar algún documento digital a un número de fax a través de una impresora virtual.
- Interfaz para tarifas.
- Configuración gráfica de parámetros de red.
- Opciones para reiniciar/apagar remotamente.
- Reportes de llamadas entrantes/salientes y uso de canales.
- Módulo de correo de voz integrado.
- Interfaz web para correo de voz.
- Interfaz de ayuda embebido.
- Servidor de mensajería instantánea (*Openfire*) integrado.

1.2.2 Software *Asterisk*

Software desarrollado por la compañía *Digium*, la cual está encargada del desenvolvimiento de código fuente y en hardware de telefonía de bajo costo que funciona con *Elastix*. Programa de software libre que convierte el ordenador en una central telefónica multifuncional, haciendo operativas las funcionalidades de PBX del *Elastix*. Integra requisitos de telefonía clásica con nuevas capacidades derivadas de su potente arquitectura, el cual puede conectar un elevado número de teléfonos para hacer llamadas entre sí. El sistema *Asterisk* ofrece todo lo que una empresa puede esperar para una completa solución de *Call Center*, pues en cuanto a funcionalidad dispone de todas las opciones que tienen las grandes centralitas propietarias (Cisco, Avaya, Alcatel y *Siemens*), desde las más básicas (desvíos, capturas, transferencias y multi-conferencias) hasta las más avanzadas (buzones de voz, IVR[5], ACD[6])

[5] *Interactive Voice Response:* Operadora automática sin limitaciones de opciones y funcionalidad.

1.2.3 Módulo de *Call Center*

Elastix fue la primera versión en incluir un módulo de *Call Center* de código abierto con marcador predictivo. Este módulo puede ser cargado desde la misma interfaz web del *Elastix* a través de un cargador de módulos que posee. Con el mismo se puede implementar un proyecto de centro de llamadas en un menor tiempo, permitiendo la gestión de agentes, colas, campañas entrantes y salientes, desarrollo de formularios, llamadas predictivas y gestión de tiempos de descanso. Cuenta además con una serie de estadísticas básicas y opciones de monitorización en tiempo real, tanto de los agentes como de las llamadas entrantes. La versión utilizada del módulo de *Call Center* es la 2.1.3, la cual es funcional para *Asterisk.*

El módulo de *Call Center* puede manejar campañas entrantes y salientes. Algunas de sus características son (3):

- Marcador predictivo de código abierto.
- Soporte para *Do-Not-Call List*.
- Soporte para campañas salientes y entrantes.
- Los formularios pueden ser asociados a una campaña y diseñados a través de un "Web *wizard*".
- Un "*Script*" puede ser asociado a una campaña.
- Consola de agente.
- Soporte para varios tipos de *breaks*[7].
- Reportación avanzada.
- Capacidad de conexión a bases de datos: *Oracle, MS-SQL, MySQL* y *PostgreSQL*.

---

[6] *Automatic Call Distribution:* Sistema de gestión de llamadas por colas y agentes.

[7] Se le denomina al motivo por el cual el agente no está funcional durante un tiempo determinado.

Campañas Salientes:

- Genera llamadas desde un listado de teléfonos.
- Las llamadas son asignadas a los agentes para intercomunicar con los clientes.
- Marcador predictivo genera las llamadas buscando la mayor eficiencia.
- El operador recibe información del usuario y la ingresa a la base de datos a través de formularios.
- La información puede ser exportada a hojas de cálculo.

Campañas Entrantes:

- *Call Center* recibe llamadas y se las asigna a los agentes a través de colas.
- Se puede alimentar una base de números telefónicos y clientes para conocer quien está generando la llamada y brindar mejor atención al cliente.

1.2.4 Aplicación

“*Un programa diseñado para asistir en la realización de una tarea específica, tal como un procesador de textos, contabilidad o gestión de inventario*”. (4)

1.2.5 Aplicaciones externas

Se emplea este término para hacer referencia al conjunto de aplicaciones de software que existen y se pueden crear, que requieren hacer una conexión al *Elastix* para hacer uso de sus funcionalidades. Las mismas son independientes a este sistema, es decir, no son propias del *Elastix*, por lo que se denominan como “externas”.

1.2.6 Interfaz de software

“*Medio físico, lógico común y necesario de dos sistemas para intercambiar comunicación*”. (5) “*También puede ser el punto en el que tiene lugar la conexión entre dos elementos de tal forma que ambos puedan trabajar en cooperación.”* (4) En el campo de la informática se distinguen distintos tipos de interfaces que

actúan a diversos niveles, desde las que permiten a personas comunicarse con los programas, hasta las de hardware, que conectan entre sí los dispositivos y componentes dentro de ordenadores.

A nivel de software se encuentran las interfaces de usuarios, que cuentan con el diseño gráfico, los comandos, mensajes y otros elementos que permiten a un usuario comunicarse con un programa. Existen tres tipos básicos de interfaces de usuario: la interfaz de línea de comandos, la interfaz controlada por menús y la interfaz gráfica de usuario. La interfaz que se desarrolla en el presente trabajo de diploma, facilita la comunicación entre aplicaciones, pues permite la conexión entre el módulo de *Call Center* del *Elastix y* otras aplicaciones. A nivel de hardware se entienden por interfaces, las tarjetas, los conectores y otros dispositivos con que se conectan los diversos componentes a la computadora para permitir el intercambio de información.

1.2.7 Servicios Web

El término "servicios web" designa una tecnología que permite que las aplicaciones se comuniquen en una forma que no depende de la plataforma ni del lenguaje de programación. Un servicio web es una interfaz de software que describe un conjunto de operaciones a las cuales se puede acceder por la red a través de mensajería XML estandarizada. Un grupo de servicios web que interactúa de esa forma define la aplicación de un servicio web específico en una arquitectura orientada a servicios (SOA). La industria de software finalmente se está dando cuenta de que integrar aplicaciones de software en varios sistemas operativos, lenguajes de programación y plataformas de hardware no es algo que puede ser resuelto por un entorno patentado y específico. Por tanto, en términos técnicos, los servicios web pueden manejar datos con mucha más facilidad y permiten una comunicación más libre entre los software. (6)

1.3 Objeto de estudio y análisis de soluciones existentes

El acceso de aplicaciones externas al motor del *Call Center* del *Elastix* a través de una interfaz de software cobra gran importancia en la actualidad. La mayoría de los proyectos de software que trabajan estas tecnologías realizando aplicaciones que utilizan funcionalidades del *Call Center,* lo hacen de forma directa, poniendo a conocimiento de todos los desarrolladores los protocolos, permisos, nombres de

usuarios, contraseñas y otros parámetros del servidor. A continuación se explica el mecanismo actual existente para acceder al *Asterisk* y a módulos del *Elastix* con el fin de comprender la necesidad de realizar dicha interfaz de software.

1.3.1 *Asterisk* AGI

AGI (*Asterisk Gateway Interface* por sus siglas en inglés) es una manera de interactuar con *Asterisk* desde un programa de línea de comandos. Este permite a *Asterisk* ejecutar un proceso o aplicación externa que puede estar escrito en cualquier lenguaje de programación con el fin de poder controlar canales telefónicos, la posibilidad de reproducir audio, obtener datos del cliente y ejecutar varios comandos. Al momento de la invocación de un programa AGI se le pasan ciertos parámetros y este a su vez responde con comandos AGI que son entendidos por *Asterisk*. Los AGI son llamados desde el plan de marcación de *Asterisk* y no desde una petición externa explícita, por lo que está pensado para comunicaciones locales y no desde equipos remotos (7).

1.3.2 *Asterisk* AMI

AMI (*Asterisk Manager Interface* por sus siglas en inglés) es una manera de comunicarse con *Asterisk* desde un programa cliente por medio de un *socket* TCP/IP con el fin de poder enviar acciones (comandos), leer eventos y recibir la respuesta de los mismos. Los clientes pueden ser implementados en cualquier lenguaje de programación que maneje *sockets* TCP/IP. Este tipo de interfaz es muy útil a la hora de desarrollar aplicaciones externas que utilicen *Asterisk* como IVRs (*Interactive Voice Response por sus siglas en inglés*), *paneles* de control y visualización, marcadores automáticos y aplicaciones que interactúan con bases de datos (7).

AMI puede ser ejecutado desde equipos remotos, es esto último lo que lo convierte en una herramienta poderosa. AMI permite desarrollar poderosas aplicaciones cliente que pueden ejecutarse en otros computadores, liberando con esto de carga al servidor *Asterisk*. Un ejemplo de esto es el módulo para *Call Center*, el cual está codificado en gran parte usando AMI. Debido a que a través de AMI equipos remotos están en la capacidad de controlar *Asterisk* se requiere que estos equipos se autentiquen. Estos usuarios,

claves y permisos se definen en un archivo llamado "*manager.conf*". Cada aplicación externa establece una conexión al AMI con cada uno de los servidores *Asterisk* tanto de conmutación como de agentes. La aplicación crea tantos hilos como conexiones a servidores haya. Con relación al AMI, también se han reportado problemas de bloqueos en los hilos, por la razón expuesta anteriormente. Los eventos del "*Manage*r" no están muy bien estructurados como protocolo y son difíciles de interpretar. La conexión al AMI permite al cliente esencialmente escribir cualquier dato en el *socket* permitiendo un acceso directo al módulo, lo que puede convertirse en una falla de seguridad para la aplicación.

1.3.3 ECCP (*Elastix Call Center Protocol*)

El ECCP es un protocolo basado en XML cuya finalidad es proveer un API[8] de comunicaciones a través de un puerto TCP para que aplicaciones externas puedan comunicarse con el motor de *Call Center* que provee *Elastix*. Esto facilita que terceros desarrollen sus propias consolas de agente u otro tipo de aplicaciones. Este soporta la comunicación de eventos asincrónicos, lo que elimina la necesidad del *polling*[9]*,* lo cual genera consultas constantes al servidor y un desperdicio innecesario de recursos del servidor del *Elastix* (3).

El protocolo ECCP está orientado a sesiones, teniendo la capacidad de soportar múltiples sesiones de comunicación, pudiendo atender de esta manera a varias aplicaciones externas a la vez. Esto abre las puertas a dos escenarios diferentes: primero, conexión directa al servidor y segundo, conexión mediante una aplicación intermediaria. La segunda es la opción más adecuada, pues al estar instalada en un equipo remoto se liberan recursos del servidor *Elastix*, no teniendo este que encargarse de gestionar las sesiones para las aplicaciones externas. En el caso del primer escenario cada vez que un software se conecta al servidor debe abrir una sesión, a diferencia de que mediante la otra vía es una sola la que se inicia.

---

[8] A*pplication Programming Interface* por sus siglas en inglés.

[9] Operación de consulta constante para crear una actividad sincrónica sin el uso de interrupciones.

1.3.4 Elastix Web Service

Este es un módulo del *Elastix* disponible desde que fue lanzada la versión 2.0.4, el cual permite obtener o modificar información importante de ciertos módulos del servidor *Elastix*, los cuales son: módulo *Calendar*, módulo *AddressBook*, módulo *Fax*, módulo *My_Extension* y módulo *VoiceMail.* Se trata de un protocolo basado en SOAP (*Simple Object Access Protocol* por sus siglas en inglés), que posibilita acceder a la información del servidor del *Elastix* a través del sistema de permisos del mismo. De esta forma pueden crearse perfiles de usuarios solo para consultas de *Web Services* y que tengan acceso a solo determinados módulos. En esta fase el *Elastix Web Service* está implementado solo en los módulos mencionados, no siendo así para el del *Call Center.*

1.3.5 Acceso al módulo de *Call Center* en el centro de Telemática

Para el desarrollo de aplicaciones que se realizan en el centro de Telemática de la UCI es necesario utilizar las facilidades y potencialidades que brinda el módulo de *Call Center* del *Elastix*, para lo cual los miembros de los equipos de proyectos tienen que acceder al servidor donde se encuentra. Para ello cada equipo de desarrollo tiene que implementar el acceso a través del protocolo ECCP, estableciendo varias sesiones desde diferentes equipos remotos en el servidor, lo cual provoca que el *Elastix* pierda en rendimiento. Además, cada equipo de desarrollo de la aplicación externa debe implementar el protocolo, lo cual provoca que el mismo código esté en diferentes aplicaciones y no en un único software de forma tal que pueda ser reutilizado por los demás. También, para que los desarrolladores puedan acceder al *Call Center* hay que ponerles a su disposición todas las claves para entrar al servidor, lo cual no puede garantizar que los usuarios tengan acceso a determinadas funcionalidades del *Call Center* y no a todas.

A partir del estudio realizado de las principales vías y software existentes para establecer la conexión con el módulo de *Call Center* se determinó que ninguna satisface las condiciones necesarias para dar solución a la situación problémica de la investigación. También en entrevistas realizadas con desarrolladores de la compañía *PaloSanto Solutions*, se confirmó el planteamiento anterior.

1.4 Interfaz de software para acceder al *Call Center*

El software resultante de esta investigación permite cambiar el escenario actual en el centro, facilitando el trabajo a los profesionales y estudiantes que laboran en el mismo. La aplicación otorga los permisos necesarios para que determinados usuarios utilicen solamente las funcionalidades del módulo de *Call Center* del *Elastix* que necesitan y no otras. Las mismas se ofrecen a través de servicios web que consumen las aplicaciones externas. De esta forma cada usuario tiene su clave para consumir los servicios web a los cuales tiene autorización, a través de una aplicación intermediaria, la cual es la única en establecer comunicación con el *Call Center* y en implementar el protocolo ECCP. La interfaz tiene en cuenta la restricción de las diferentes direcciones IP para el acceso al *Call Center*. En la propuesta de solución, explicada en el capítulo 2, se aborda detalladamente cómo se efectúa este proceso de conexión.

1.5 Metodologías de desarrollo

Las metodologías de desarrollo de software surgen por la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar las aplicaciones. Los requisitos para los diferentes programas que se realizan varían, dando lugar a que existan una gran variedad de metodologías para la creación del software. Existen dos grandes grupos (8):

- Las metodologías orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usan, las cuales son llamadas metodologías pesadas.
- Las metodologías orientadas a la interacción con el cliente y el desarrollo incremental del software, las cuales muestran versiones parcialmente funcionales al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando, las cuales son llamadas metodologías ligeras o ágiles.

1.5.1 Programación Extrema o XP (*EXTREME PROGRAMMING*)

La programación extrema utiliza un enfoque orientado a objetos como su paradigma de desarrollo preferido. Abarca un conjunto de reglas y prácticas que ocurren en el contexto de cuatro actividades en el marco de trabajo: planeación, diseño, codificación y pruebas. (9) Esta es la que se utilizó en el desarrollo de la interfaz de software.

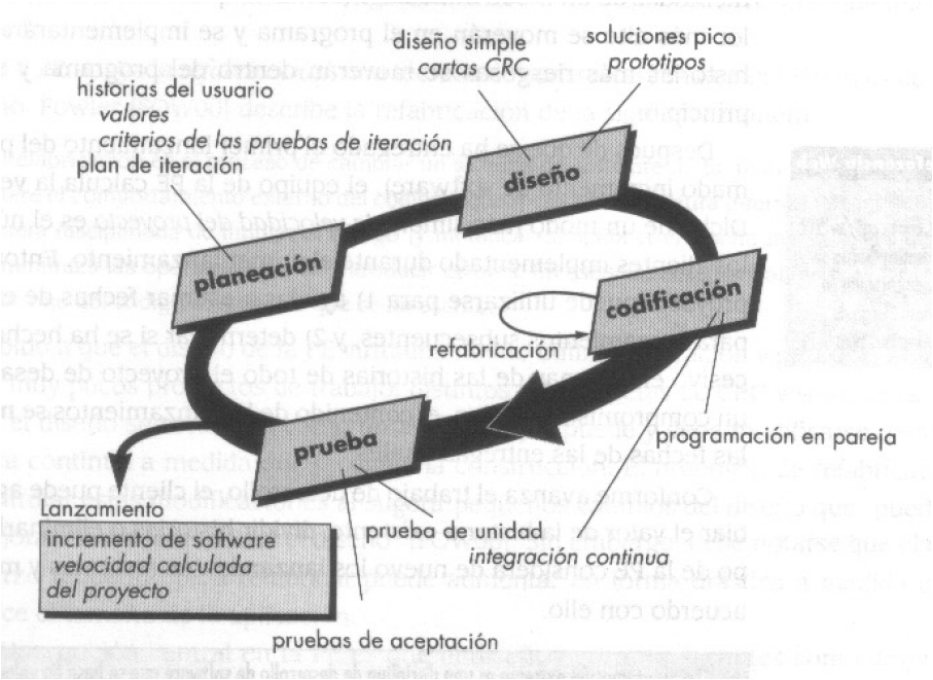


Figura 1: Etapas de la Metodología XP (9)

A continuación se exponen las principales facilidades que propone la metodología seleccionada (10):

- Empieza en pequeño y añade funcionalidades con retroalimentación continua.
- El manejo del cambio se convierte en parte sustantiva del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes de que sean necesarias.
- El cliente o el usuario se convierte en miembro del equipo.

También resulta importante esclarecer elementos distintivos de XP, los cuales se han tomado en cuenta a la hora de determinar el tipo de metodología a emplear en el presente trabajo. Los mismos son (10):

- La comunicación entre los usuarios y los desarrolladores.
- La simplicidad al desarrollar y codificar los módulos del sistema.
- La retroalimentación concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

1.6 Sistemas Gestores de Bases de Datos (SGBD)

Un sistema de bases de datos es un archivo compuesto de registros, donde cada uno contiene campos junto con un conjunto de operaciones para realizar búsquedas, ordenaciones, reordenaciones y otras funciones. (4) Una base de datos es un conjunto ordenado e interrelacionado de los datos de una organización cualquiera. El modelo de base de datos a utilizar es el relacional y su principio básico es representar tanto las entidades denominadas también tablas, como las asociaciones con la ayuda de relaciones. El SGBD que se escogió fue *PostgreeSQL*.

1.6.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, cuyo código fuente está disponible libremente. Utiliza un modelo cliente-servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema, pues un fallo en uno de los procesos no afecta el resto y el sistema continúa funcionando.

Algunas características del gestor de base de datos a utilizar son (11):

- Integridad referencial.
- Replicación asincrónica/sincrónica.
- Copias de seguridad en caliente *(Online/hotbackups).*
- Juegos de caracteres internacionales.

- Numerosos tipos de datos y posibilidad de definir nuevos tipos. Además de los tipos estándares en cualquier base de datos, tenemos disponibles, entre otros, tipos geométricos, de direcciones de red, de cadenas binarias, XML y matrices.
- Soporta el almacenamiento de objetos binarios grandes (gráficos, videos y sonido).
- APIs para programar en C/C++, Java, .Net, *Perl*, *Python*, *Ruby*, PHP, *Scheme*, Qt y muchos otros.
- Llaves primarias (*primary keys*) y foráneas (*foreign keys*).
- Columnas auto-incrementales.
- Consultas recursivas.
- *Joins*.
- Vistas (*views*).
- Disparadores (*triggers*) comunes, por columna y condicionales.
- Reglas (*Rules*).
- Herencia de tablas (*Inheritance*).

1.7 Herramientas de Desarrollo

1.7.1 Lenguajes de programación

1.7.1.1 Lenguaje de programación PHP

PHP (acrónimo de "PHP: *HypertextPreprocessor*") es un lenguaje de código abierto interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor. (12) PHP se creó específicamente para la generación de páginas web. Tareas comunes de programación en este campo como acceder a la información enviada en un formulario y conectarse a una base de datos, son a menudo más sencillas en PHP. A esto se añaden valores como el hecho de ser un proyecto de código abierto, gratuito y multiplataforma. Dicho lenguaje es fácil de usar por los principiantes, pero a la vez ofrece muchas características avanzadas a los programadores profesionales. El utilizado para el desarrollo de la interfaz de software es la versión 5.4, siendo empleado para el desarrollo del subsistema de gestión y servicios web.

1.7.1.2 Lenguaje de programación C++

El lenguaje C++ se escogió para la implementación del subsistema de servicios de comunicación. Entre sus principales características está el soporte para la programación orientada a objetos y el soporte de plantillas o programación genérica, además de ser un lenguaje de alto nivel que tiene gran potencial para poder trabajar tanto en bajo, como en alto nivel. Posee dos propiedades importantes que son difíciles de encontrar en otros lenguajes que son la posibilidad de redefinir los operadores, comúnmente conocido como la sobrecarga de operadores, y la identificación de tipos en tiempo de ejecución. También presenta una biblioteca estándar muy poderosa, usado en el ámbito educacional y profesional.

1.7.2 Lenguaje de programación del lado del cliente.

1.7.2.1 HTML

HTML (acrónimo de "HTML: *HyperText Markup Language"*) es un lenguaje de composición de documentos y especificación de ligas de hipertexto que define la sintaxis y coloca instrucciones especiales que no muestra el navegador, aunque sí le indica cómo desplegar el contenido del documento incluyendo texto, imágenes y otros medios soportados. (13) Se utiliza para detallar la estructura y el contenido en forma de texto y complementarlo con objetos tales como imágenes.

1.7.2.2 JavaScript

JavaScript es un lenguaje de programación interpretado, se define como orientado a objetos, basado en prototipos, imperativo y dinámico. Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web, permitiendo mejoras en la interfaz de usuario y páginas web dinámicas (14). Dispone de un conjunto de componentes para incluir dentro de una aplicación web, como cuadros, áreas de texto, campos numéricos y editores HTML. En el presente trabajo se emplea en el desarrollo de las validaciones de los datos en el lado del cliente.

1.7.3 Herramienta CASE *Visual Paradigm Suite* 5

Visual Paradigm es una potente herramienta CASE[10] empleada para visualizar y diseñar elementos de software, para ello utiliza el lenguaje UML, proporciona a los desarrolladores una plataforma que les permite diseñar un producto con calidad de forma rápida. Facilita la interoperabilidad con otras herramientas CASE como *Rational Rose*. Se integra con *NetBeans* (de *Sun*), Eclipse (de IBM), *JDeveloper* (de *Oracle*) y *JBuilder* (de *Borland*). *Visual Paradigm* es la herramienta CASE que se emplea en la modelación de este proyecto por su característica de ser multiplataforma y por las facilidades que brinda. Ofrece entorno de creación de diagramas para UML 2.0, disponibilidad en múltiples plataformas y puede integrarse en los principales IDE (acrónimo de "IDE: I*ntegrated Development Environment",* en español, entorno de desarrollo integrado). Soporta una gama de lenguajes en la generación de código e ingeniería inversa como Java, C++, CORBA IDL, PHP, Esquema de XML, Ada y *Python*. La generación de código soporta C #, VB .NET, Lenguaje de Definición de Objeto (ODL), *Flash Action Script, Delphi, Perl,* Objetivo-C, y *Ruby*. (15)

1.7.4 *Framework* de Desarrollo

Un *framework* (marco de trabajo), en el desarrollo de software, es una aplicación con una estructura definida, conceptos, prácticas y criterios del desarrollo de un software, normalmente con artefactos o módulos de software concretos, que permite de una forma más fácil y organizada la creación de otro proyecto de software (16).

1.7.4.1 *Symfony* 2.2

Symfony es un *framework* PHP que facilita el desarrollo de las aplicaciones web. *Symfony* se encarga de todos los aspectos comunes y aburridos de las aplicaciones web, dejando que el programador se dedique a aportar valor desarrollando las características únicas de cada proyecto. También aumenta exponencialmente la productividad y ayuda a mejorar la calidad de las aplicaciones web aplicando todas

[10] *Computer Aided Software Engineering* (Ingeniería de Software Asistida por Computadora)

las buenas prácticas y patrones de diseño que se han definido para la web. (17) Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Fue utilizado para el desarrollo del subsistema de gestión y servicios web. Entre las principales características que este *framework* posee están (18):

- Se construye a base de *bundles* (*plugins* en *Symfony* 1).
- Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web.
- Flexible, debido a que cuenta con un *micro-kernel* basado en un contenedor de inyección de dependencia y un manejador de eventos.
- Proporciona las herramientas que en gran medida mejoran la productividad de los desarrolladores, como la barra de depuración web, soporte nativo de entornos y páginas detalladas de errores.
- Tomó lo mejor de los conceptos de otros *frameworks* de desarrollo como *Django, Spring* y *Ruby on Rails*. También aprovecha componentes de *Zend Framework* y de *Doctrine*.
- Proporciona seguridad integrada y promueve el desarrollo web utilizando buenas prácticas.
- Fácil de instalar y configurar.
- Independiente del sistema gestor de bases de datos.

1.7.4.2 Framework Qt 4.8.3

Qt es un *framework* multiplataforma para desarrollar interfaces gráficas de usuario. Uno de los rasgos principales que presenta esta herramienta, es que utiliza como lenguaje nativo a C++, lo que favorece la creación de aplicaciones sustentadas en la fortaleza de este lenguaje. Para el desarrollo del subsistema de servicios de comunicación fue utilizada la versión Qt 4.8.3. Entre las principales características que este *framework* posee están:

- Contiene una serie de librerías y clases con una gran herencia de programación orientada a objetos.
- Se dispone de buena cantidad de documentación.
- Cuenta con una arquitectura lista para *plugins*, permitiendo el desarrollo mediante la integración con otros tipos de productos.

- Posee la posibilidad del desarrollo bajo otros lenguajes, entre los que se encuentran: *PythonQt bindings* para *Python*, *Qyoto bindings* para C# u otros lenguajes .NET y PHP-Qt *bindings* para PHP.

1.7.5 *NetBeans* 7.3

NetBeans es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Existe un número importante de módulos para extenderlo. *NetBeans* IDE es un producto libre y gratuito sin restricciones de uso. (19). La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Este es el entorno de desarrollo seleccionado para la realización del subsistema de gestión y servicios web de la interfaz de software.

1.7.6 Qt *Creator 2.3*

Qt *Creator* es un Entorno Integrado de Desarrollo (IDE) creado por *Trolltech*, multiplataforma, diseñado para hacer que el desarrollo en C++ de la aplicación Qt sea más rápido y fácil. Algunas de sus principales características son (20):

- Posee un avanzado editor de código C++.
- Soporta los lenguajes: C#/.NET *Languages* (Mono), *Python, PyQt* y *PySide,* Ada, Pascal, *Perl*, PHP y *Ruby.*
- Posee también una GUI integrada y diseñador de formularios.
- Herramienta para proyectos y administración.
- Ayuda sensible al contexto.
- Depurador visual.
- Resaltado y auto-completado de código.
- Soporte para refactorización de código.

Se utilizó la versión 2.3 para la realización del subsistema de servicios de comunicación.

1.7.7 Servidor web: Apache 2

Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Entre sus principales características se encuentran (21):

- Corre en una multitud de sistemas operativos, lo que lo hace prácticamente universal.
- Es una tecnología gratuita de código fuente abierto.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado *script* cuando ocurra un error en concreto.
- Tiene una alta configurabilidad en la creación y gestión de *logs*[11]. Este permite la creación de ficheros de *log* a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor.

1.8 Conclusiones

En este capítulo se han caracterizado los principales términos asociados al tema, lo cual permitió una mayor comprensión del objeto de estudio, realizando un análisis de los factores que influyen en la situación problémica. A partir del análisis de las diferentes vías de conexión existentes al módulo de *Call Center*, se definió que la vía más factible para dar solución al problema es a través de una interfaz de software, utilizando servicios web. También, la caracterización de las diferentes herramientas y tecnologías a utilizar, posibilitó tener mayor seguridad para hacer uso de las mismas en aras de cumplir el objetivo general.

[11] Registro de errores.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA, EXPLORACIÓN Y PLANIFICACIÓN

2.1 Introducción

En el presente capítulo se hace una descripción de las características del sistema a desarrollar. Se realiza el análisis a través de las fases de Exploración y Planificación propias de la metodología de desarrollo utilizada, donde se confeccionan las Historias de Usuarios (HU) para cada iteración definida por el equipo de desarrollo.

2.2 Propuesta del Sistema

La solución propuesta está diseñada para que sea utilizada por cualquier equipo de desarrollo, que necesite para la implementación de determinado software una o varias funcionalidades que brinda el módulo de *Call Center* del *Elastix*. La idea consiste en que cada usuario puede utilizar las funcionalidades que tenga autorización a modo de servicio web. El sistema está compuesto por dos subsistemas, uno encargado de la gestión de usuarios y servicios web y el otro de los servicios de comunicación con el *Call Center*, como pueden observar en el diagrama siguiente:

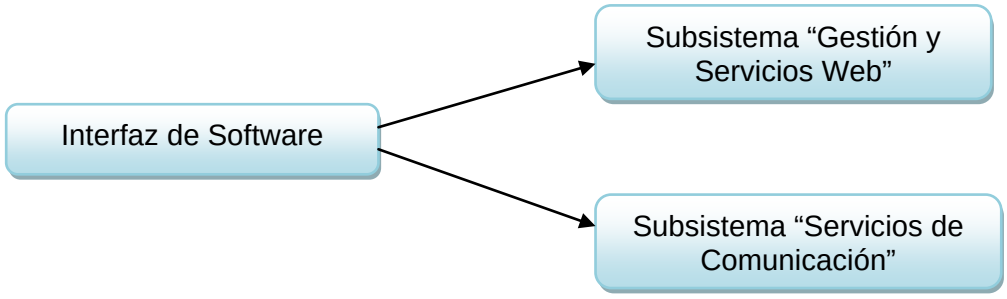


Figura 2: Composición de la interfaz de software

La interfaz de software puede estar instalada en uno o dos servidores, pues los subsistemas pueden estar en computadoras distintas como se representa en la figura 3. Una vez que el usuario de aplicaciones externas tenga que acceder a consumir un servicio web, este pedido es enviado al subsistema de gestión

y servicios web, el cual comprueba que dicho usuario tiene autorización a acceder a esta funcionalidad. En caso afirmativo el servicio web accede a través del protocolo ECCP al subsistema de servicios de comunicación, el cual obtiene la información del *Call Center* del *Elastix* mediante el protocolo TCP/IP. En el caso que no tenga permisos se notifica un error al cliente.

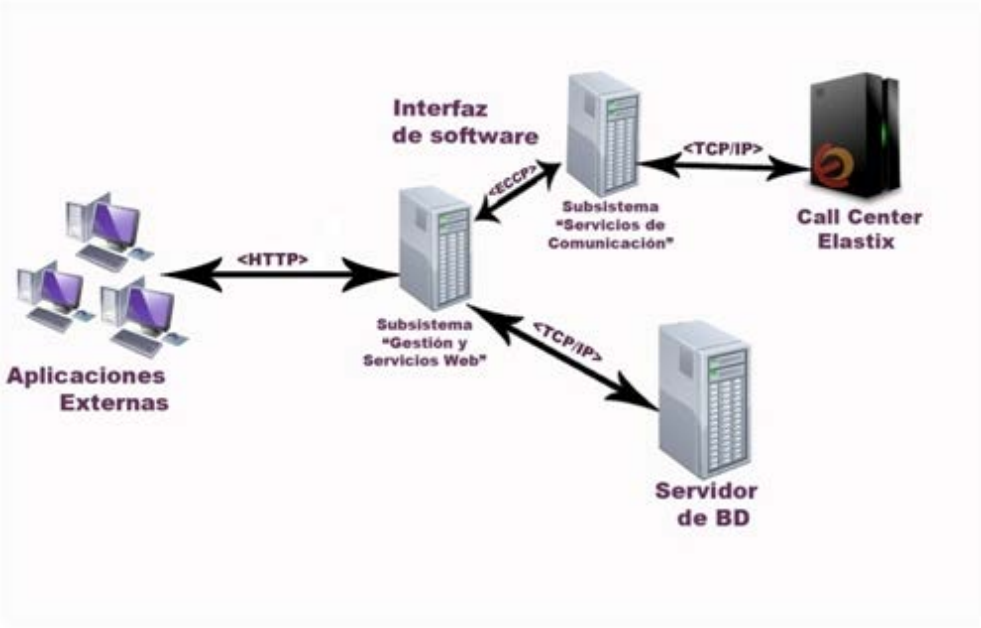


Figura 3: Propuesta de Solución

2.2.1 Características de la solución

Dentro de las características distintivas que tiene la interfaz es que determina los permisos de acceso apropiados y para ello implementa una lista de control de acceso o ACL (*Access Control List* por sus siglas en inglés). Se tiene un control para poder acceder a la interfaz de software, desde el subsistema de gestión y servicios web, permitiendo o denegando dicho acceso a partir de una condición establecida, la cual es la dirección de red desde la cual se intenta entrar. La restricción radica en permitir a aplicaciones externas conectarse solamente desde determinadas direcciones de red para que hagan uso de los servicios web.

Una de las facilidades que aporta el sistema es que las aplicaciones que necesiten utilizar el *Call Center* no tienen que implementar para el desarrollo de las mismas el protocolo ECCP, el cual está implementado únicamente en los subsistemas de la interfaz, evitando que cada software tenga que implementar la conexión con ECCP y que el equipo de programadores tenga que conocer sobre cómo realizar esta conexión y trabajar con el protocolo. La solución planteada resuelve este problema a través de la interfaz, pues solo hay que saber cómo se consume un servicio web.

2.2.2 Descripción del proceso de negocio

Para mayor comprensión del funcionamiento de la interfaz de software resulta necesario explicar el proceso de negocio, basándose en el usuario de aplicaciones externas que inicia el proceso de conexión al *Call Center*, para hacer uso de una de sus funcionalidades. La descripción del proceso está enfocada a mostrar cual es el flujo automatizado de comunicación entre la aplicación externa y el *Call Center* (Ver figura 4).

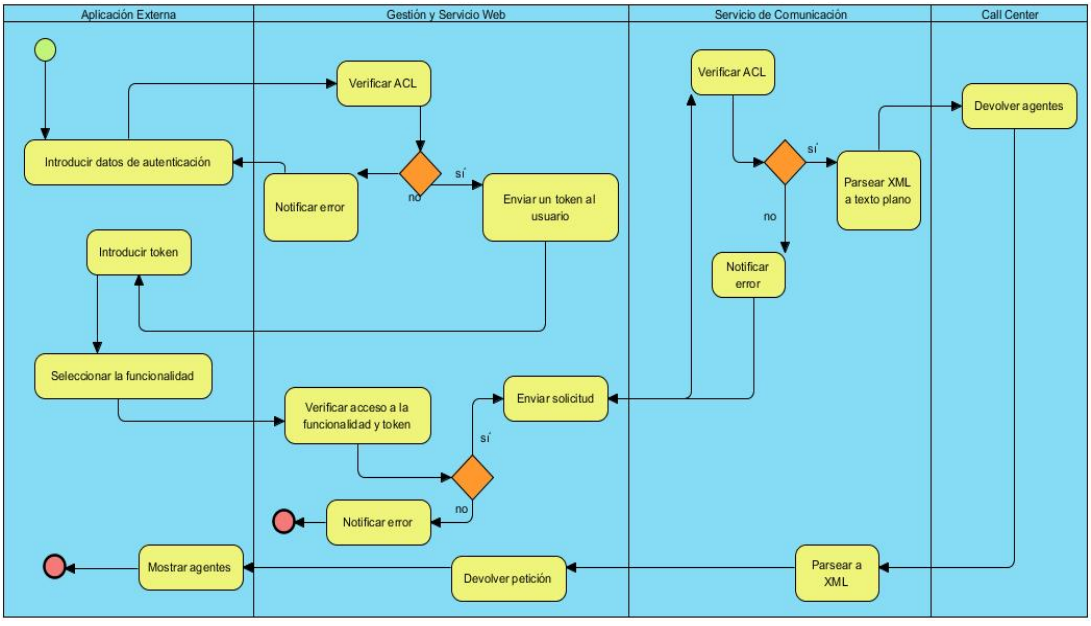


Figura 4: Proceso automatizado para la conexión

Para ello, el usuario ingresa desde su aplicación los datos para autenticarse, los cuales son verificados por el subsistema de gestión y servicios web (en lo adelante SGSW), además de comprobar la lista de control de acceso (dirección IP desde donde se desea acceder). En caso negativo, se notifica un error al usuario y en el caso contrario se envía un *token* de seguridad al mismo, el cual debe ingresar desde su aplicación y seleccionar la funcionalidad obtener estado de agente. Estos datos son enviados al SGSW, donde se verifica el *token* y si el usuario tiene acceso a la funcionalidad solicitada, en caso negativo se notifica un error y en caso positivo es enviada la solicitud al subsistema de servicios de comunicación (en lo adelante SSC). El SSC verifica las ACL para comprobar que se puede establecer la conexión entre los subsistemas, de no poder notifica un error al SSC y si es posible abre una conexión, toma el XML enviado y lo parsea a texto plano. Una vez interpretado lo que se solicita se pide al *Call Center* la información, el cual devuelve la respuesta al SSC. Este subsistema realiza el proceso inverso, llevando la información recibida a XML y la envía al SGSW. Este último devuelve a la aplicación externa la solicitud, la cual puede ser mostrada por el usuario en la misma.

2.3 Funcionalidades de la interfaz de software

La interfaz de software para la conexión de aplicaciones externas al módulo de *Call Center* del *Elastix* debe permitir:

- Gestionar usuarios.
    1. Insertar usuario.
    2. Modificar usuario.
    3. Eliminar usuario.
    4. Buscar usuario.
- Autenticar usuario.
- Gestionar funcionalidades.
    1. Insertar funcionalidad.
    2. Editar funcionalidad.
    3. Eliminar funcionalidad.

- Gestionar roles.
    1. Insertar rol.
    2. Editar rol.
    3. Eliminar rol.
- Gestionar permisos de usuarios.
    1. Asignar permisos a un usuario determinado.
    2. Modificar permisos a un usuario determinado.
    3. Eliminar permisos a un usuario determinado.
    4. Mostrar permisos de un usuario determinado.
- Gestionar la lista de control de acceso para aplicaciones externas.
    1. Insertar regla al usuario de aplicación externa.
    2. Modificar regla al usuario de aplicación externa.
    3. Eliminar regla al usuario de aplicación externa.
- Generar reporte.
- Implementar el protocolo ECCP en el subsistema de gestión y servicios web.
- Implementar los servicios web de las funcionalidades del *Call Center* para aplicaciones externas:
    1. Originar llamada.
    2. Obtener estado de agente.
    3. Loguear agente.
    4. Desloguear agente.
    5. Poner llamada en espera.
    6. Recuperar llamada en espera.
    7. Transferir llamada.
    8. Poner agente en pausa.
    9. Liberar agente que está en pausa.
    10. Obtener listado de tipos de pausas.
    11. Colgar llamada que atiende el agente.
- Establecer conexión entre subsistemas.

- Gestión de los *logs* del subsistema de servicios de comunicación.
- Establecer conexión con el *Call Center* del *Elastix* en el subsistema de servicios de comunicación.

2.4 Lista de Reserva del Producto

Para el desarrollo de la interfaz de software se confecciona la Lista de Reserva del Producto, la cual define las propiedades o cualidades que el producto debe tener; son las características que lo hacen atractivo, usable, rápido y confiable. Para lograr la satisfacción del cliente y una buena calidad en el sistema se listaron los siguientes.

2.4.1 Usabilidad

No es necesaria una preparación previa para utilizar el sistema, pues este puede ser usado por personas con conocimientos básicos en el trabajo con servicios web.

2.4.2 Disponibilidad

El sistema estará disponible las 24 horas del día para que las aplicaciones externas puedan hacer uso del mismo en el momento que lo necesiten. Esta característica es fundamental debido a que continuamente se actualizan los datos del *Call Center*.

2.4.3 Hardware

Para la instalación de la aplicación se debe disponer de una computadora de 512 MB de RAM o superior, además de 60 GB de disco duro o superior. También puede estar instalado en dos computadoras con estas prestaciones mínimas.

2.4.4 Software

En las computadoras clientes se requiere un sistema operativo con navegadores *Internet Explorer* o *Mozilla Firefox*. En el servidor donde está la interfaz se necesita de un intérprete del lenguaje PHP y en otro servidor se debe tener *Elastix, c*omo distribución de software libre de comunicaciones unificadas que integra una interfaz web y con su módulo de *Call Center* instalado.

2.4.5 Interfaz de usuario

La aplicación posee una interfaz sencilla, dirigida a las personas que se relacionen con el sistema. El diseño se realiza siguiendo las pautas de diseño para los productos que son creados en la UCI, teniendo en cuenta los colores y la marca correspondiente, que por ser un software relacionado con la Telemática sería Xilema y por tanto, es el azul el color que debe predominar.

2.4.6 Restricciones de diseño e implementación

Se hace uso del estándar de codificación que proponen los *framework* utilizados.

2.5 Personas relacionadas con el sistema

Se define como personas relacionadas con el sistema a aquellas que tienen permisos para utilizar los diferentes servicios web desde aplicaciones externas, además de los administradores que estén definidos.

Tabla 1: Descripción de personas relacionadas con el sistema.

Personas relacionadas con el sistema.	Justificación.
Administrador	Persona encargada de administrar los servicios web y de la configuración del sistema.
Cliente de aplicaciones externas	Persona encargada de consumir los servicios que le están permitidos.

2.6 Fase de Exploración

2.6.1 Historias de Usuario

En XP la gestión de requisitos es simple, el cliente escribe las historias de usuarios tal y como ven las necesidades del sistema. Son descripciones cortas y escritas en el lenguaje del usuario, sin terminología técnica. A continuación se evidencias algunas de las HU pertenecientes al tipo de prioridad alta:

Tabla 2: HU # 9 Implementar el protocolo ECCP en el Subsistema de Gestión y Servicios Web.

Historia de usuario	
Número: 9	**Nombre Historia de Usuario:** Implementar el protocolo ECCP en el Subsistema de Gestión y Servicios Web
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Lester González López José Carlos Pérez Zamora	**Iteración Asignada:** 3
Prioridad en Negocio: Alta	**Puntos Estimados:** 1/3
Riesgo En Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Se debe implementar en el SGSW el protocolo ECCP para abrir y cerrar la conexión con el subsistema de servicios de comunicación.	
Observaciones:	

Tabla 3: HU # 14 Transferir llamada.

Historia de usuario	
Número: 14	**Nombre Historia de Usuario:** Transferir llamada
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Lester González López	**Iteración Asignada:** 3

José Carlos Pérez Zamora	
Prioridad en Negocio: Alta	**Puntos Estimados:** 1/3
Riesgo En Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Al solicitar el servicio el cliente debe de ingresar la extensión a transferir la llamada.	
Observaciones:	

Tabla 4: HU # 15 Agente en *break* o pausa.

Historia de usuario	
Número: 15	**Nombre Historia de Usuario:** Agente en *break* o pausa
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Lester González López José Carlos Pérez Zamora	**Iteración Asignada:** 3
Prioridad en Negocio: Alta	**Puntos Estimados:** 1/3
Riesgo En Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Esta HU debe permitir el trabajo con pausas o *break* para los agentes, pues deben listarse los tipos de pausas, así como poner o liberar a un agente de su estado de pausa.	

Observaciones: Se deben de llenar todos los campos obligatorios.

Tabla 5: HU # 16 Establecer conexión entre subsistemas.

Historia de usuario	
Número: 16	**Nombre Historia de Usuario:** Establecer conexión entre subsistemas.
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Lester González López José Carlos Pérez Zamora	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 1/3
Riesgo En Desarrollo: Medio	**Puntos Reales:** 1
Descripción: Es la que establece una nueva conexión, la abre y la cierra. Se encuentra implementada en el subsistema de servicios de comunicación.	
Observaciones:	

Tabla 6: HU # 18 Establecer conexión con el *Call Center* del *Elastix*.

Historia de usuario	
Número: 18	**Nombre Historia de Usuario:** Establecer conexión con el *Call Center* del *Elastix*

Modificación de Historia de Usuario Número: Ninguna	
Usuario: Lester González López José Carlos Pérez Zamora	**Iteración Asignada:** 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 2/3
Riesgo En Desarrollo: Medio	**Puntos Reales:**
Descripción: Es la que se encarga de abrir la conexión al *Call Center*, enviar los datos y recibirlos, así como de cerrar la conexión.	
Observaciones:	

2.7 Planificación

Durante la fase de planificación se realiza una estimación del esfuerzo que cuesta implementar cada Historia de Usuario. Este se expresa utilizando el punto como medida, el cual se considera la unidad como una semana ideal de trabajo donde los miembros del equipo de desarrollo trabajan el tiempo planeado sin interrupción alguna.

2.7.1 Estimación de esfuerzo por Historias de Usuario

A continuación se presenta la estimación del esfuerzo por cada HU propuesta para la interfaz de software:

Tabla 7: Estimación de esfuerzo por Historia de Usuario.

Historia de Usuario	Puntos de Estimación
1. Gestionar usuarios	1
2. Autenticar usuario	1
3. Gestionar funcionalidades	1
4. Gestionar roles	1

5. Gestionar permisos a aplicaciones externas	1
6. Gestionar la Lista de Control de Acceso	1
7. Diseñar la base de datos	1
8. Generar reporte	1
9. Implementar protocolo ECCP en el subsistema de gestión y servicio web	1
10. Originar y colgar llamada	1
11. Obtener estado de agente	1
12. Registro de agentes	1
13. Llamada en espera	1
14. Transferir llamada	1
15. Agente en *break* o pausa	1
16. Establecer conexión entre subsistemas	1
17. Gestión de *logs* del subsistema de servicios de comunicación	1
18. Establecer conexión con el *Call Center* del *Elastix*	2

2.7.2 Plan de Iteraciones

El plan de iteraciones consiste en seleccionar las Historias de Usuarios que corresponden a cada iteración. Se establece una división de tres iteraciones.

2.7.2.1 Iteración 1

En la iteración 1 se lleva a cabo el desarrollo de las HU de la 1 a la 8, relacionadas con la gestión de usuarios, funcionalidades, roles y los permisos.

2.7.2.2 Iteración 2

En la iteración 2 se lleva a cabo el desarrollo de las HU de la 16 a la 18, relacionadas con la implementación del subsistema de servicios de comunicación.

2.7.2.3 Iteración 3

En la iteración 3 se lleva a cabo el desarrollo de las Historias de Usuario de la 9 a la 15, las cuales están vinculadas a los diferentes servicios web del *Call Center* que son brindados a aplicaciones externas. Al terminar esta iteración se tiene la versión 1.0 del producto final.

2.7.3 Plan de duración de las iteraciones

Se encarga de mostrar el orden en que se implementa cada Historia de Usuario en cada iteración así como la duración estimada de la misma.

Tabla 8: Plan de duración de las iteraciones.

Iteración	Orden de las Historias de Usuarios a implementar	Duración total
1	• Gestionar usuarios • Autenticar usuario • Gestionar funcionalidades • Gestionar roles • Gestionar permisos a aplicaciones externas • Gestionar la Lista de Control de Acceso • Diseñar la base de datos • Generar reporte	8 semanas
2	• Establecer conexión entre subsistemas • Gestión de *logs* del subsistema de servicios de comunicación • Establecer conexión con el *Call Center* del *Elastix*	4 semanas

3	• Implementar protocolo ECCP en el subsistema de gestión y servicio web • Originar y colgar llamada • Obtener estado de agente • Registro de agentes • Llamada en espera • Transferir llamada • Agente en *break* o pausa	7 semanas

2.7.4 Plan de entregas

El plan de entrega detalla la fecha de fin de cada iteración. Se traza en función de dos parámetros: tiempo de desarrollo ideal y grado de importancia para el cliente.

Tabla 9: Plan de entregas.

Sistema	Final Iteración 1	Final Iteración 2	Final Iteración 3
Versión 1.0	11 de marzo de 2013	8 de abril de 2013	27 de mayo de 2013

2.8 Conclusiones

En el presente capítulo se realizó una caracterización de la propuesta del sistema, lo cual facilitó la comprensión del proceso automatizado del negocio para la posterior implementación. La confección de las Historias de Usuarios (HU) a partir de las necesidades del cliente permitió la planeación del desarrollo del software, al lograr una estimación del tiempo para ello, dejando establecido un plan de iteraciones y entregas para el producto.

CAPÍTULO 3: DISEÑO, IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

En este capítulo se define la arquitectura y los patrones arquitectónicos utilizados en cada subsistema, se realiza el modelo físico de la base de datos y se definen cada una de las tareas de la ingeniería asociadas a cada Historia de Usuario, con el fin de que sean implementadas cada una de las funcionalidades hasta terminar el producto y realizar las pruebas de aceptación.

3.2 Arquitectura

La arquitectura no es más que una vista estructural de alto nivel que define los estilos o grupos de estilos adecuados para cumplir con las características no funcionales de un software. Esta viene siendo el resultado de ensamblar determinado número de elementos arquitectónicos para satisfacer varios requisitos o funcionalidades. Debido a que los sistemas de software evolucionan de manera tal que resulta complejo que sean diseñados, especificados y entendidos por un solo individuo, es que la arquitectura de software juega un rol fundamental como disciplina. (22). La arquitectura utilizada para los dos subsistemas es la cliente – servidor (Ver figura 5). Esta tecnología consiste básicamente en un cliente que realiza peticiones a otro programa del servidor a través de la red, el cual consulta en caso de ser necesaria la base de datos y le da respuesta al cliente.

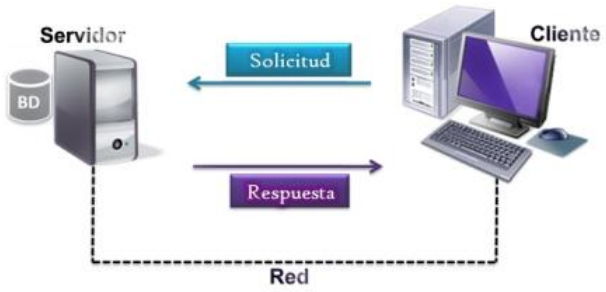


Figura 5: Arquitectura Cliente – Servidor

Los patrones o estilos arquitectónicos son patrones de diseño de software que brindan soluciones a problemas de arquitectura. Estos expresan un esquema de organización estructural para el desarrollo de un software, que consta de subsistemas, sus responsabilidades e interacciones. Para la creación del subsistema de gestión y servicios web fue utilizado el estilo arquitectónico Modelo – Vista – Controlador (MVC) y para el subsistema de servicios de comunicación el patrón de arquitectura de n-capas.

3.2.1 Estilo arquitectónico Modelo – Vista – Controlador

El *framework Symfony* 2.2 está desarrollado bajo las restricciones, suposiciones y recomendaciones de la arquitectura Modelo – Vista – Controlador. MVC sugiere la separación del software en tres componentes (Ver figura 6): el modelo, la vista y el controlador, los cuales se describen a continuación:

- Modelo: consiste en la representación de la información que maneja la aplicación. El modelo en sí lo constituyen los datos puros y la lógica de los propios datos, que puestos en el contexto del sistema proveen de información al usuario y en algunos casos a la propia aplicación. Este está representado en las clases "*Entity*" que posee *Symfony*.
- Vista: es la representación del modelo en forma gráfica y disponible para la interacción con el usuario. En el caso de una aplicación web, la "Vista" sería una página HTML con contenido dinámico sobre la que el usuario puede realizar sus operaciones. Estas se representan en las "*Views*" que trae *Symfony*, las cuales poseen la extensión *html.twig*.
- Controlador: es la parte encargada de manejar y responder las solicitudes del usuario, procesando toda la información necesaria y modificando el "Modelo" en caso de ser necesario. Este está constituido por las clases "*Controller*" de la aplicación.

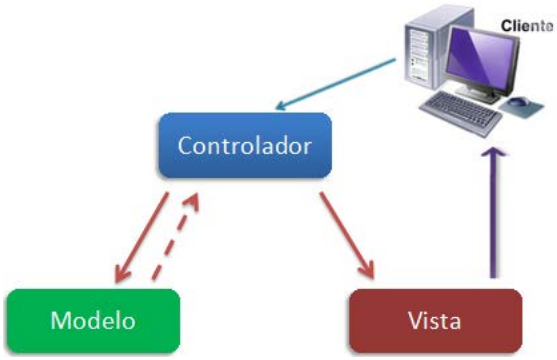


Figura 6: Patrón arquitectónico MVC

El ciclo de vida del MVC es normalmente representado por los tres componentes presentados y el cliente (también conocido como usuario o actor). El primer paso es cuando el usuario hace una solicitud al controlador con información sobre lo que desea realizar. Seguidamente el controlador decide a quién debe delegar la tarea y es aquí donde el modelo empieza su trabajo. En esta etapa, el modelo se encarga de realizar operaciones sobre la información que maneja para satisfacer el pedido del controlador. Al finalizar este proceso, el modelo devuelve al controlador la información resultante de sus operaciones, el cual a su vez redirecciona a la vista, la cual se encarga de transformar los datos en información visualmente entendible para el usuario. (23)

3.2.2 Estilo arquitectónico n-capas

La programación en capas no es más que un estilo arquitectónico donde el objetivo principal es separar los diferentes aspectos del desarrollo, tales como cuestiones de presentación, lógica de negocio y mecanismos de almacenamiento. El subsistema de servicios de comunicación se realizó utilizando este estilo, por lo que se definieron las siguientes capas: ACL, *Connection Manager*, *Interpreted*, *Connection Manager Call Center* y la de los *Logs*. En esta última es donde se registran todos los errores que ocurren en las restantes capas, por lo que se representa en una única partición vertical, relacionándose con las demás, las cuales integran una segunda partición vertical. Aquí, la *Connection Manager* es la encargada

de que el SSC interactúe con el SGSW, abriendo para ello la conexión entre ambos subsistemas. La ACL es la que verifica que se pueda establecer la conexión y en el *Interpreted* es donde se implementa el ECCP. En la *Connection Manager Call Center* se realiza la conexión a las funcionalidades del *Call Center*, permitiendo que el SSC se relacione con este módulo del *Elastix*. A continuación les ofrecemos una figura donde se representan las mismas:

Figura 7: Patrón arquitectónico "n-capas"

Su principal ventaja radica en que el desarrollo se puede llevar a cabo en varios niveles, donde si existiera algún cambio, solo se trabajaría en el nivel requerido sin revisar entre el código mezclado.

3.3 Patrones de Diseño

Un patrón de diseño es una descripción de clases y objetos, comunicándose entre sí, adaptada para solucionar un problema en específico de diseño general. Puede identificar clases, instancias, roles, colaboraciones y la distribución de responsabilidades. Son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción e interfaces. También constituyen el esqueleto de las soluciones a problemas comunes en el desarrollo de software. (24)

Para el diseño de la interfaz de software se tuvieron en cuenta los patrones GRASP[12].

3.3.1 Patrones GRASP

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

- **Patrón creador:**

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

  Solución: se aplica para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto sólo pueda ser creada por el objeto que contiene la información necesaria para ello. Entonces, la solución está en asignar a una clase la responsabilidad de crear una instancia de otra clase.

  Por ejemplo, la clase "WebServiceController.php" es la responsable de crear una nueva instancia de la clase "ECCP.php".

- **Patrón controlador:**

  Problema: ¿Cómo atender un evento del sistema?

  Solución: el controlador es encargado de recibir la petición del usuario y en función de ella, envía la solicitud a las diferentes clases con el propósito de que sea procesada. Un controlador es un objeto de interfaz que se encarga de manejar un evento del sistema.

---

[12] *General Responsibility Asignment Software Patterns* por sus siglas en inglés (Patrones generales de software para asignar responsabilidades).

Las clases "DefaultController.php", "UsuarioController.php", "FuncionalidadController.php", "RolController.php" y "WebServiceController.php" son las responsables de atender un evento del sistema.

- **Patrón alta cohesión:**

Problema: ¿Cómo lograr que las clases trabajen en una misma área de aplicación?

Solución: con este patrón podemos tener las responsabilidades de una clase bien enfocadas y relacionadas. Se aplica para realizar un diseño que evite contener clases con un alto grado de abstracción, que asuman responsabilidades que puedan haber delegado a otros objetos o que tengan complejas responsabilidades.

A todas las clases le son asignadas responsabilidades con el objetivo de que trabajen en una misma área de aplicación y no tengan mucha complejidad.

- **Patrón bajo acoplamiento:**

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

Solución: el acoplamiento mide la fuerza con que una clase está conectada con otra, de esta forma una clase debe tener un mínimo de dependencia con otras clases. Es importante para realizar un diseño de clases independientes que puedan soportar los cambios de una manera fácil y que a su vez faciliten la reutilización.

A la "WebServiceController.php", le son asignadas responsabilidades de forma tal que solo se comunique con la clase "ECCP.php", garantizando un mínimo de dependencia, además del resto de las clases que no crean instancias de otras clases.

- **Patrón experto:**

Problema: ¿Cómo pretender que cada clase cumpla con la responsabilidad que le fue asignada?

Solución: asignar la responsabilidad a la clase que posee la información necesaria para cumplirla.

Las clases "Funcionalidad.php", "Rol.php" y "Usuario.php", poseen la información necesaria para cumplir con cada una de las responsabilidades que le corresponden.

3.4 Tarjetas Clase – Responsabilidad - Colaborador

La metodología XP como parte de la fase de diseño propone el modelo de Clase – Responsabilidad – Colaborador (CRC), lo que permite organizar las clases más relevantes para las funcionalidades del sistema. Un modelo CRC es una colección de tarjetas índices estándar que representan clases y se analizan basándose en sus responsabilidades con respecto al sistema. Se dividen en tres secciones, a lo largo del borde superior de la tarjeta se escribe el nombre de la clase y en el cuerpo se listan las responsabilidades a la izquierda y los colaboradores a la derecha. (9) Una responsabilidad es algo que la clase hace y los colaboradores son aquellas clases que se requieren para que una clase obtenga la información necesaria para realizar su responsabilidad. A continuación se muestran dos tarjetas CRC, el resto de las mismas, correspondientes a las clases más relevantes de la interfaz de software, pueden observarse en los anexos (**Ver Anexo 2**).

Tabla 10: Clase ServerSocket.

Clase ServerSocket	
Descripción: clase encargada de gestionar las conexiones entre los subsistemas	
Responsabilidad	**Colaborador**
Abrir conexiones	AppLogger, ConfiguracionManager Connectionsocket
Cerrar conexiones	AppLogger

Tabla 11: Clase ConnectionSocket.

Clase ConnectionSocket	
Descripción: clase encargada de establecer la conexión entre los subsistemas	
Responsabilidad	**Colaborador**
Abrir conexión	aclManager, xmlparsermanager
Cerrar conexión	
Enviar Datos	Xmlparsermanager, Eccpclassmanager
Recibir Datos	Xmlparsermanager, Eccpclassmanager

3.5 Modelo físico de la Base de Datos

Para la implementación del subsistema de gestión y servicios web se definió el siguiente modelo físico de la base de datos:

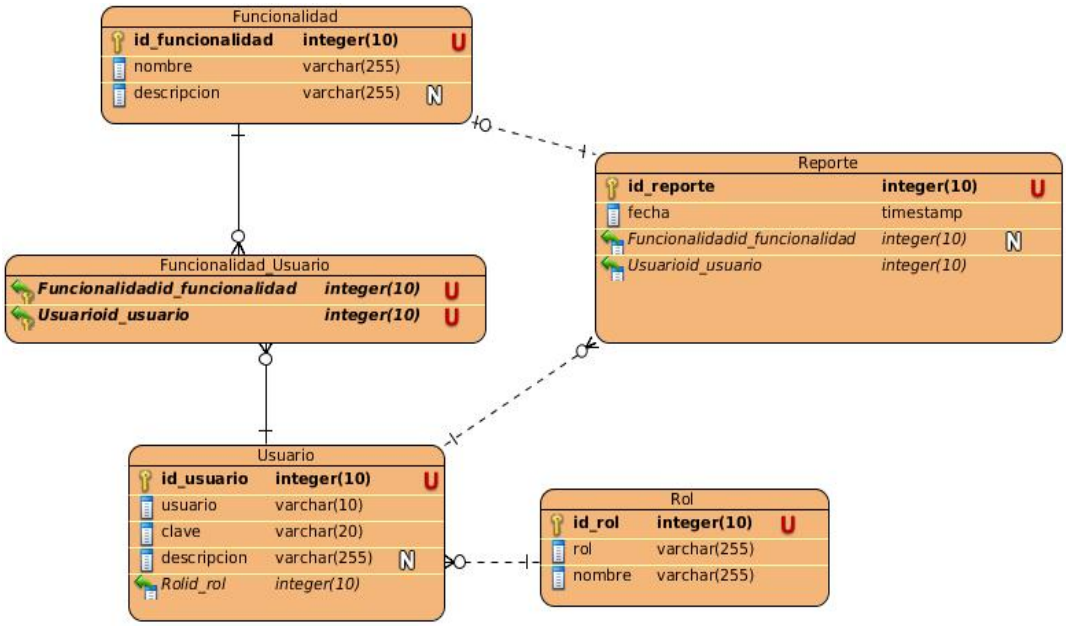


Figura 8: Modelo físico de la Base de Datos

3.6 Tareas de la Ingeniería

Las tareas de la ingeniería son descritas por el equipo de desarrollo a partir de las HU, las cuales describen tareas asociadas a cada HU, ofreciendo de esta manera más detalles para la implementación de las mismas y estimando el tiempo más cercano a la realidad.

Estas tareas se representan mediante tablas divididas por secciones. A continuación se muestran las tareas relacionadas con la HU número 16, el resto pueden observarse en los anexos (**Ver Anexo 3**):

Tabla 12: Tarea "Iniciar conexión".

Tarea de Ingeniería	
Número Tarea: 23	**Número Historia de Usuario:** 16
Nombre de Tarea: Iniciar conexión	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 2/5
Fecha de Inicio: 11/03/2013	**Fecha Fin:**12/03/2013
Programador Responsable: José Carlos Pérez Zamora	
Descripción: Verifica el IP e inicia la conexión abriendo el puerto.	

Tabla 13: Tarea "Nueva Conexión".

Tarea de Ingeniería	
Número Tarea: 24	**Número Historia de Usuario:** 16
Nombre de Tarea: Nueva Conexión	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 2/5
Fecha de Inicio: 12/03/2013	**Fecha Fin:** 14/03/2013
Programador Responsable: José Carlos Pérez Zamora	
Descripción: Abre un nodo de conexión, para que se establezca dicha conexión a través de este.	

Tabla 14: Tarea "Cerrar Conexión".

Tarea de Ingeniería	
Número Tarea: 25	**Número Historia de Usuario:** 16
Nombre de Tarea: Cerrar Conexión	
Tipo de Tarea: Desarrollo	**Puntos estimados:** 2/5
Fecha de Inicio: 14/03/2013	**Fecha Fin:** 16/03/2013
Programador Responsable: José Carlos Pérez Zamora	
Descripción: Desconecta los nodos de conexión asociados y cierra el puerto.	

3.7 Prueba

Durante esta etapa se prueban todos los componentes del producto tanto por el cliente como por el equipo de desarrollo, con el objetivo de medir la calidad del software. El proceso de pruebas está encaminado a medir el cumplimiento de las funcionalidades establecidas por el cliente, reduciendo de esta manera el número de errores no detectados. Esto es propio de la metodología XP y se efectúa en la fase de Prueba, dividiéndose en dos grupos:

- Pruebas unitarias: encargadas de verificar el código diseñado por los programadores.
- Pruebas de aceptación o pruebas funcionales: tienen como objetivo evaluar si al final de una iteración están correctas las funcionalidades requeridas por el cliente.

3.7.1 Pruebas unitarias

Estas pruebas no generan ningún artefacto, aunque resultan importantes con el fin de disminuir las no conformidades al finalizar el software. Además, resulta necesario que durante la implementación del sistema cada desarrollador tenga que ir probando constantemente lo que va realizando para garantizar que las funcionalidades exigidas por el cliente estén bien desarrolladas.

3.7.2 Pruebas de aceptación

El principal propósito de este tipo de pruebas es verificar los requisitos del sistema, enfocándose en medir sus características generales y funcionalidades. Son creadas a partir de las HU, llegando a realizarse todas las pruebas necesarias que garanticen el correcto funcionamiento de la misma. En las pruebas se especifican, desde la perspectiva del cliente, los diferentes escenarios para que una HU haya sido implementada correctamente. Al finalizar debe garantizarse que los requerimientos han sido cumplidos y que el sistema es aceptable.

Una prueba de aceptación es como una caja negra, donde cada una de ellas representa una salida esperada del sistema y es responsabilidad del cliente verificar la corrección de las pruebas. La realización de estas y la publicación de los resultados debe ser lo más rápido posible, para que los desarrolladores puedan realizar con rapidez los cambios necesarios. Las pruebas de aceptación correspondiente a cada una de las funcionalidades de la interfaz de software se representan mediante tablas dividas en las siguientes secciones:

Clases válidas: se hace la descripción de los pasos efectuados para la realización de la prueba, se tiene en cuenta las entradas válidas que hace el usuario con el fin de obtener el resultado esperado.
Clases inválidas: se hace la descripción de los pasos efectuados para la realización de la prueba, se tiene en cuenta cada una de las entradas inválidas o erróneas que hace el usuario con el fin de ver el resultado que se obtiene y cómo reacciona el sistema, para que posteriormente pueda corregirse el problema presentado.
Resultado esperado: se hace una breve descripción del resultado que se espera para cualquier tipo de entrada.
Resultado de la prueba: se hace una breve descripción del resultado de la prueba que se obtiene.
Observaciones: algún comentario que el probador desea hacerle a la sección que se prueba.

Todas las pruebas de aceptación se realizaron de conjunto con el cliente. Se detectó un total de 32 no conformidades, donde 18 fueron resueltas, 14 no procedían y no quedó ninguna sin atender, ilustrando los resultados en la siguiente tabla:

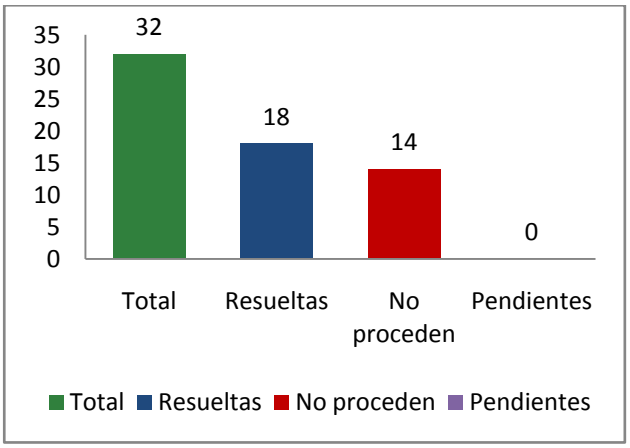


Figura 9: Gráfico de no conformidades

En los anexos (**Ver Anexo 4**) se pueden observar algunas de las pruebas relacionadas con el subsistema de gestión y servicios web. Las pruebas que evidenciaron no conformidades fueron comprobadas nuevamente después de haber sido resueltas por los desarrolladores, quedando el cliente satisfecho con el resultado de las mismas. También se cuenta con un acta de conformidad con el producto firmada por el cliente.

3.8 Conclusiones

La selección de la arquitectura de software, así como los estilos arquitectónicos utilizados y los patrones de diseño, facilitó la comprensión de la estructura del sistema y cómo están integrados sus componentes, además de proporcionar una visión global del sistema a implementar. La confección de las tarjetas CRC y las tareas de la ingeniería asociadas a las HU facilitó la construcción de la interfaz de forma ágil como plantea la metodología. También, las pruebas realizadas al sistema, demostraron la eficacia del software en cuanto a su funcionamiento y cumplimiento de los requisitos del cliente.

CONCLUSIONES GENERALES

Con la realización del presente trabajo de diploma se caracterizaron todos los procesos relacionados con el proceso de conexión de aplicaciones externas al módulo de *Call Center* del *Elastix*, facilitando mayor comprensión del objeto para el cumplimiento del objetivo general. También fueron seleccionadas las herramientas, metodología de desarrollo de software y tecnologías a emplear. Se hizo una descripción detallada del sistema desarrollado, haciendo énfasis en las diferentes fases de la metodología empleada. Se determinó la arquitectura y patrones arquitectónicos para cada subsistema del software, así como los patrones de diseño utilizados en la interfaz.

Con el desarrollo de la interfaz de software se cumple lo siguiente:

- Se garantiza el acceso a las funcionalidades del módulo de *Call Center* del *Elastix* por aplicaciones externas, sin que estas tengan que implementar el protocolo ECCP, por lo que se les facilita la implementación.
- Se puede tener control de las personas que trabajan con el *Call Center* del *Elastix*, permitiendo el acceso al mismo a través de la interfaz mediante un usuario y contraseña que se le entrega al desarrollador.
- Permite asignar a cada usuario las funcionalidades a las cuales tendrá acceso, para asegurarse de que no trabaje con aquellas que no son necesarias para el desarrollo de su aplicación.

Por lo anteriormente expuesto, se puede concluir que el objetivo general del presente trabajo se ha cumplido satisfactoriamente, sirviendo de guía cada una de las tareas definidas para la modelación del sistema.

RECOMENDACIONES

Después de exponer las conclusiones del presente trabajo de diploma se enumeran las siguientes recomendaciones para posibles mejoras:

1. Ampliar los servicios que se brindan para aplicaciones externas con el resto de las funcionalidades que ofrece el *Call Center*.
2. Crear un módulo para la interfaz donde se pueda obtener los diferentes reportes que genera el *Call Center*.
3. Publicar el presente trabajo para que sea utilizado por otras instituciones que trabajan con el *Call Center.*

REFERENCIAS BIBLIOGRÁFICAS

1. Landívar, Edgar. *Comunicaciones Unificadas con Elastix.* 2008-2009. pág. 256. Vol. I.

2. *Manual del Usuario en Español, Elastix 0.9-alpha.* Ecuador : Emp. Palosanto Solutions, 2009.

3. Sitio Oficial de la Comunidad Elastix. *Sitio Oficial de la Comunidad Elastix.* [En línea] [Citado el: 25 de marzo de 2013.] http://www.elastix.org.

4. Autores, Colectivo de. *Diccionario de informática e Internet de Microsoft.* Madrid : Impresos y Revistas S.A., 2001.

5. Autores, Colectivo de. *Diccionario Ilustrado de las Ciencias y la Tecnología.* Barcelona : Océano.

6. developerWork . [En línea] IBM. [Citado el: 20 de 03 de 2013.] http://www.ibm.com/developerworks/ssa/webservices/newto/websvc.html.

7. Reina, Ing. Oscar Andrés Rocha. *Diseño e Implementación de una aplicación para la distribución de las llamadas en múltiples servidores Asterisk manejando el ACD para Call Center bajo Linux.* Bogotá D.C. : s.n., 2009.

8. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso unificado de desarrollo de software.* Madrid : Pearson Educación, S.A., 2000.

9. Pressman, Roger S. *Ingeniería de Software.* New York : McGraw-Hill, 2010.

10. Sánchez, María A. Mendoza. Informatizate. *Informatizaje.* [En línea] 7 de junio de 2004. [Citado el: 15 de enero de 2013.] http://www.informatizate.net.

11. Portal en Español sobre PostgreSQL. *Portal en Español sobre PostgreSQL.* [En línea] 2 de octubre de 2010. [Citado el: 17 de enero de 2013.] http://www.postgresql.org.es/.

12. Autores, Colectivo de. Manual de PHP. [En línea] 27 de mayo de 2006. [Citado el: 18 de enero de 2013.] http://www.php.net/docs.php.

13. Musciano, Chuck y Kennedy, Bill. *HTML: La guía completa.* México, DF : McGraw-Hill, 1999.

14. Blogspo. [En línea] [Citado el: 20 de enero de 2013.] http://3plejlife. blogspot.com/2011/02/javascript-orientado-objetos-clases.html.

15. Características de Visual Paradigm. [En línea] [Citado el: 23 de enero de 2013.] http://www.versioncero.com/noticia/210/visual-paradigm-for-um.

16. Ecured. *Ecured.* [En línea] [Citado el: 10 de abril de 2013.] http://www.ecured.cu/index.php/Framework.

17. Symfony. [En línea] [Citado el: 29 de 11 de 2011.] http://www.symfony.es/que-es-symfony/.

18. Ecured . *Ecured.* [En línea] [Citado el: 11 de abril de 2013.] http://www.ecured.cu/index.php/Symfony.

19. Netbeans. [En línea] [Citado el: 29 de 11 de 2011.] http://netbeans.org/index_es.html.

20. Ecured. *Ecured.* [En línea] [Citado el: 12 de abril de 2013.] http://www.ecured.cu/index.php/Qt_Creator.

21. Introducción a Apache. [En línea] [Citado el: 24 de enero de 2013.] http://www.ciberaula.com.

22. Camacho, Erika, Cardeso, Fabio y Núñez, Gabril. *Arquitecturas de software.* 2004.

23. Martell Fernández, Vladimir, Figueroa Hidalgo, Daira y Vega Ortiz, Yurisbel. *Propuesta de diseño para proyectos informáticos que utilizan Symfony como Framework.* 2010.

24. García Peñalvo, Francisco José, Conde González, Miguel Ángel y Bravo Martín, Sergio. *Ingeniería de Software: diseño orientado a objetos.* Salamanca : Universidad de Salamanca, 2008.

25. Características de XML. [En línea] [Citado el: 22 de enero de 2013.] http://www.dcc.uchile.cl/~rbaeza/inf/xml.html .

26. Características de UML. [En línea] [Citado el: 21 de enero de 2013.] http://www.abcdatos.com/tutoriales/tutorial/l7157.html.

27. Dirphp. [En línea] 27 de 11 de 2011. http://www.dirphp.com/dir/Software_y_servidores_PHP/Sitio_oficial_del_Servidor_de_Base_de_datos_MySQL_50.html.

28. Ecured. *Ecured.* [En línea] [Citado el: 12 de abril de 2013.] http://www.ecured.cu/index.php/SQLite.

29. Ecured. *Ecured.* [En línea] [Citado el: 10 de abril de 2013.] http://www.ecured.cu/index.php/MySQL.

BIBLIOGRAFÍA

- Autores, Colectivo de. *Diccionario de informática e Internet de Microsoft.* Madrid : Impresos y Revistas S.A., 2001.
- Autores, Colectivo de. *Diccionario Ilustrado de las Ciencias y la Tecnología.* Barcelona : Océano.
- Autores, Colectivo de. Manual de PHP. [En línea] 27 de mayo de 2006. [Citado el: 18 de enero de 2013.] http://www.php.net/docs.php.
- Blogspo. [En línea] [Citado el: 20 de enero de 2013.] http://3plejlife. blogspot.com/2011/02/javascript-orientado-objetos-clases.html.
- Camacho, Erika, Cardeso, Fabio y Núñez, Gabril. *Arquitecturas de software.* 2004.
- Características de UML. [En línea] [Citado el: 21 de enero de 2013.] http://www.abcdatos.com/tutoriales/tutorial/l7157.html.
- Características de XML. [En línea] [Citado el: 22 de enero de 2013.] http://www.dcc.uchile.cl/~rbaeza/inf/xml.html .
- DeveloperWork . [En línea] IBM. [Citado el: 20 de 03 de 2013.] http://www.ibm.com/developerworks/ssa/webservices/newto/websvc.html.
- García Peñalvo, Francisco José, Conde González, Miguel Ángel y Bravo Martín, Sergio. *Ingeniería de Software: diseño orientado a objetos.* Salamanca : Universidad de Salamanca, 2008.
- Introducción a Apache. [En línea] [Citado el: 24 de enero de 2013.] http://www.ciberaula.com.
- Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso unificado de desarrollo de software.* Madrid : Pearson Educación, S.A., 2000.
- Landívar, Edgar. *Comunicaciones Unificadas con Elastix.* 2008-2009. pág. 256. Vol. I. *Manual del Usuario en Español, Elastix 0.9-alpha.* Ecuador : Emp. Palosanto Solutions, 2009.
- Martell Fernández, Vladimir, Figueroa Hidalgo, Daira y Vega Ortiz, Yurisbel. *Propuesta de diseño para proyectos informáticos que utilizan Symfony como Framework.* 2010.
- Musciano, Chuck y Kennedy, Bill. *HTML: La guía completa.* México, DF : McGraw-Hill, 1999.
- Portal en Español sobre PostgreSQL. *Portal en Español sobre PostgreSQL.* [En línea] 2 de octubre de 2010. [Citado el: 17 de enero de 2013.] http://www.postgresql.org.es/.

- Pressman, Roger S. *Ingeniería de Software.* New York : McGraw-Hill, 2010.
- Reina, Ing. Oscar Andrés Rocha. *Diseño e Implementación de una aplicación para la distribución de las llamadas en múltiples servidores Asterisk manejando el ACD para Call Center bajo Linux.* Bogotá D.C. : s.n., 2009.
- Sánchez, María A. Mendoza. Informatizate. *Informatizaje.* [En línea] 7 de junio de 2004. [Citado el: 15 de enero de 2013.] http://www.informatizate.net.
- Sitio Oficial de la Comunidad Elastix. *Sitio Oficial de la Comunidad Elastix.* [En línea] [Citado el: 25 de marzo de 2013.] http://www.elastix.org.
- Symfony. [En línea] [Citado el: 29 de 11 de 2011.] http://www.symfony.es/que-es-symfony/.