ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN CTI CON GEORREFERENCIACIÓN DE NÚMEROS TELEFÓNICOS FIJOS UTILIZANDO EL TAPI DE LA CENTRAL TELEFÓNICA IP OFFICE DE AVAYA

TOMO II

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

CHACÓN HERRERA DIEGO JEFFERSON
contact@chacondiego.com

DIRECTOR: ING. HIDALGO LASCANO PABLO WILLIAM
pablo.hidalgo@epn.edu.ec

Quito, Junio 2015

CONTENIDO

TOMO I

TOMO II

ÍNDICE DE FIGURAS

ILUSTRACIONES DE ANEXOS

ÍNDICE DE TABLAS

CAPÍTULO III

3. DESARROLLO, DESPLIEGUE Y PRUEBAS

3.1. FASE DE CONSTRUCCIÓN (C)

Dentro de la metodología de Proceso Unificado, la fase de construcción es aquella en la que se invierte la mayor parte de los recursos técnicos, económicos y humanos para convertir los requerimientos, diseños y prototipos generados en las etapas de inicio y elaboración en el producto final (entiéndase software informático) que solucione las necesidades de los usuarios.

Entradas de Esta Fase

Obviamente las entradas más importantes de la fase de construcción constituyen todos los resultados provenientes de las etapas de análisis y diseño de las Fases de Inicio (IA) y Elaboración (EAD). Esto quiere decir que todos los diagramas, flujos de eventos y actividades, diagramas de comunicaciones, bases de datos, protocolo de comunicaciones y diseños de interfaces serán vitales en el desarrollo de la solución.

Adicionalmente, el establecimiento de la línea base de la arquitectura y el estudio de la fase de elaboración ha dado como resultado un prototipo inicial que incluye una implementación de los casos de uso más importantes como la generación y detección de eventos del sistema y el envío y recepción de llamadas en los agentes. Dicho prototipo a su vez será utilizado como base para la construcción del sistema final que será detallado en las secciones posteriores.

Como no podía ser de otra manera, los requisitos definidos en las fases de inicio (IR) y elaboración (ER) también forman parte de las entradas de la fase de construcción debido a que éstos deben ser satisfechos adecuadamente y debe comprobarse que cumplen con las expectativas planteadas.

3.1.1. ANÁLISIS (CA)

Si bien el modelo de funcionamiento del sistema ya ha sido descrito (diseñado) con anterioridad, aún no han sido planteadas de manera específica las técnicas, herramientas y tecnologías que serán utilizadas para su desarrollo, las cuales deben ser cuidadosamente analizadas (y de ser el caso, diseñadas) para que se adapten a los objetivos globales del proyecto mientras se ajustan a las limitaciones de presupuesto, tiempo y conocimientos inherentes en su creación.

En la sección de análisis de la fase de construcción, por tanto, serán examinadas las alternativas que estén directamente relacionadas con el desarrollo e implementación del sistema, estableciendo el entorno de creación, trabajo, ejecución y despliegue del mismo.

3.1.1.1. Servidor CTI (CA-CTI)

En las fases anteriores se ha determinado que el sistema obligatoriamente deberá utilizar la siguiente tecnología:

- IA-AA-CTI: Interfaz de Programación de Aplicaciones de Telefonía (TAPI) 2.0 proporcionada por Avaya e incluida en el instalador *USER4_2_43.exe.*

Tomando en cuenta que la interfaz anterior sólo puede ser utilizada en sistemas operativos Windows, a continuación se detallarán las características del entorno de desarrollo de este módulo:

- La creación de este componente se realizará sobre un sistema operativo Microsoft Windows 7 Professional de 64 bits con *Service Pack 1* que incluya las últimas actualizaciones disponibles.
- Al tratarse de una aplicación de tipo MFC (debido a la utilización de TAPI 2.0), el Entorno de Desarrollo Integrado (IDE) utilizado para la creación de este módulo será Microsoft Visual Studio 2008 Professional Edition.

- De forma adicional, se creará una excepción en el firewall de Windows que permita la interconexión del archivo ejecutable de este módulo que será llamado ***callmanager.exe***.
- Considerando que la creación de un instalador simplifica y facilita el despliegue de este componente, se utilizará el mismo entorno de trabajo para tal propósito.

Las características del entorno de ejecución serán descritas más adelante en las disciplinas de pruebas y despliegue de la etapa de transición de este mismo capítulo.

3.1.1.2. Servidor Principal (CA-CORE)

Para la creación de este componente en fases anteriores se ha establecido lo siguiente:

- IA-AA-CORE: Por su carácter distribuido y multiplataforma, se utilizará el *framework* de desarrollo Java Enterprise Edition 6.0.

Con respecto a lo anterior para la creación del servidor principal se considerará lo siguiente:

- El sistema se desarrollará mediante el empleo del Kit de Desarrollo de Java (JDK o *Java Development Kit*) **Enterprise Edition 7** incluido en el instalador *jdk-7u67-windows-x64.exe* descargado directamente de la página del fabricante *http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html*. Debe aclararse que la decisión de elegir la séptima versión de Java EE en lugar de la sexta se debe a que la última había quedado obsoleta y su soporte era mínimo. La versión siete incluye mejoras de estabilidad, seguridad y rendimiento que de ninguna forma interfieren con las necesidades del proyecto en cuestión.
- Tomando en cuenta que la ejecución de este módulo requiere de un servidor de aplicaciones compatible con el *framework* anterior, en la tabla 3.1 se

presenta un análisis comparativo de las alternativas gratuitas más relevantes con el objeto de encontrar la propuesta más apropiada.

Producto	Características
GlassFish Open Source Edition	Gran confiabilidad y desempeño. Soportado por Oracle (empresa que posee los derechos de Java). Incluye gran variedad de características avanzadas.
JBoss AS	Bajo consumo de recursos. Optimizado para funcionar rápidamente.
Caucho Resin	Optimizado para la ejecución en la nube. Monitoreo detallado.
Apache TomEE	Curva de aprendizaje simplificada. Ligero.
Apache Geronimo	Facilidad de uso. Actualización frecuente.

Tabla 3.1 Características de los Principales Servidores Java EE Gratuitos

Aunque cualquiera de las ofertas presentadas cumple con las características y requisitos mínimos necesarios para la creación de este módulo, se ha decido utilizar la versión *Open Source* de *GlassFish* por la confiabilidad, estabilidad y buen desempeño que posee, así como por pertenecer a la empresa que proporciona el JDK subyacente. Por tal motivo, la versión del servidor de aplicaciones a emplear será la 3.1.2.2 (que puede ser instalada en sistemas Windows, Linux y AIX) disponible de forma gratuita desde la siguiente dirección: *http://glassfish.java.net/download-archive.html*. En cuanto al entorno de desarrollo tanto el JDK como el servidor de aplicaciones (*GlassFish*) serán instalados y empleados un computador que tenga características similares al utilizado en el desarrollo del servidor CTI.

En lo que respecta a la creación de la aplicación web se utilizará el modelo de diseño basado en componentes JSF versión 2.1 (incluido en el mismo JDK descrito anteriormente) que integrado con otros *frameworks* permiten mejorar la experiencia del usuario final y la calidad del producto (tipo RIA, por ejemplo). Entre las plataformas compatibles basadas en AJAX (*Asynchronous JavaScript And XML*) más conocidas se pueden mencionar:

- PrimeFaces.
- ICEfaces.
- RichFaces.
- Apache MyFaces.

Luego de analizar las alternativas presentadas, se ha escogido a ***PrimeFaces*** como válida por su facilidad de uso, reducido tamaño y mínimos requerimientos, que en su quinta versión puede ser descargada desde el siguiente enlace: *http://primefaces.org/downloads.html*.

En lo que respecta a la tecnología de acceso a datos (API de Persistencia de Java o JPA) se utilizará ***EclipseLink*** sobre alternativas como *Hibernate* u *OpenJPA* al ser la implementación de referencia para JPA 2.0, la cual puede ser obtenida de forma gratuita desde el siguiente enlace *http://www.eclipse.org/eclipselink/downloads/*. Cabe recalcar que JPA se utilizará únicamente para recuperar información inherente de la plataforma y no para extraer información de bases de datos externas (según lo especificado en la sección 2.4.2.9, EAD-BDD).

Seguidamente, como resultado del análisis IA-AR-4c, es obligatoria la utilización de un gestor de base de datos local multiplataforma, cuyas principales opciones de código abierto están presentadas en la tabla 3.2. Cabe añadir que en la tabla anterior no han sido incluidos sistemas de gestión de bases de datos de tipo NoSQL (no solo SQL) ya que en este proyecto ha sido considerado un enfoque de desarrollo tradicional (aunque es importante notar que las más recientes versiones de *EclipseLink* tienen soporte para gestores de este tipo).

Gestor BDD	Principales Características
PostgreSQL	Características avanzadas. Amplia documentación. Licencia PostgreSQL (licencia “liberal” similar a BSD o MIT). Fiel a los estándares.
MySQL	Fácil de usar. Mantenido y actualizado por Oracle. Documentación extensa. Licencia GPL o propietaria (según sea el caso).
Apache Derby	Distribuido bajo licencia Apache 2.0. Huella pequeña. Enfocada en aplicaciones móviles o de pocos datos.
Firebird	Posee características sencillas. Poca documentación.

Tabla 3.2 Principales Gestores de Bases de Datos Multiplataforma

De entre las alternativas propuestas, por tanto, se ha seleccionado a **PostgreSQL** por ser un sistema de características avanzadas completamente soportado por la comunidad y que a diferencia de su principal competidor, MySQL, no está restringido a ningún impedimento de licenciamiento para aplicaciones de código abierto, comerciales o gubernamentales (sin embargo, es válido aclarar que cualquiera de las opciones presentadas en la tabla 3.2 cumple con las características requeridas para el desarrollo de este proyecto). La versión 9.2 será la escogida para el almacenamiento y recuperación de la información la cual puede ser obtenida desde la página web oficial *http://www.postgresql.org/download/*. De igual manera será necesario descargar e instalar el controlador (*driver*) JDBC de PostgreSQL que puede ser obtenido desde la dirección *http://jdbc.postgresql.org/download.html*.

Finalmente, en lo concerniente al desarrollo mismo de este componente, es imperativo escoger un Entorno de Desarrollo Integrado (IDE) de la variedad de soluciones existentes en el mercado. De las alternativas disponibles se ha

seleccionado la plataforma de desarrollo **NetBeans IDE 7.3** debido a que incluye las siguientes características y funcionalidades:

- Multiplataforma y multilenguaje.
- Gratuito.
- Integra el servidor de aplicaciones *GlassFish* (opcional).
- Incluye herramientas que facilitan el rápido desarrollo de aplicaciones.

Entre las demás opciones disponibles compatibles con Java EE 7.0 es posible encontrar IDEs como Eclipse, IntelliJ IDEA, KDevelop o *Rational Application Developer*.

3.1.1.3. Cliente de Escritorio (CA-DES)

Al igual que con el módulo principal, este componente al ser multiplataforma será desarrollado en Java. Dado que el JDK descrito en la sección anterior incluye las herramientas y características para el desarrollo de aplicaciones Java SE (*Standard Edition*) orientadas hacia el usuario final, la creación de este componente se realizará utilizando los mismos recursos que los utilizados para el desarrollo del módulo principal (es decir, será creado en el mismo entorno de trabajo). Sin embargo, la versión de JRE (*Java Runtime Environment*) base para la ejecución del aplicativo de escritorio de los agentes será la 1.7 que puede ser descargada de la siguiente página web *http://www.java.com/es/download*.

Adicionalmente, en lo que respecta al agente de escritorio, otro motivo de análisis tiene que ver con el fácil despliegue (distribución e instalación) del mismo en los *call centers* de las instituciones. Si bien éste es un punto que no tiene mucha relevancia en el cumplimiento de los objetivos y requerimientos planteados para este proyecto, se ha considerado apropiado incluir las tecnologías que los *frameworks* de implantación de software proporcionan. En este sentido, el aplicativo de escritorio podrá ser instalado de dos formas:

- Manualmente mediante el uso de un instalador tradicional.
- Se utilizará la tecnología *Java Web Start* [E-9] que permite descargar y ejecutar aplicaciones Java desde los navegadores web. En otras palabras, conforme a

lo diseñado en la sección la sección 2.4.2.7.2 (EAD-CORE-Web en la que se determina que los agentes de call center también dispondrán de un acceso web) los usuarios tendrán la posibilidad de descargar la última versión del programa eliminando complejos procedimientos de instalación o actualización.

Por otro lado, uno de los puntos más importantes a tratar está relacionado con la búsqueda y presentación de la localización geográfica en el transcurso de una llamada telefónica. En primer lugar, es conveniente realizar un análisis comparativo de los principales servicios de presentación de mapas que proporcionan un API que puede ser utilizado por Java (ver tabla 3.3).

Servicio	Características
Google *Maps*	Poseen funcionalidades avanzadas.
Bing *Maps*	Licenciamiento gratuito únicamente para aplicaciones web.
OpenStreetMap	Proyecto colaborativo. No posee licencias de uso. Mapas menos descriptivos.
MapQuest	Dos versiones, licenciada y abierta (última basada en *OpenStreetMap*). Simple de utilizar.

Tabla 3.3 Principales Servicios de Mapas Disponibles

Nótese en la tabla anterior que los servicios de mapas de Google (*Google Maps*) y Bing (*Bing Maps*) son bastante parecidos al punto de mantener un esquema de licenciamiento similar. Ambos proporcionan un API gratuito exclusivo para el desarrollo de aplicaciones web cuyos usuarios puedan acceder gratuitamente a sus servicios, lo que significa que la inclusión de estas soluciones en una intranet (y por ende en el agente *desktop* de este proyecto) está restringida[16].

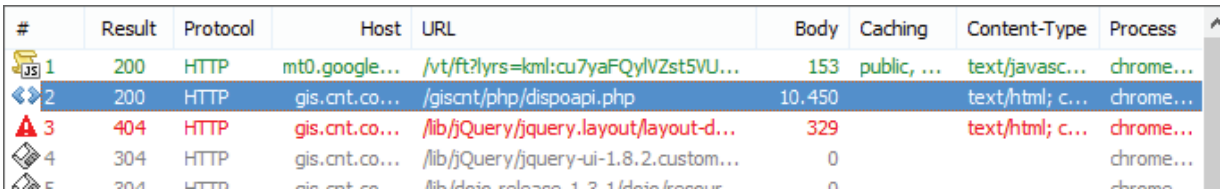
[16] Los términos de uso de *Bing Maps* pueden ser leídos en la siguiente página web *http://www.microsoft.com/maps/product/terms.html*. Las restricciones en la creación de sitios comerciales o aplicaciones internas para *Google Maps* pueden ser encontradas en la dirección *https://developers.google.com/maps/faq?hl=es#tos_commercial* donde se especifica que sería necesaria la adquisición de una licencia especial de tipo *Business* para este proyecto.

Si bien es cierto que hay alternativas que permiten utilizar estos servicios desde aplicaciones de escritorio (como la utilización de JavaFX que “simule” un entorno web, por ejemplo), se ha considerado conveniente estudiar otras alternativas gratuitas que eviten aumentar el tiempo o la complejidad en la creación del producto. *MapQuest* provee herramientas para la creación de aplicaciones de escritorio y comerciales las cuales son obtenidas mediante la adquisición de una licencia especial (*Licensed Data*), mientras que su versión gratuita tiene como base otra solución, ***OpenStreetMaps***, que al no tener ningún costo ni restricciones de licenciamiento ha sido escogida como motor de generación de mapas para este proyecto. De forma adicional, el componente *JXMapViewer* permite incrustar servicios de mapeo en aplicaciones de escritorio Java basadas en *Swing*, siendo compatible con *OpenStreetMaps*, reduciendo significativamente el tiempo de desarrollo de esta funcionalidad.

Como ha sido descrito en las secciones 2.3.4.2.2 y 2.4.2.4 (IA-AR-2c y EAD-INF respectivamente) cuando el sistema no pueda encontrar las coordenadas geográficas del número telefónico fijo en la base de datos externa de la institución, se utilizará como alternativa el empleo del servicio gratuito de disponibilidad geográfica proporcionada por la Corporación Nacional de Telecomunicaciones (CNT). Un análisis más profundo del servicio en cuestión revela que éste es provisto solamente mediante llamadas HTTP convencionales y que no existe un API específico (como un servicio web) sobre el que se pueda trabajar. La inspección del tráfico HTTP intercambiado entre el cliente (navegador web) y el servidor mediante herramientas de análisis como *Fiddler2* revela lo siguiente:

1. La primera llamada realizada por el cliente se efectúa a la dirección *http://gis.cnt.com.ec/giscnt/php/dispoapi.php* (ver figuras 3.1 y 3.2), en la cual se crea una *cookie* que posteriormente será utilizada en la búsqueda de la localización del número telefónico fijo.

Figura 3.1 Servicio de Disponibilidad Geográfica de Telefonía Fija Provisto por la CNT



#	Result	Protocol	Host	URL	Body	Caching	Content-Type	Process
1	200	HTTP	mt0.google...	/vt/ft?lyrs=kml:cu7yaFQylVZst5VU...	153	public, ...	text/javasc...	chrome...
2	200	HTTP	gis.cnt.co...	/giscnt/php/dispoapi.php	10.450		text/html; c...	chrome...
3	404	HTTP	gis.cnt.co...	/lib/jQuery/jquery.layout/layout-d...	329		text/html; c...	chrome...
4	304	HTTP	gis.cnt.co...	/lib/jQuery/jquery-ui-1.8.2.custom...	0			chrome...

Figura 3.2 Carga de la Página Principal

2. Cuando es introducido el número telefónico a buscar, el cliente realiza una llamada (solicitud POST) a la dirección *http://gis.cnt.com.ec/giscnt/php/retornarNumeroXY.php* en la cual se incluye como contenido la cadena de texto "***&numero*=22554787*&metros*=100**" para la búsqueda de las coordenadas del contacto dentro de un radio de distancia, como se muestra en la figura 3.3.

Figura 3.3 Solicitud POST de Búsqueda de Coordenadas

3. La respuesta a la solicitud anterior incluye una cadena de texto compuesta por varios valores numéricos separados por comas en los que es posible diferenciar la longitud (campo número 3) y la latitud (campo número 4), como se ejemplifica en la figura 3.4.

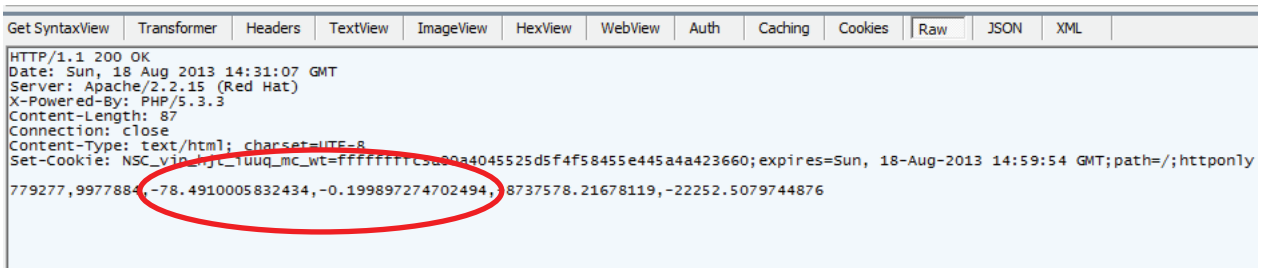


Figura 3.4 Respuesta que Contiene la Localización Geográfica del Número Telefónico Fijo

La información geográfica anterior puede ser utilizada por el aplicativo de escritorio del agente que, mediante el uso de *JXMapViewer* y *OpenStreetMaps* será capaz de presentar en un mapa la ubicación del número de contacto. Es crucial aclarar que para que lo anterior sea posible se requieren de librerías Java adicionales como ***Apache HttpComponents*** que sean capaces de manipular el protocolo HTTP desde el servidor principal realizando solicitudes y procesando respuestas.

Al ser un cliente HTTP válido, de ninguna manera se infringen reglas de mal uso del servicio, alcanzando un nivel similar al de un *web-crawler* (araña) que inspecciona las páginas web de manera automática. El hecho de que se utilice *OpenStreetMaps* (y no *Google Maps*) como servidor de mapas hace posible presentar la información geográfica en el cliente de escritorio, utilizando el servicio de CNT como un proveedor de coordenadas geográficas solamente. Dicho de otro modo, CNT provee localización mientras que *OpenStreetMaps* proporciona esquematizado.

3.1.2. IMPLEMENTACIÓN (CM)

Si bien en el análisis anterior (CA) fueron descritas las directrices de creación del proyecto es finalmente en la disciplina de implementación (CM) en la que el sistema quedará constituido en términos de componentes utilizables: ficheros de código fuente, *scripts*, ficheros de códigos binarios, archivos ejecutables, entre otros. Para explicar la composición y las interacciones de cada uno de ellos se utilizarán diagramas de secuencia, actividades y clases tomando como base los diagramas detallados en la fase de elaboración (específicamente la sección EAD).

Es imperativo tomar en cuenta que el término *implementación* aquí utilizado está relacionado con la *disciplina* de implementación *(*del inglés *implementation)* de la fase de construcción según la metodología de proceso unificado (es decir, cuando la *especificación* de la solución es *implementada* en forma de código). Si bien en la terminología de la industria de TI esta palabra está asociada con la implantación del software, en este proyecto el término que describe el procedimiento de instalación del mismo será conocido como *despliegue (deployment)*.

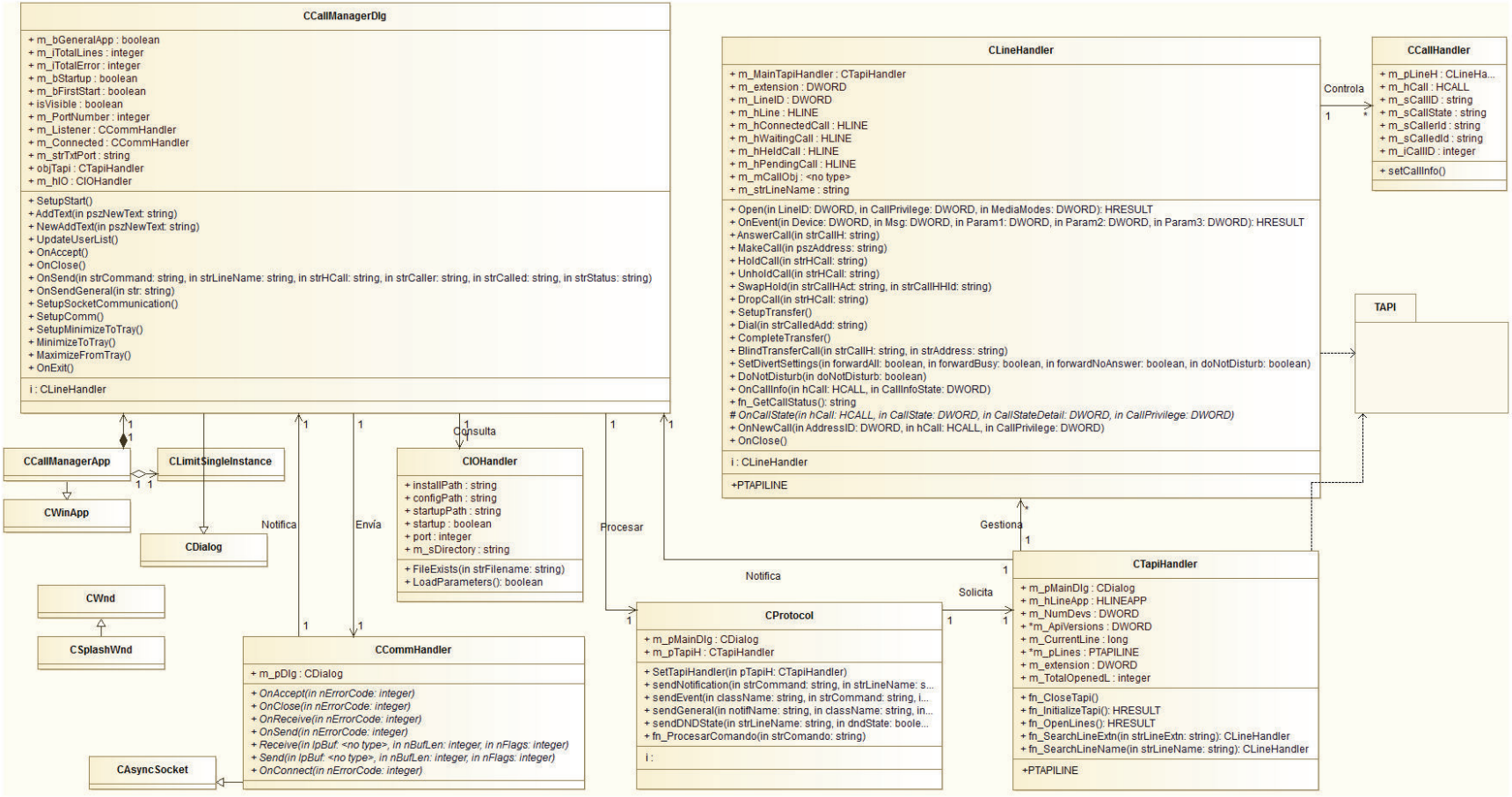


Figura 3.5 Diagrama de Clases del Servidor CTI

3.1.2.1. Servidor CTI (CM-CTI)

Primeramente se utilizará el diseño planteado en el apartado 2.4.2.6 (EAD-CTI) para crear el modelo de clases final del servidor CTI que está presentado en la figura 3.5.

Como es lógico suponer, debido a que este componente está basado en los ejemplos proporcionados por el fabricante, gran parte del modelo y código en lo referente a la gestión del TAPI y de las líneas telefónicas tendrá parecido con el original, con las obligatorias modificaciones y adiciones propias que este proyecto requiere. La más relevante tiene que ver con la gestión múltiple de líneas, ya que el ejemplo principal gestiona los eventos de una sola línea telefónica. Por otra parte, es notable aclarar que se ha considerado la existencia de una única instancia de la aplicación en ejecución con el ánimo de evitar la gestión del TAPI desde múltiples fuentes que pueden producir inconsistencias en su desempeño. Para que esto sea posible se emplea un *mutex* dentro de la clase *CLimitSingleInstance* que es llamada en el arranque del programa. De esta manera sólo puede haber una única instancia del aplicativo CTI en funcionamiento.

En lo que respecta a la iniciación misma de los servicios prestados por este módulo (como por ejemplo la carga del TAPI, la apertura de las líneas telefónicas o la escucha en el puerto de comunicaciones), ésta se realiza en la función *SetupStart* de la clase *CCallManagerDlg*, la cual está transcrita en su totalidad en la figura 3.6. Allí es posible notar que en primer lugar la interfaz de telefonía (TAPI) es inicializada mediante la invocación de la función *fn_InitializeTapi()* del objeto *objTapi,* que es la instancia de la clase *CTapiHandler.* Cuando se determina que el resultado de la iniciación de la interfaz es positivo se procede a llamar a la función *fn_OpenLines()* del mismo objeto para detectar y abrir todas las líneas activas asociadas a la central telefónica que podrán ser utilizadas por el aplicativo. Es aquí donde son invocados varios métodos del API entre ellos *lineOpen* (del TAPI) mientras se crea un objeto de tipo *CLineHandler* para cada una de las líneas detectadas. Seguidamente, *objTapi* se enlaza como referencia en la instancia de la clase *CProtocol* llamada *m_ObjProtocol* para que esta última pueda ser capaz de procesar los eventos CTI. A continuación se

inicializan los objetos *m_Listener* y *m_Connected* de la clase *CCommHandler* para la escucha y establecimiento de la conexión con el servidor principal que habilitan el envío y recepción de mensajes. Finalmente, la interfaz gráfica de usuario es actualizada mediante el llamado de la función local *UpdateStatus()* presentando los resultados del inicio del aplicativo al administrador.

```
void CCallManagerDlg::SetupStart()
{
	AddText(" ******************** INICIO DE LA APLICACIÓN
******************** ");
	// Crear la nueva instancia del manejador del TAPI.
	objTapi.SetParentDlg(this);
	// Inicializar el objeto TAPI.
	AddText(" INFO: INICIALIZANDO TAPI...");
	HRESULT hr = objTapi.fn_InitializeTapi();
	if (hr == S_OK)
	{
		AddText(" INFO: TAPI inicializado exitosamente");
		// Cargar y abrir las líneas que están asociadas con la
central Avaya
		AddText(" INFO: Abriendo líneas AVAYA");
		hr = objTapi.fn_OpenLines();
		if (hr == S_OK)
		{
			CString totalLines;
			totalLines.Format("%i",objTapi.m_TotalOpenedL);
			AddText(" INFO: " + totalLines + " Líneas cargadas
exitosamente");
			// Asignar referencia al protocolo
			m_ObjProtocol.SetTapiHandler(&objTapi);
			// Inicializar los parámetros de la comunicación.
			m_Listener.SetParentDlg(this);
			m_Connected.SetParentDlg(this);
			if (m_PortNumber != 0)
				SetupSocketCommunication() ;
			m_bGeneralApp=true;
			UpdateStatus();
		}
		else
		{
			AddText("ERROR: Problemas en la carga de las líneas");
			m_iTotalError++;
		}
	}
	else
	{
		AddText("ERROR: Problemas en el inicio del TAPI");
		m_iTotalError++;
	}
}
```

Figura 3.6 Código de Arranque del TAPI y las Comunicaciones en el Servidor CTI

En un mayor detalle, la primera función que tiene que ser llamada para inicializar el TAPI es *lineInitializeEx* (función propia de TAPI) cuya invocación se produce en la función *fn_InitializeTapi* del objeto *CTapiHandler* (objeto *objTapi* de la figura anterior). Es aquí donde se especifican parámetros importantes como la versión del API a trabajar (*dwAPIVersion*), el identificador de la instancia de la aplicación o la función de retorno (*LineHandlerCallback*) que será llamada cada vez que sea recibido un evento relacionado con las líneas telefónicas.

```
HRESULT tr = ::lineInitializeEx(    &hLineApp,
                                    hInst,
                                    LineHandlerCallback,
                                    pszAppName,
                                    &m_NumDevs,
                                    &dwAPIVersion,
                                    &params);
```

Figura 3.7 Función de Inicialización del TAPI

La generación de un evento detectado por el TAPI desencadena la invocación de la función *LineHandlerCallback* que se encarga de la búsqueda de la línea telefónica causante (*CLineHandler*) y se llamada a la función *OnEvent* para determinar el tipo de evento (llamada, respuesta, estado, etc.) y posteriormente crear la cadena que se envía al servidor principal según lo determinado en el protocolo.

```
static void CALLBACK LineHandlerCallback(
                                          DWORD   dwDevice,
                                          DWORD   nMsg,
                                          DWORD   dwInstance,
                                          DWORD   dwParam1,
                                          DWORD   dwParam2,
                                          DWORD   dwParam3)
{
      CLineHandler* pLineH = (CLineHandler*) dwInstance;
      if (pLineH)
            pLineH->OnEvent(dwDevice, nMsg, dwParam1, dwParam2, wParam3);
}
```

Figura 3.8 Función de Retorno en TapiHandler para el Procesamiento de Eventos

En lo que respecta a la interfaz gráfica, conforme a lo definido en el punto 2.4.2.6 (sección EAD-CTI del Proceso Unificado) el resultado final es el mostrado en la figura

3.9. Los botones “Iniciar” y “Detener” tienen como propósito arrancar e interrumpir los servicios prestados de forma manual en caso de ser necesario. Cuando la aplicación inicializa apropiadamente el TAPI, de manera inmediata se presenta en pantalla cada una de las líneas abiertas detectadas que pueden ser utilizadas en el aplicativo. La creación de una nueva extensión en la central telefónica requerirá del reinicio manual de los servicios de este módulo para poder ser utilizada en el sistema. Además, se han tomado medidas para que el cuadro de diálogo quede minimizado en el área de notificaciones con el propósito de evitar el cierre del mismo por error.

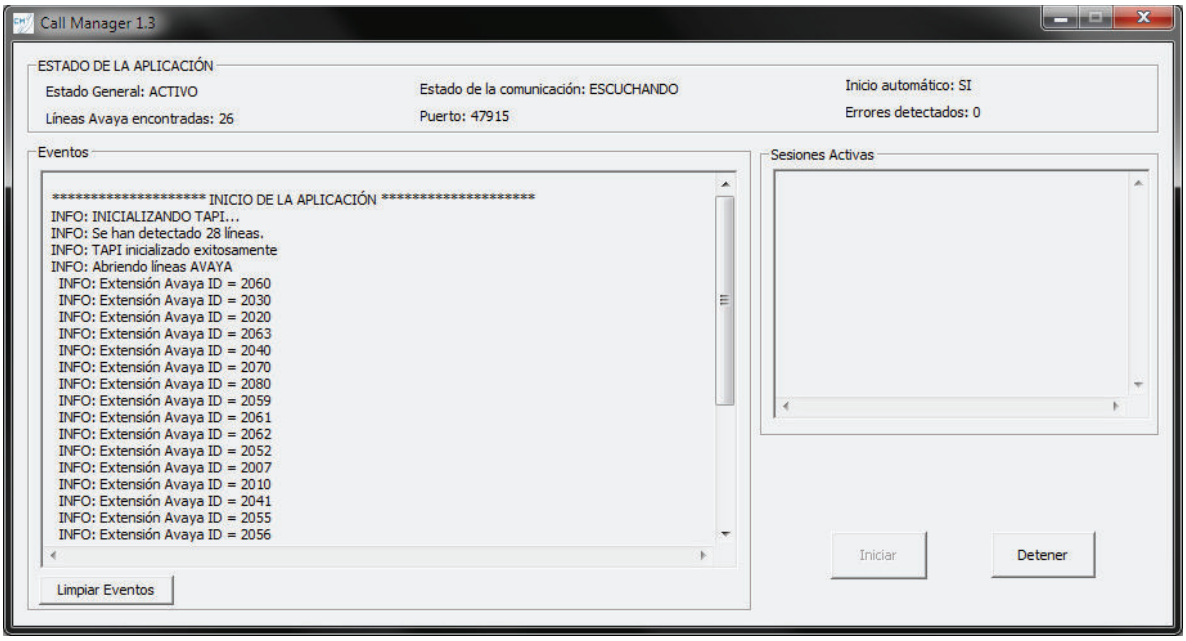


Figura 3.9 Cuadro de Diálogo Principal del Servidor CTI

El resultado final de la creación de este componente es un archivo ejecutable llamado ***CallManager.exe*** que utiliza un archivo de configuración de nombre *CallManager.conf* en el que se almacenan variables como el número de puerto de escucha o si es necesario arrancar los servicios de forma automática al iniciar la aplicación. La clase encargada de recuperar estos campos es *CIOHandler* que es llamada en el arranque de la aplicación. Estas medidas han sido tomadas con el propósito de mantener los servicios en ejecución durante el mayor tiempo posible sin intervención alguna por parte del administrador. En este sentido, al tratarse de una aplicación de tipo MFC ha sido conveniente incluir un enlace de acceso directo

(*CallManager.lnk*) en el directorio de arranque de Windows que deberá ser creado al instalar el aplicativo para que pueda ser iniciado tan pronto como arranque el sistema operativo.

3.1.2.2. Biblioteca Común (CM-BC)

En lo referente a la construcción del sistema multiplataforma (véase análisis IA-AR-4) el primer punto a tomar en cuenta tiene que ver con la creación de una biblioteca que reúna las clases y enumeraciones comunes utilizadas tanto por el cliente de escritorio como por el servidor principal con el propósito de evitar redundancias en su programación. Como resultado, ha sido creada la biblioteca *CallManagerCommons*, presentada en el diagrama de clases subsiguiente y cuyos elementos más importantes están descritos a continuación:

- **AttendedCall.** Esta clase será utilizada para crear un historial de las llamadas más recientes procesadas por el agente del *call center*.
- **CallNotification.** Considerada una de las clases más importantes, encapsula la información proporcionada por la central telefónica a través del aplicativo CTI con respecto a una llamada telefónica, como el nombre de la línea, número de extensión, estado de la llamada, números de origen y destino, entre otros.
- **ClientDataDetails.** Clase utilizada para encapsular la información de los contactos desde las bases de datos externas a la plataforma.
- **Protocol.** Es una clase de crucial importancia, cuyo propósito es el de crear todas y cada una de las cadenas de mensajes que serán enviadas o recibidas por la red de datos.
- **Class, Commnand, Notification.** Enumeraciones que agrupan las opciones que conforman los mensajes a ser transmitidos entre los componentes del sistema. El formato de dichos mensajes está descrito en el apartado *Comunicaciones* (sección 2.4.2.1, EAD-COM) de este documento.
- **EncryptionHandler.** Con el propósito de proporcionar un básico nivel de seguridad en las comunicaciones (puesto que existe la posibilidad de

transmitir información valiosa de los contactos de las instituciones); se ha considerado necesaria la ofuscación de dicha información mediante el uso del esquema de cifrado AES (*Advanced Encryption Standard*) incluido en las bibliotecas de Java. La seguridad es un requisito opcional secundario planteado en la sección 2.3.3.3 (IR-NF).

- **Enumeraciones.** Todos los elementos titulados en mayúsculas, como APPUSER, CLASS, NOTIFICATION, COMMAND, etc., corresponden a las enumeraciones que contienen información estática utilizada por ambos módulos.

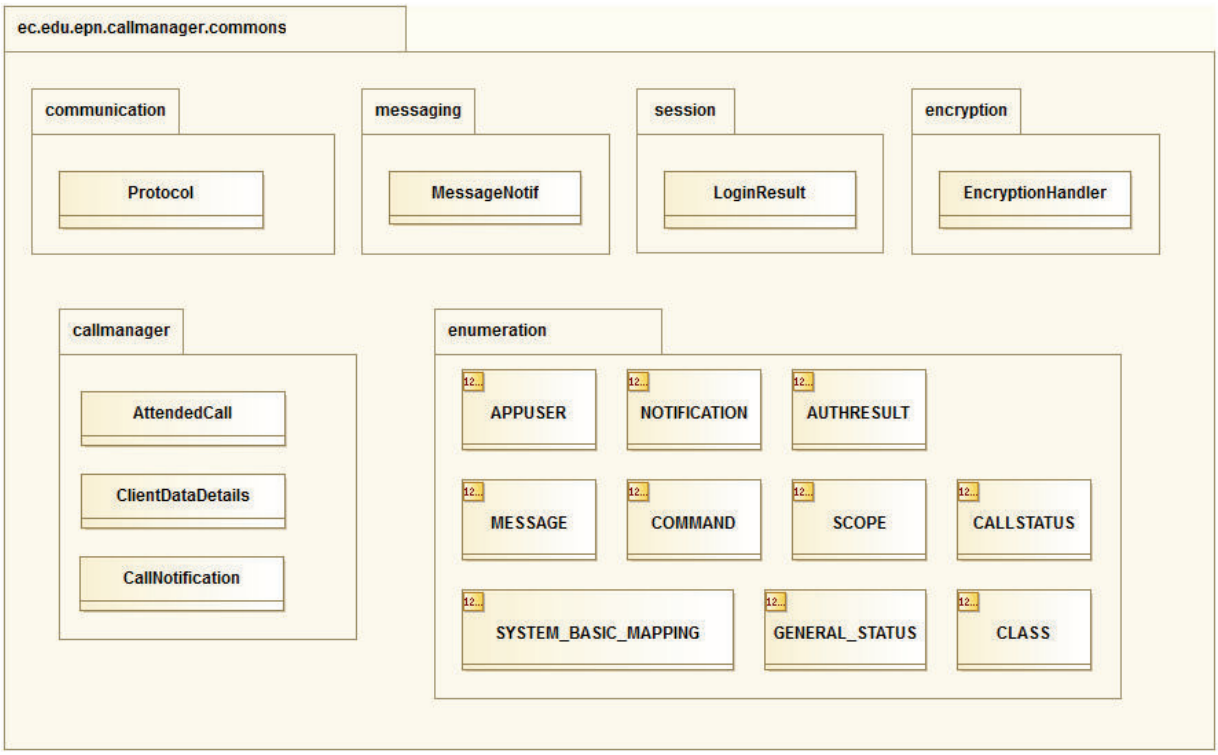


Figura 3.10 Diagrama de Clases de la Biblioteca *CallManagerCommons*

El resultado de la creación de esta biblioteca es un archivo Java de nombre ***CallManagerCommons.jar*** que será importado y enlazado en el desarrollo del módulo principal y del agente de escritorio.

3.1.2.3. Servidor Principal (CM-CORE)

La implementación del componente de mayor complejidad de la solución comenzó con la creación de un proyecto de tipo Aplicación Empresarial (*Enterprise Application*) de Java EE en *NetBeans* que tiene por nombre CMW1 (*Call Manager Web 1*). El hacerlo crea dos módulos principales, ***CMW1-ejb***, que es donde se procesa la lógica de negocio de la aplicación y ***CMW1-war***, que es donde se crean las páginas de web de administración de la solución.

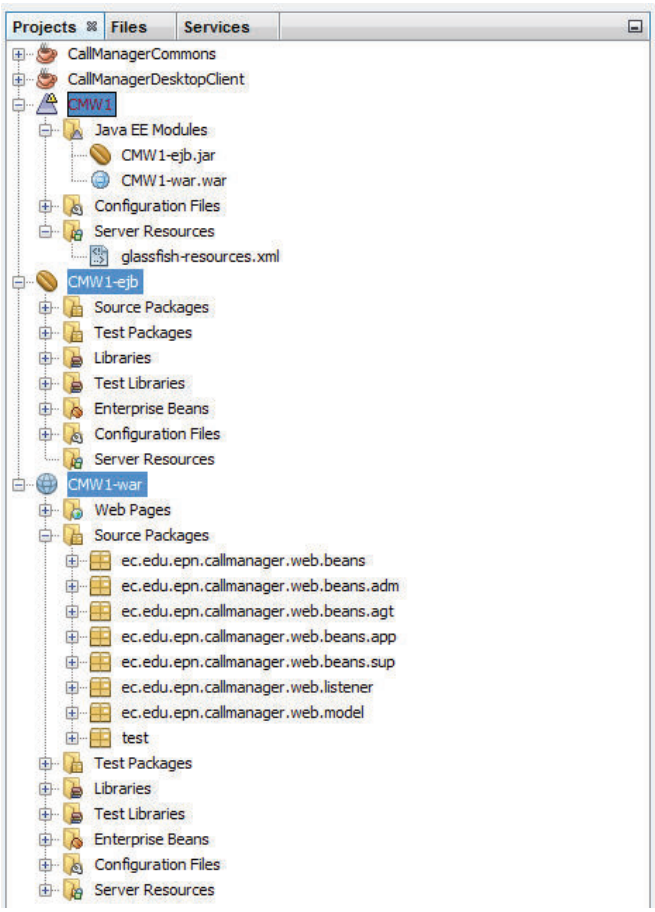


Figura 3.11 Gestión del Proyectos CMW1 en *NetBeans*

El conjunto de paquetes está denominado en función de los módulos del proyecto, que en este caso serán *ec.edu.epn.callmanager.ejb* para los componentes web, mientras que los elementos bajo *ec.edu.epn.callmanager.web* corresponden al módulo de *front-end* de la solución.

3.1.2.3.1. Componente de Negocio (CM-CORE-Bss)

Si bien en la sección 2.4.2.7 (EAD-CORE-Bss) de este documento ya ha sido descrito el funcionamiento de los elementos primarios que conforman la capa de negocio del servidor principal, es aquí donde es posible apreciar en su totalidad la importancia del EJB de tipo *singleton* llamado *MainController*, el cual es el encargado de ser el punto central de gestión de todos los eventos, solicitudes y consultas entre los componentes internos del servidor principal. La figura 3.12 muestra que *MainController,* ubicado en el paquete *main,* es referenciado e interconectado entre otros menos relevantes, con los siguientes elementos:

- *ServerComm.* Clase que hereda de *DefaultComm*, es la encargada de mantener la comunicación con el servidor CTI. El objeto de esta clase puede ser instanciado manualmente por petición del administrador del sistema o de manera automática en el arranque del servidor de aplicaciones.
- *DeskServerHandler*. La instancia de esta clase es ejecutada en otro hilo de ejecución con el propósito de administrar y controlar las comunicaciones establecidas con los agentes que han iniciado sesión en el sistema. En otras palabras, la adición y remoción de los sockets relacionados con los agentes conectados es realizada en esta clase.
- *UserMain*. Las sesiones establecidas de los usuarios son gestionadas desde un *hashmap* denominado *sessionMap* que contiene elementos de tipo *UserMain* que están mapeados con las líneas telefónicas subyacentes. Esto quiere decir que el identificador que permite relacionar usuarios (UserMain) con el socket de comunicación de los agentes (DeskComm) es la línea telefónica.
- *EventController*. Cada evento generado en el sistema pasa por este EJB que es llamado en la función *processMessage* de *MainController*. El detalle del funcionamiento de este componente será descrito más adelante.
- *SchedulerController*. Monitoreo de mensajes de tipo *keepalive*.
- *SystemParamsFacade*. Recuperación de parámetros desde la BDD.

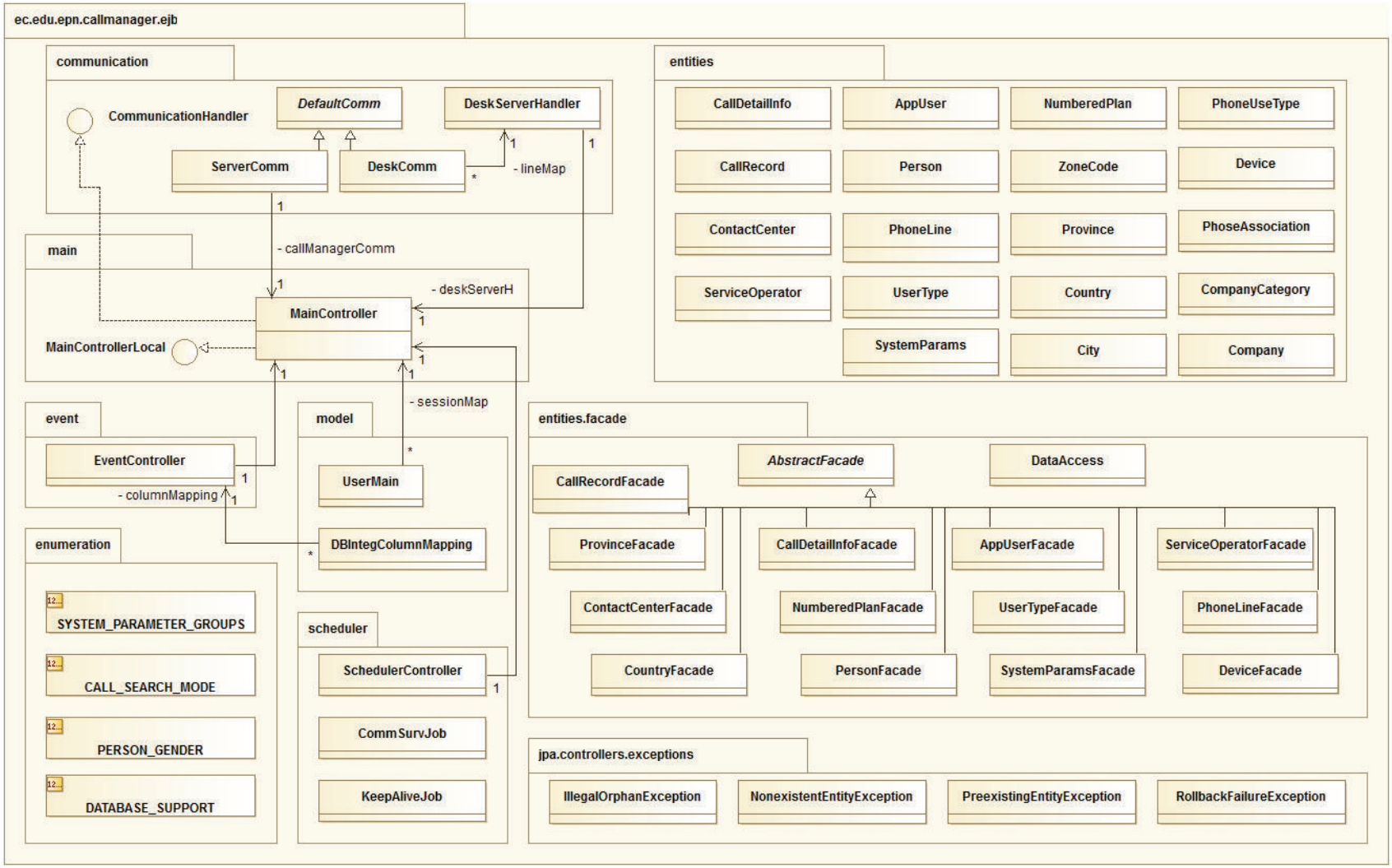


Figura 3.12 Diagrama de Clases del Componente de Negocio del Servidor Principal

Por otro lado, debido a que los módulos principal y CTI pueden operar en servidores separados, han sido tomadas en cuenta medidas de tolerancia a fallos en caso de sufrirse pérdidas de conectividad (como por ejemplo, el reinicio de algún servidor o la desconexión del equipo de la red). Para ello se ha considerado la reconexión automática entre ellos, mediante la transmisión periódica de mensajes de tipo *keepalive* que comprueben el estado operativo del enlace entre los componentes, siendo favorable utilizar un nuevo EJB de tipo *singleton* llamado *SchedulerController* ubicado dentro del paquete *scheduler* y que será inicializado por *MainController* en el arranque del sistema. Este bajo nivel de tolerancia a fallos (de comunicaciones) está previsto según los requisitos secundarios IR-RF-P5 (Tolerancia a fallos) e IR-RF-P6 (Estable) descritos en la fase de inicio.

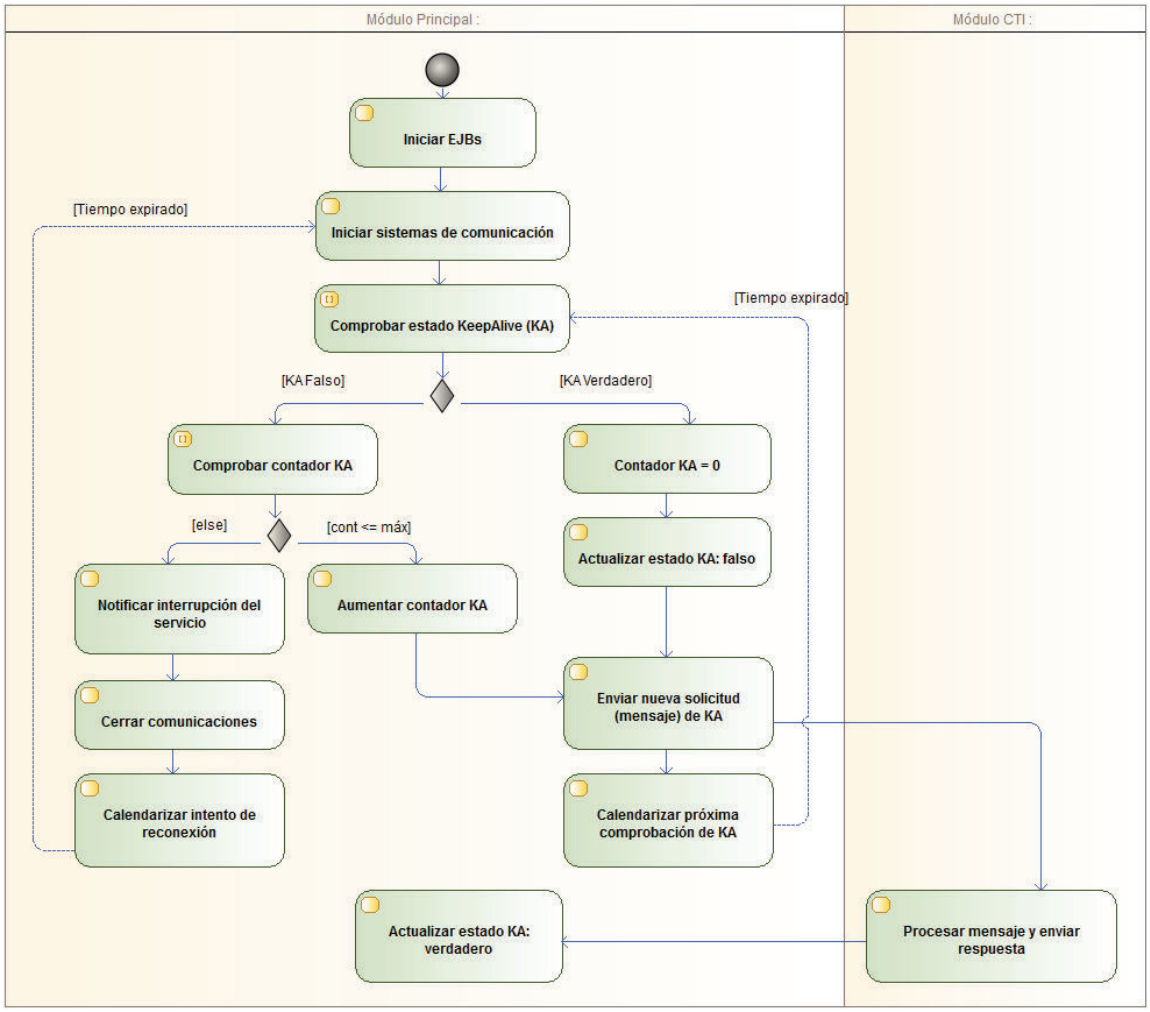


Figura 3.13 Procedimiento de Reconexión Automática del Servidor Principal.

Como puede apreciarse en el diagrama de actividades anterior, para que sea posible mantener una reconexión periódica entre los componentes principal y CTI fue necesario utilizar un parámetro booleano (*keepalive*) que represente el estado de la comunicación. De esta manera, cuando *keepalive* es verdadero, significa que el módulo CTI está activo y enviando respuestas de forma regular según lo esperado. Cuando *keepalive* es falso, en cambio, se incrementa un contador que almacena el número de ocasiones que se ha intentado "sondear la comunicación" hasta un valor máximo que permita concluir que la conexión ha sido perdida e inmediatamente se inicie un procedimiento de notificación a los usuarios del sistema.

En lo que respecta a la implementación de la búsqueda de información en el transcurso de una llamada, según lo especificado en las secciones 2.4.2.4 (EAD-INF) y 2.4.2.7.1 (EAD-CORE-Bss) de la fase de elaboración, las llamadas telefónicas gestionadas en el servidor CTI son transformadas en mensajes que se reciben en el servidor principal en la función *processMessage* de *MainController*, como es posible visualizar en el diagrama de secuencias 3.14. Éste a su vez transfiere los mensajes al método *processEventMessage* del EJB *EventController* para su procesamiento, donde la cadena de texto perteneciente al mensaje es seccionada en partes y categorizada de acuerdo a lo definido en el protocolo.

Cuando el mensaje corresponde a una llamada telefónica es reconocido como tal, a continuación se llama a la función *processCallNotification* que tiene por función recuperar y almacenar los atributos específicos de la llamada, como el número telefónico de origen, número de destino, extensión asociada, fecha y hora de la llamada, entre otros. Seguidamente se realiza la búsqueda de la información de contacto, planes de numeración y datos de georreferenciación mediante la invocación de la función *callInfoDetailsProcessing*. En el caso de haberse encontrado coordenadas geográficas que localicen la ubicación del número telefónico, éstas serán reenviadas inmediatamente al agente de escritorio junto con la información del contacto. De no ser así, se realizará un último intento de localizar la llamada a través de la función *defaultLocationFinder* que en este caso utiliza la tecnología descrita en el apartado 3.1.1.3 de este capítulo (CA-DES).

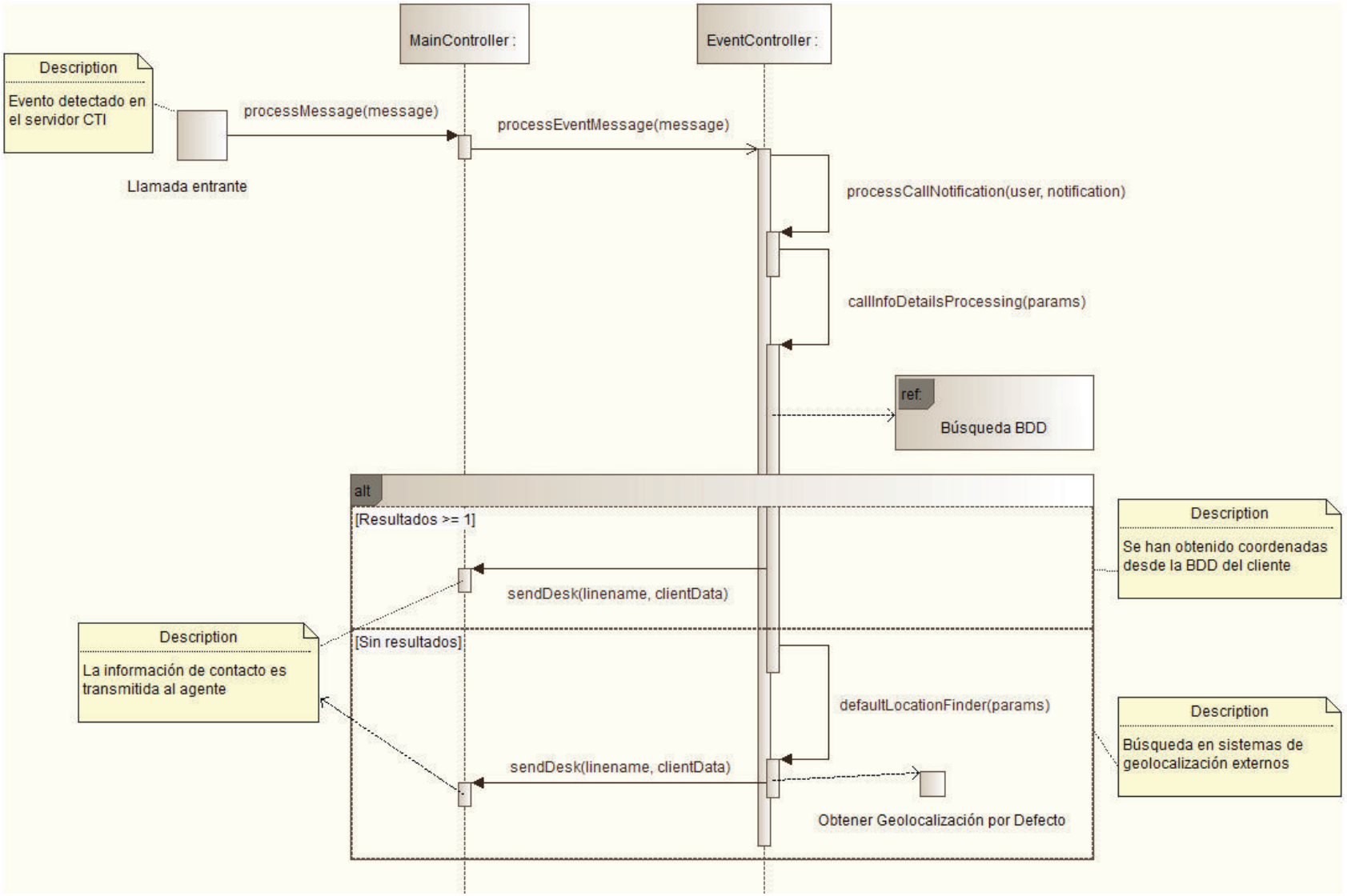


Figura 3.14 Procesamiento de Llamadas y Búsqueda de Información de Contacto en el Servidor Principal

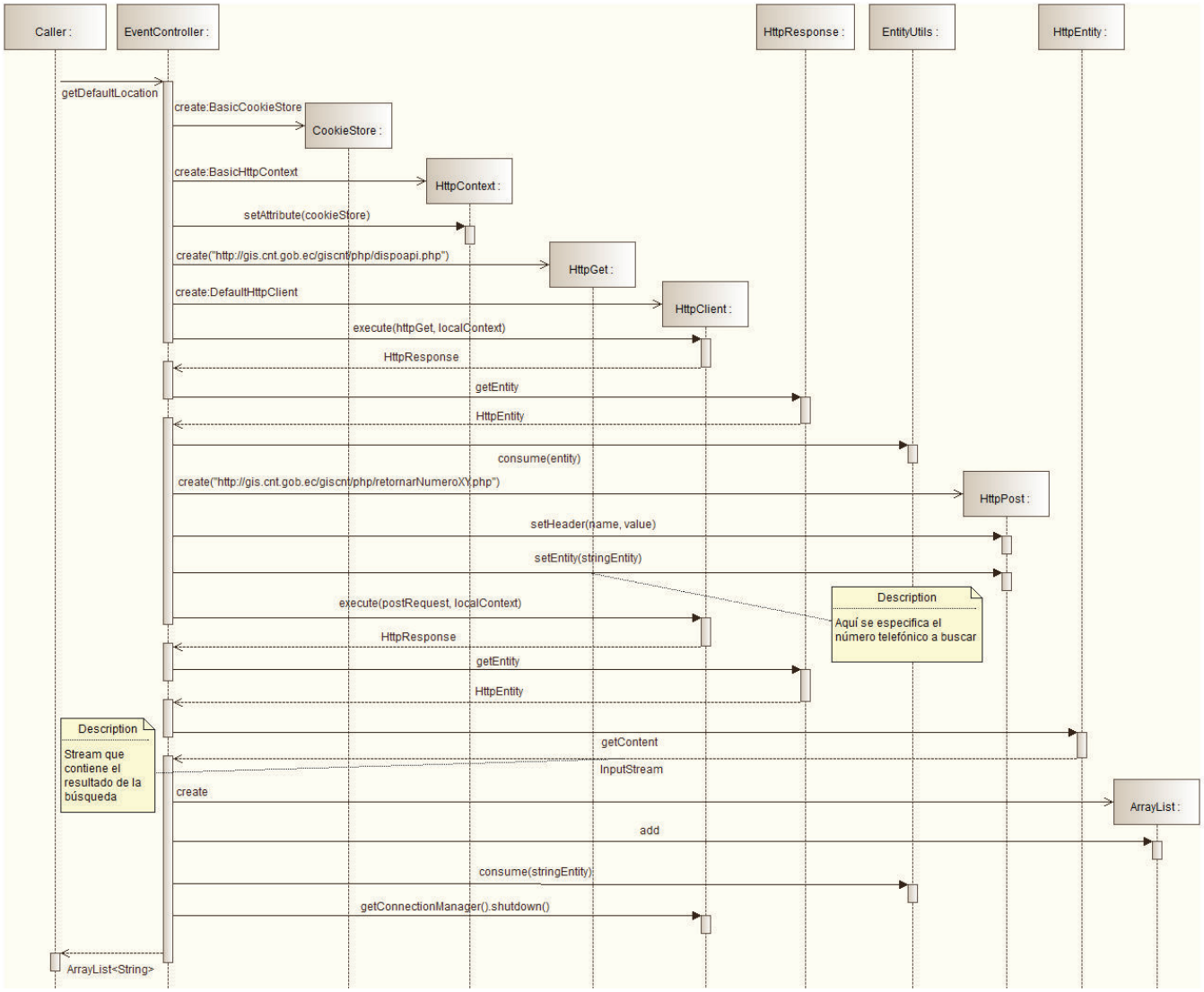


Figura 3.15 Utilización de la Biblioteca *Apache HttpComponents* para la Georreferenciación de Llamadas Telefónicas Utilizando el Servicio de Disponibilidad Geográfica de CNT

El proceso de búsqueda de información geográfica de las llamadas telefónicas en el servidor principal está esquematizado en un mayor detalle en la figura 3.15. Allí es posible apreciar el uso de la biblioteca *Apache HttpComponents* desde el EJB *EventController*. Cabe recalcar que dicho escenario únicamente ocurre cuando el aplicativo no ha sido capaz de encontrar coordenadas de geolocalización en las bases de datos externas del sistema, utilizando para ello el servicio de disponibilidad geográfica provisto por la CNT (*http://gis.cnt.com.ec/giscnt/php/dispoapi.php*).

Conforme a lo planteado en la sección 3.1.1.3 (CA-DES), el primer paso consiste en la creación de una *cookie* para posteriormente realizar una llamada al método *HttpGet* para cargar la página principal. Con el establecimiento del cliente http (*HttpClient* definido en la respuesta de *HttpGet*) el siguiente paso tiene por objeto realizar una solicitud de tipo POST a la dirección que finalmente provee la información geográfica (*http://gis.cnt.com.ec/giscnt/php/retornarNumeroXY.php*). Los parámetros de búsqueda como el número telefónico y la distancia de cercanía son establecidos en un objeto de tipo *StringEntity* que se añade a la instancia de la clase *HttpPost*. La nueva respuesta *HttpResponse* es examinada a detalle con el objeto de recuperar la información de localización requerida que luego es añadida a un *array* que se devuelve en el retorno de la función previa la liberación de los recursos utilizados. Debe notarse que el servidor principal se encarga únicamente de la búsqueda de coordenadas geográficas ya que la presentación de las mismas en un mapa es responsabilidad del cliente de escritorio.

En lo concerniente a la implementación de la búsqueda de información adicional de los contactos en bases de datos externas, el diagrama de secuencias 3.16 detalla más hondamente lo analizado en la sección 2.4.2.9 (EAD-BDD). Lo primero que debe notarse es que en el EJB *EventController,* específicamente en el método *callInfoDetailsProcessing* las interfaces *DataSource, Connection* y *PreparedStatement* de los paquetes *javax.sql* y *java.sql* son utilizadas para conectarse al repositorio de información externo perteneciente a las organizaciones interesadas. Dicha base de datos debe haber sido previamente configurada por los administradores para que sea posible acceder y leer una tabla o vista que contenga

la información de los contactos conforme a lo dispuesto en la sección 2.4.2.9 (EAD-BDD). Cabe notar que para el establecimiento de la conexión se utiliza una variable de tipo cadena llamada *cMAppDBBDSLogicalName* (ver el diagrama) que contiene el identificador del recurso del *pool* de conexiones JDBC que debe ser creado manualmente en el servidor de aplicaciones *GlassFish* (que en este proyecto se llama "*jdbc/ClientDatabaseJDBC*").

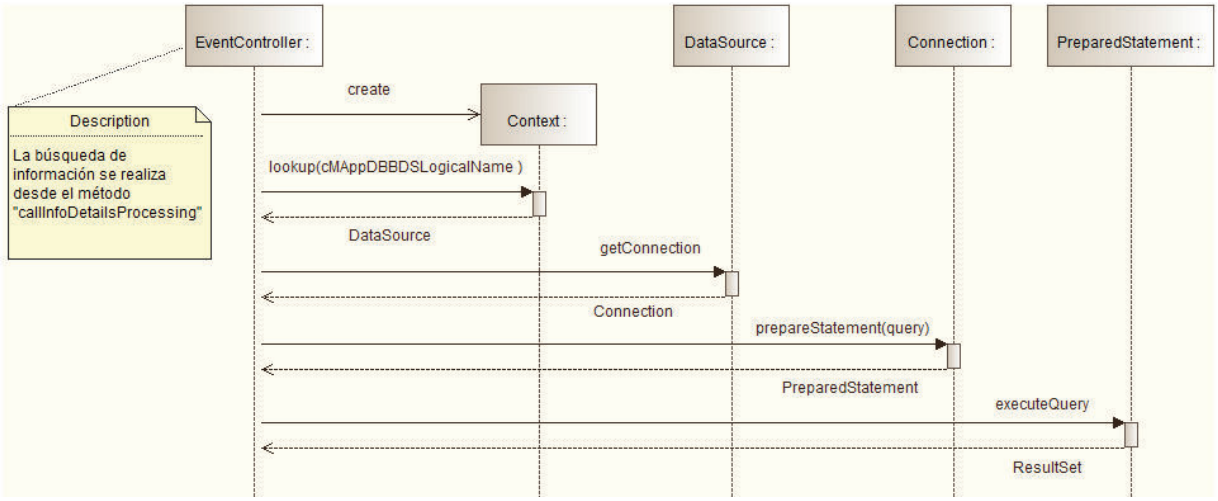


Figura 3.16 Búsqueda de Información en Bases de Datos Externas

Además, la cadena de consulta (*query*) utilizada por la función *prepareStatement* de la instancia de la clase *Connection* es distinta para cada tipo de base de datos soportada, la cual es creada en la inicialización de *EventController* según los parámetros de interconexión configurados por el administrador.

```
switch (dbSupport) {
    case POSTGRESQL:
        dbClientInfoQuery += " FROM " + cMAppDBBSchema + "." + cMAppDBBTable + " WHERE " + phoneColumnName + " = ?";
        dbClientInfoByIdQuery += " FROM " + cMAppDBBSchema + "." + cMAppDBBTable + " WHERE " + idColumnName + " = ?";
        break;
    case MYSQL:
        dbClientInfoQuery += " FROM " + cMAppDBBCatalog + "." + cMAppDBBTable + " WHERE " + phoneColumnName + " = ?";
        dbClientInfoByIdQuery += " FROM " + cMAppDBBCatalog + "." + cMAppDBBTable + " WHERE " + idColumnName + " = ?";
        break;
    case SQL_SERVER:
        dbClientInfoQuery += " FROM " + cMAppDBBTable + " WHERE " + phoneColumnName + " = ?";
        dbClientInfoByIdQuery += " FROM " + cMAppDBBTable + " WHERE " + idColumnName + " = ?";
        break;
}
```

Figura 3.17 Creación de Cadenas de Búsqueda para las BDD Soportadas

3.1.2.3.2. Componente Web (CM-CORE-Web)

El desarrollo de este componente tiene como base lo descrito en el apartado 2.4.2.7.2 (EAD-CORE-Web), en el que se menciona que cada usuario del sistema tendrá un perfil de gestión propio en el que las páginas web subyacentes utilizarán la tecnología JSF para su generación.

A lo anterior se suma lo resuelto en el análisis 3.1.1.2 (CA-CORE) en el que se establece que se utilizará la extensión *PrimeFaces* [17] con el propósito de añadir características avanzadas y brindar una mejor presentación y experiencia a los usuarios.

El siguiente paso consiste en la presentación del diagrama de clases para el componente web (figura 3.18), en el que están graficadas las principales relaciones entre los elementos internos y su interacción con los bloques del componente de negocio *ec.edu.epn.callmanager.ejb*. El conjunto de clases que ayudan a gestionar la lógica de negocio del componente web está en el paquete ***ec.edu.epn.callmanager.web.***

Los *beans administrados* están organizados dentro del paquete *beans*, los cuales son responsables de la gestión misma del aplicativo web según el rol de cada usuario.

Las clases *AgentMonitor, MainUserBeanS* y *AppLCMBean* tienen que utilizar métodos y objetos de *MainController* para brindar información relacionada con el estado de los agentes, los usuarios y la aplicación respectivamente.

Dado que casi todas las clases dentro de este paquete realizan llamadas a las fachadas (EJBs) y *entities* del componente de negocio, se ha decido crear un único enlace que conecta los paquetes *web.beans* y *ejb.entites* con el propósito de presentar la información de una forma más limpia que facilite la interpretación del diagrama. Sin embargo, es vital aclarar que cada *bean* requiere de *entities* particulares inherentes a su función.

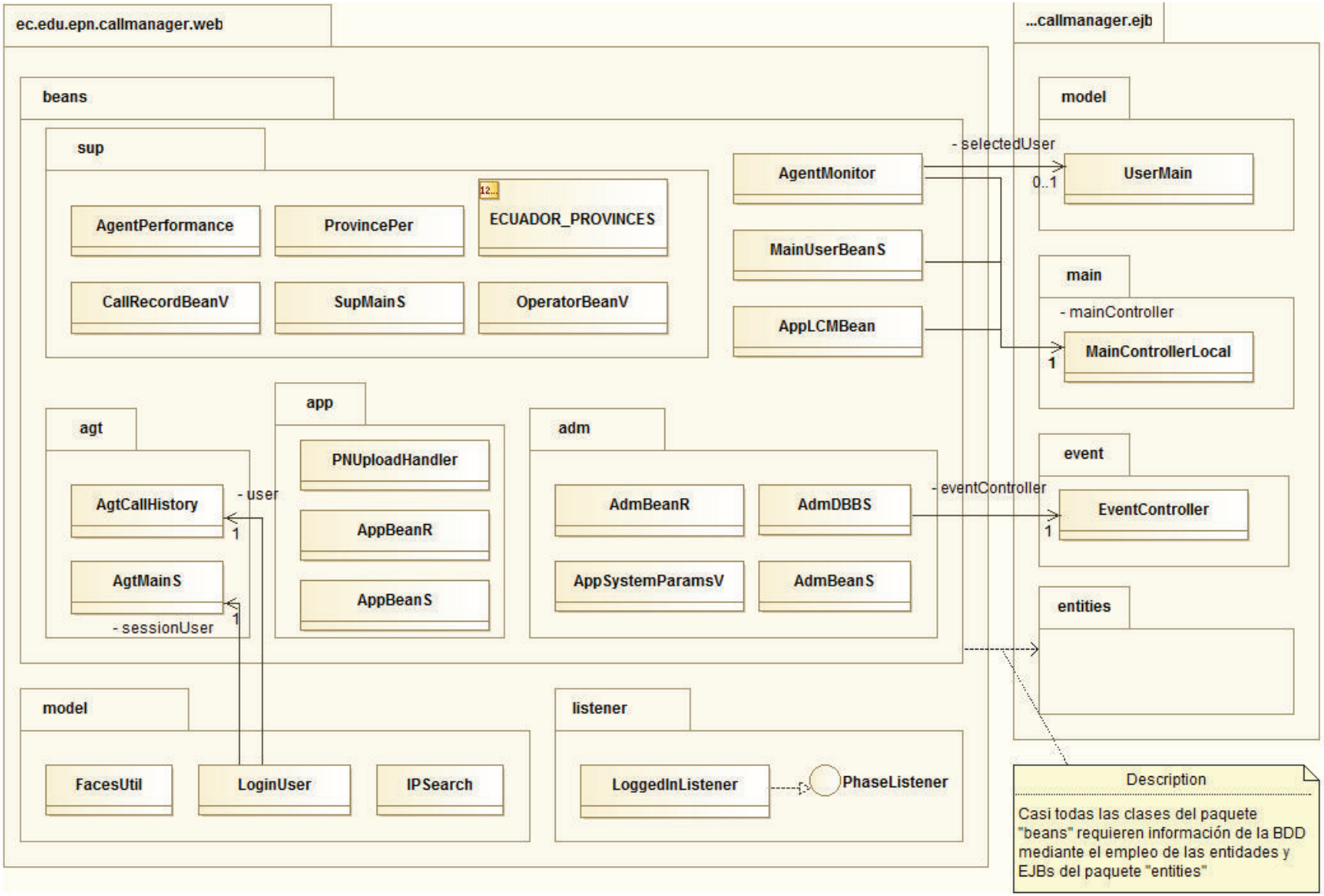


Figura 3.18 Diagrama de Clases del Componente Web del Servidor Principal

Seguidamente, utilizando como base el diseño de la interfaz gráfica y el esquema organizacional propuesto según lo especificado en el apartado 2.4.2.7.2 (EAD-CORE-Web) de la fase de elaboración, han sido creadas las páginas web que se detallarán a continuación:

1. **Pantalla de inicio de sesión.** Ubicada en el directorio raíz del aplicativo web tiene por nombre *Index.xhtml* y hace uso del *bean* de sesión *MainUserBeanS* para el ingreso y validación de credenciales en el sistema a través de *MainController*. Sin embargo, la comprobación de los perfiles de navegación adecuados es realizada en la clase *LoggedInListener*.

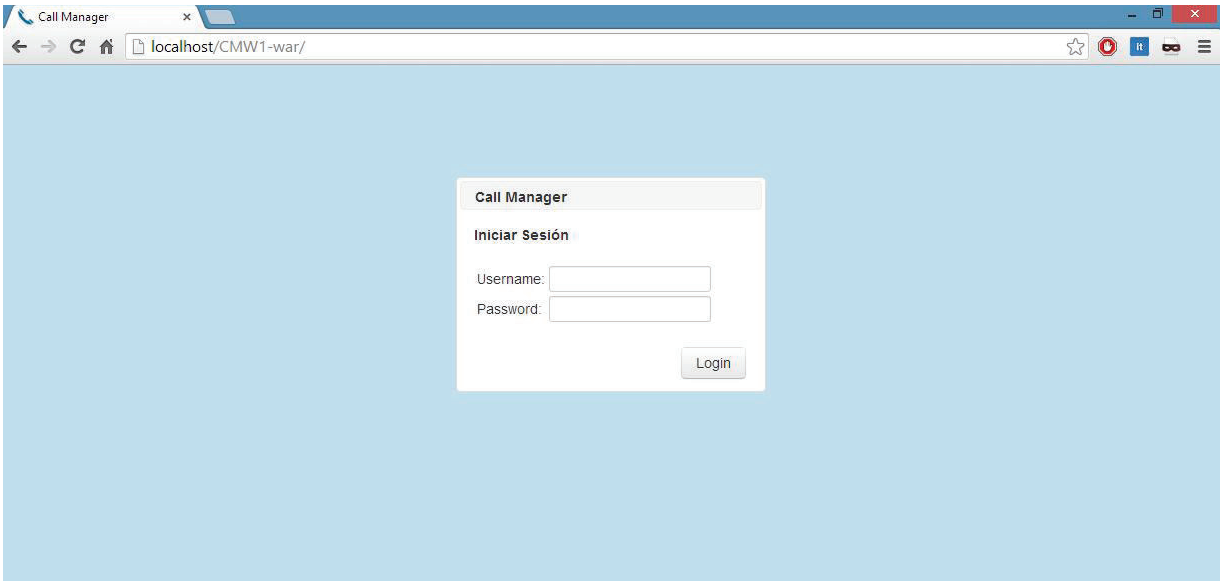


Figura 3.19 Página de Inicio de Sesión

El diagrama de secuencias de la figura 3.20 describe el proceso seguido por la aplicación para autenticar a los usuarios. Desde la pantalla de inicio de sesión se reciben las credenciales que a través del bean *MainUserBeanS* son reenviadas a *MainControllerLocal* por medio de su interfaz *authenticate*. El resultado es una enumeración de tipo AUTHRESULT, que de ser positiva (LOGGED) recuperará los detalles de usuario y lo redirigirá a la página principal de su perfil; caso contrario se notificará la equivocación en pantalla.

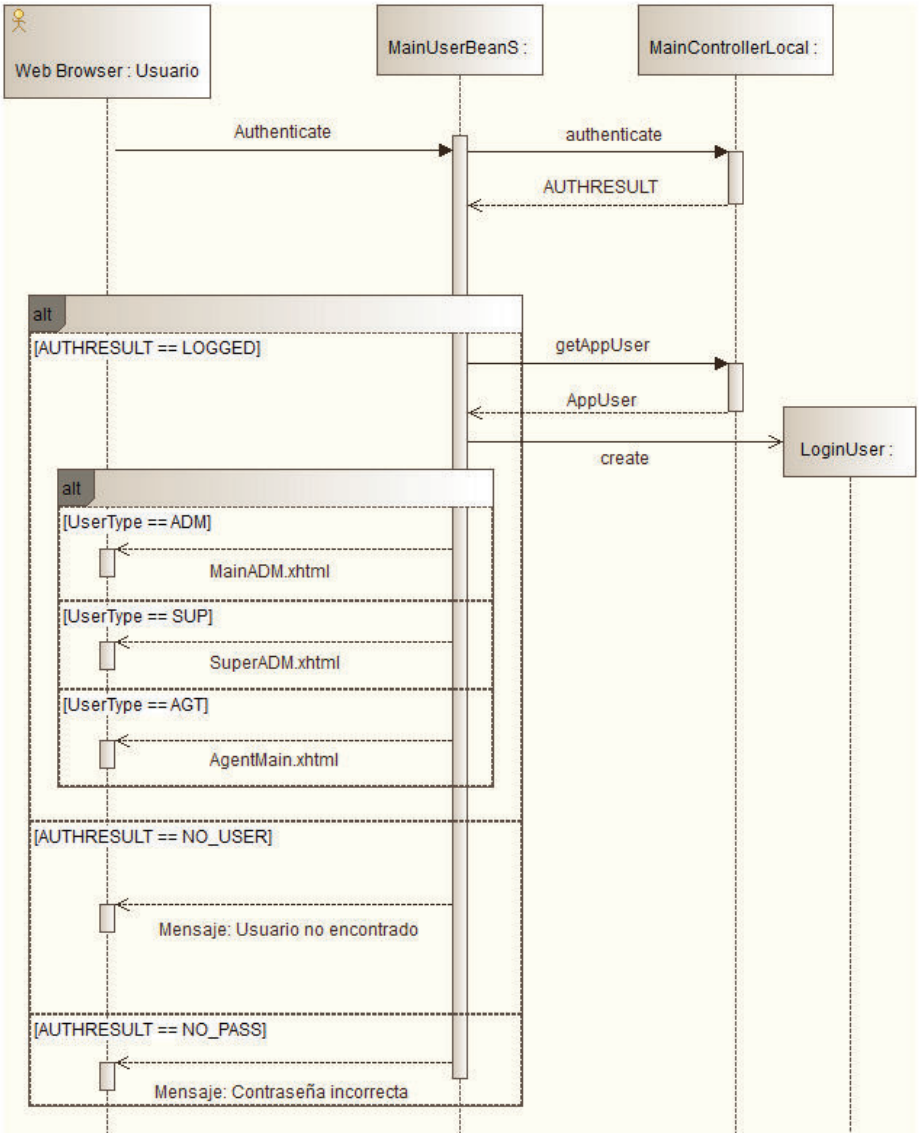


Figura 3.20 Procedimiento de Inicio de Sesión

2. **Administración.** Para la administración se utiliza la plantilla *AdminTemplate.xhtml* en la cual se incluyen atributos comunes a todas las páginas de esta sección, como la cabecera y pie de página o el menú de navegación ubicado en la parte izquierda de la pantalla.
    a. **Página principal.** La página *MainADM.xhtml* utiliza el bean de aplicación *AppLCMBean* para mostrar el estado de la comunicación del servidor principal con el servidor CTI y el número de agentes conectados al sistema en ese momento.

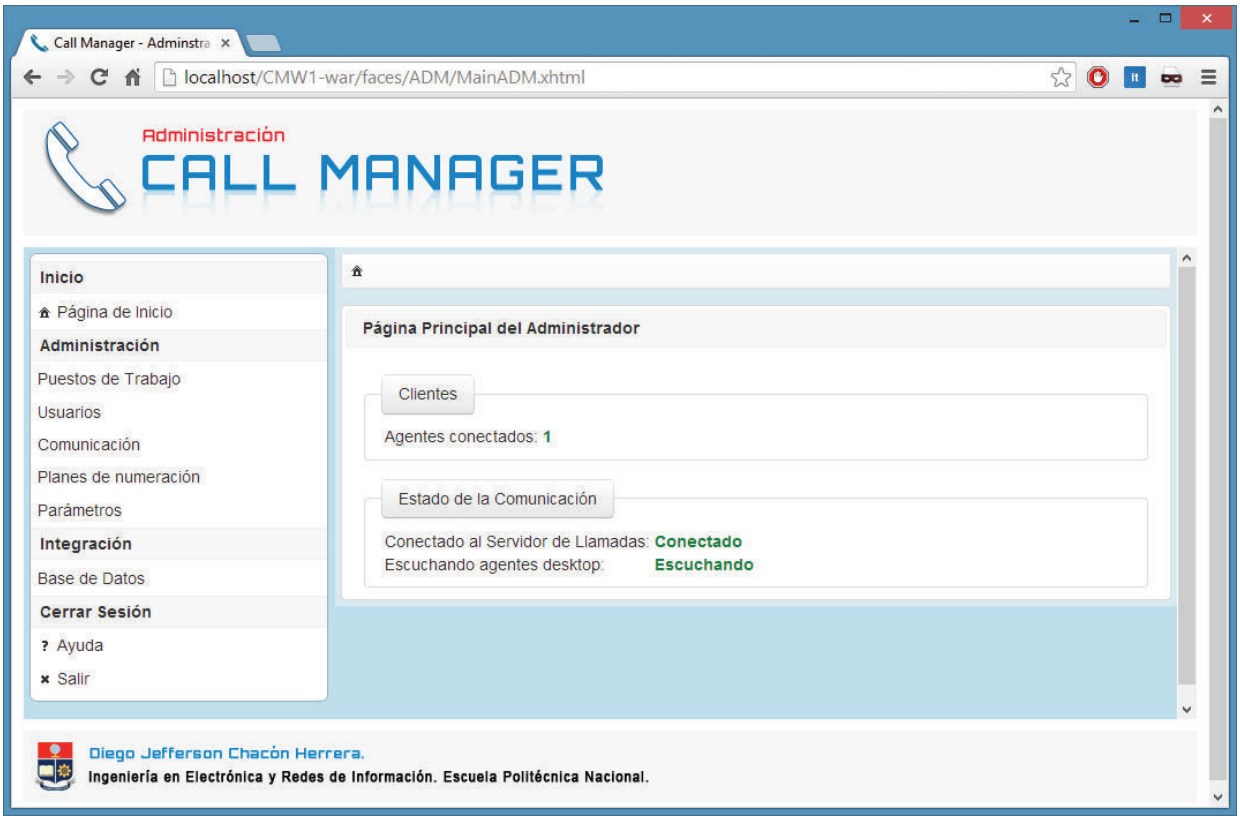


Figura 3.21 Página de Inicio del Administrador

b. **Puestos de trabajo.** La página *WorkplaceADM.xhtml* hace uso de los beans administrados *AdmBeanR* y *AdmBeanS* para la creación, modificación y eliminación de líneas telefónicas por medio de las ventanas emergentes *newExtnDlg* y *editExtnDlg*.

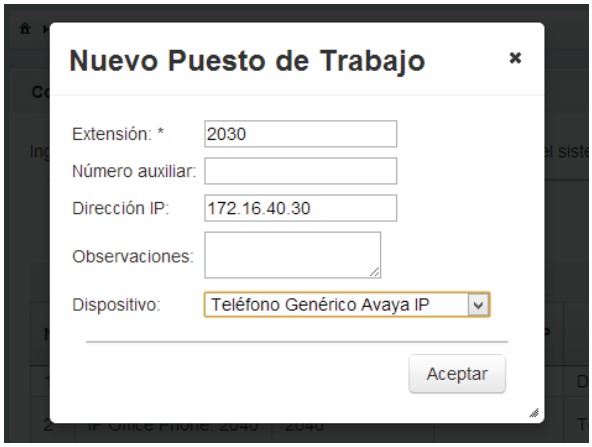


Figura 3.22 Creación de un Nuevo Puesto de Trabajo

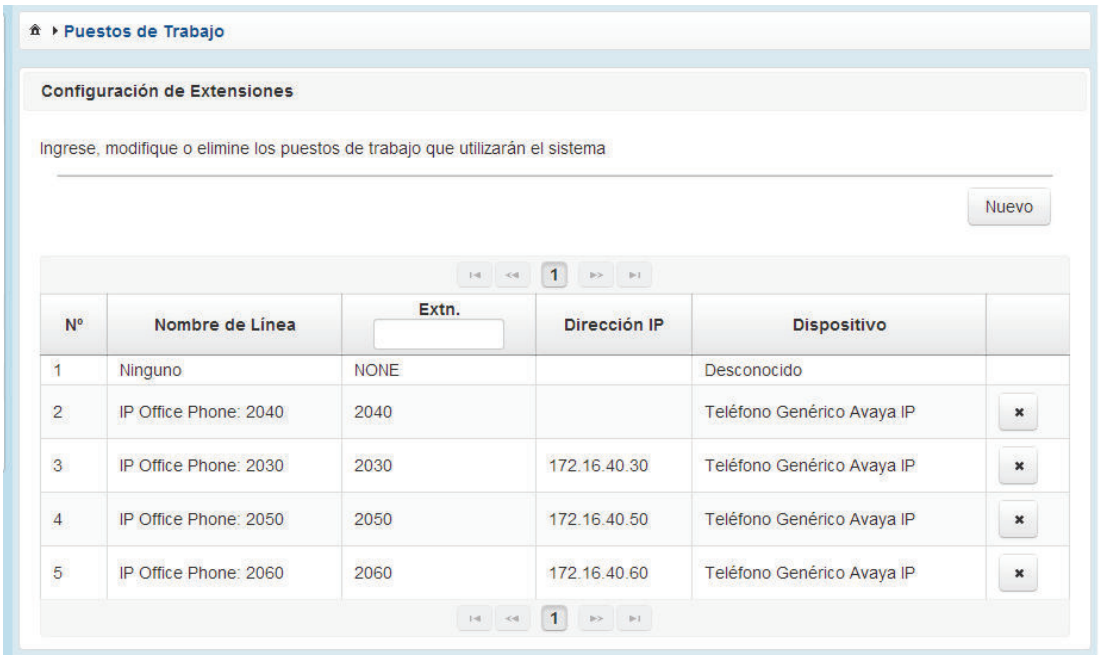


Figura 3.23 Página de Gestión de Puestos de Trabajo

c. **Usuarios.** La página *UserADM.xhtml* que utiliza los beans *AdmBeanR* y *AdmBeanS* muestra los usuarios del sistema. La creación de nuevos usuarios se realiza en la página *NewAppUserADM.xhtml*, mientras que la edición tiene lugar en *UserDetailADM.xhtml*.

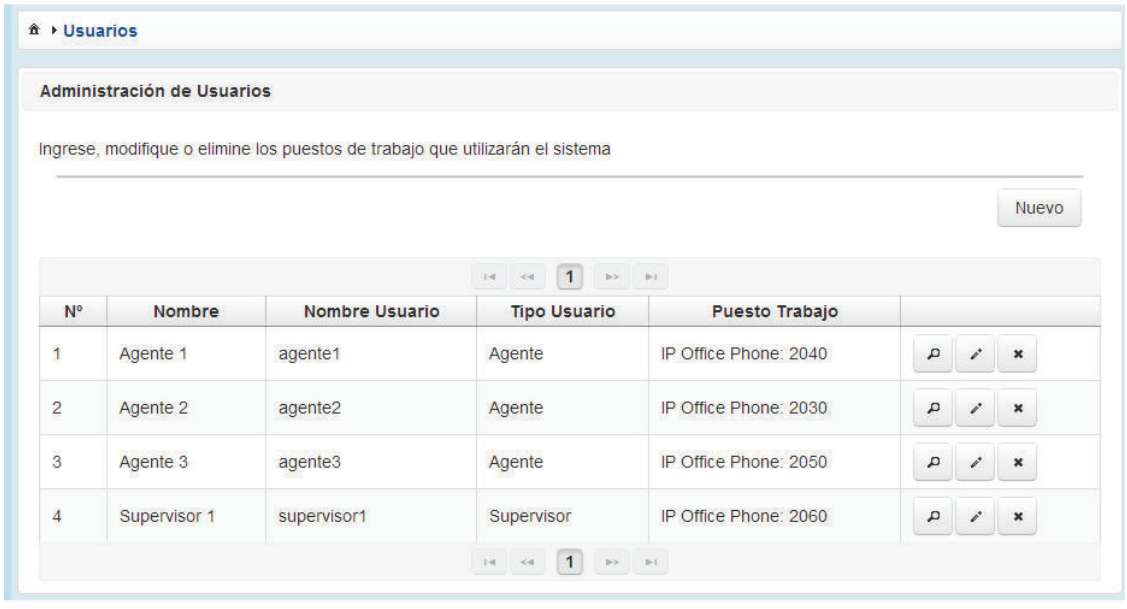


Figura 3.24 Gestión de Usuarios

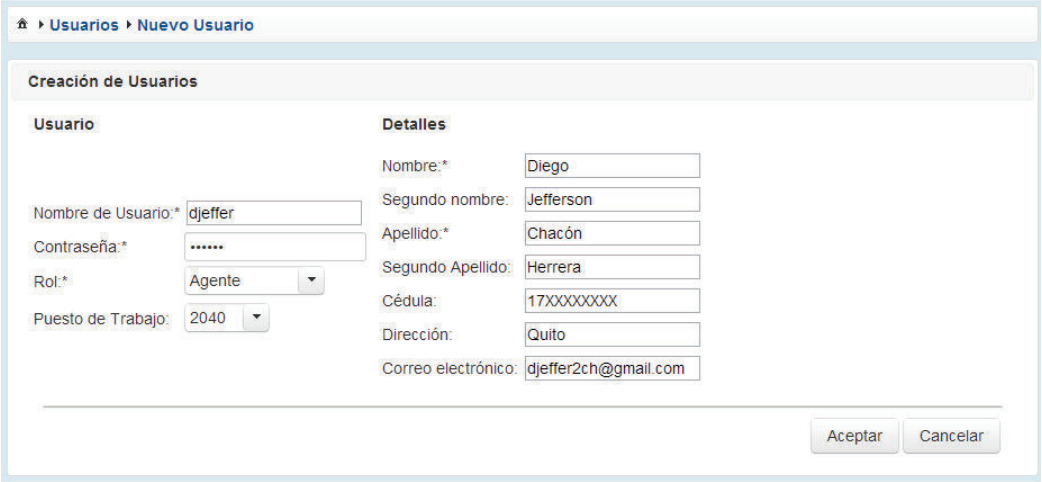


Figura 3.25 Creación de Usuarios

d. **Comunicaciones.** La configuración de los parámetros de comunicación se realiza en la página *CommADM.xhtml* mediante el *bean* de aplicación *AppLCMBean*.

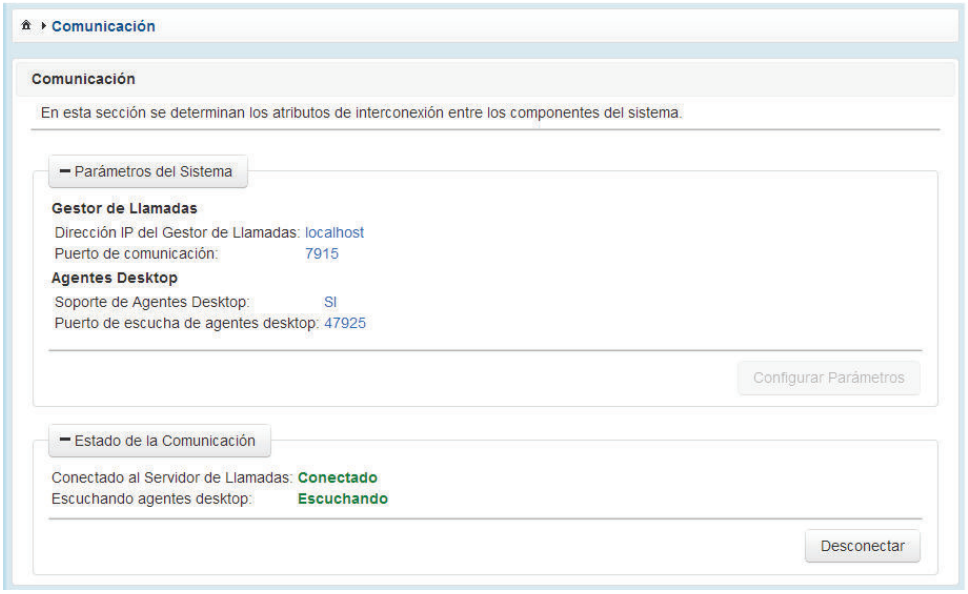


Figura 3.26 Gestión de Comunicaciones

e. **Planes de numeración.** La página *NumberedPlan.xhtml* ofrece en una tabla la visualización del estado de carga de los planes de numeración del sistema. Para la modificación particular del plan correspondiente a cada zona se utiliza la página *dNumberedPlan.xhtml*. Los *beans* utilizados en esta sección son *AppBeanS* y *AppBeanR*.

Figura 3.27 Gestión de Planes de Numeración[17]

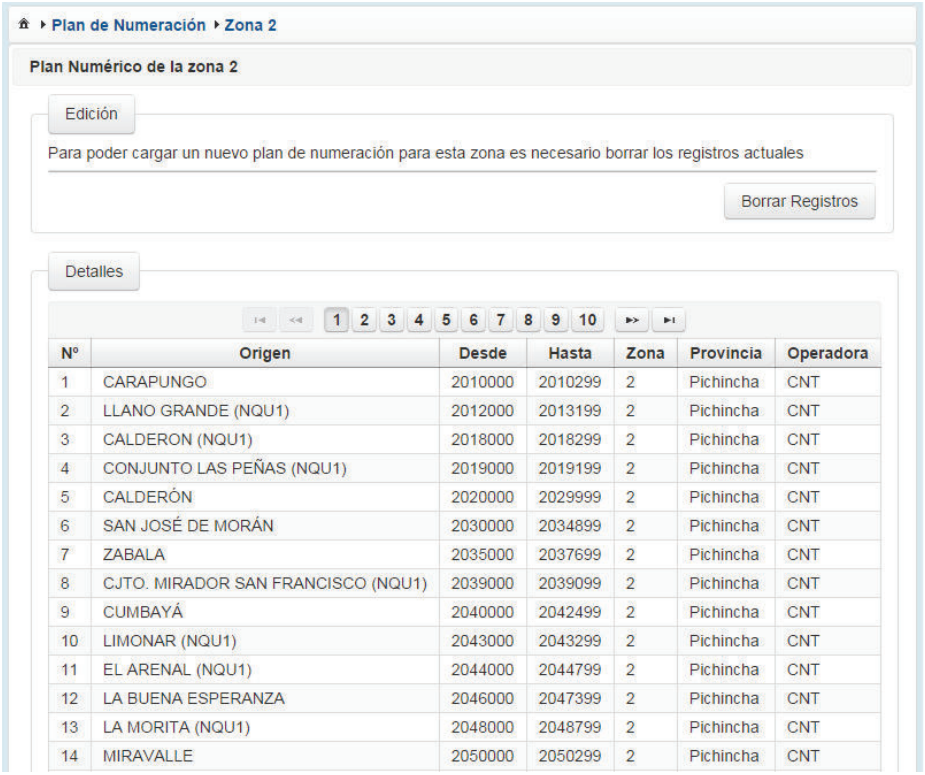


Figura 3.28 Plan de Numeración de la Zona 2 Cargado en la BDD

[17] El procedimiento de creación y carga de los planes de numeración está descrito en el **Anexo E**.

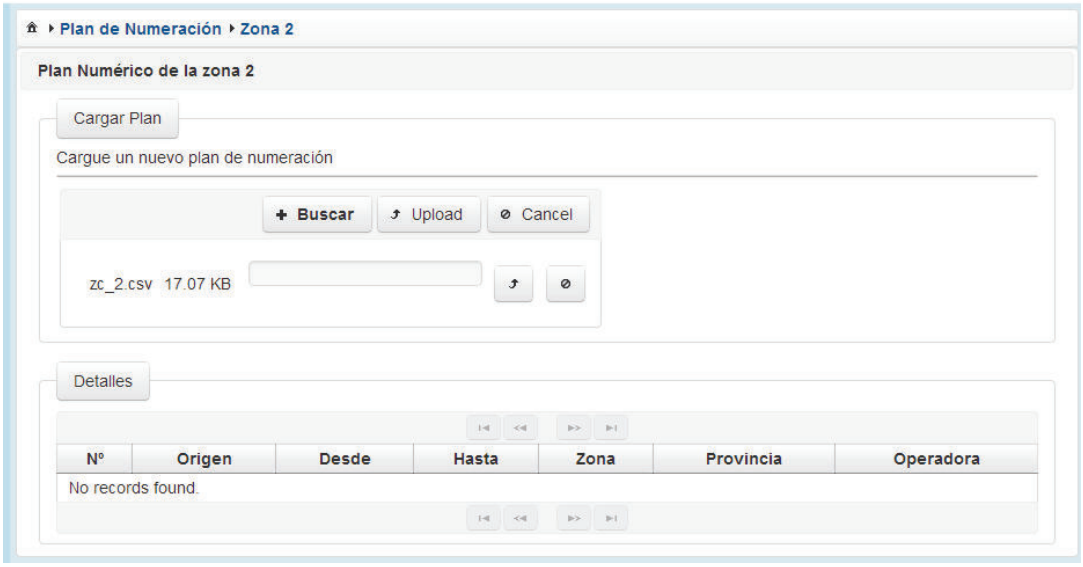


Figura 3.29 Carga de un Nuevo Plan de Numeración en el Sistema

f. **Variables del sistema.** El *bean AppSystemParamsV* es utilizado por la página *CallCenterADM.xhtml* para la gestión de variables y parámetros del sistema. Cualquier modificación que se realice en esta sección de forma obligatoria requiere reiniciar el aplicativo.

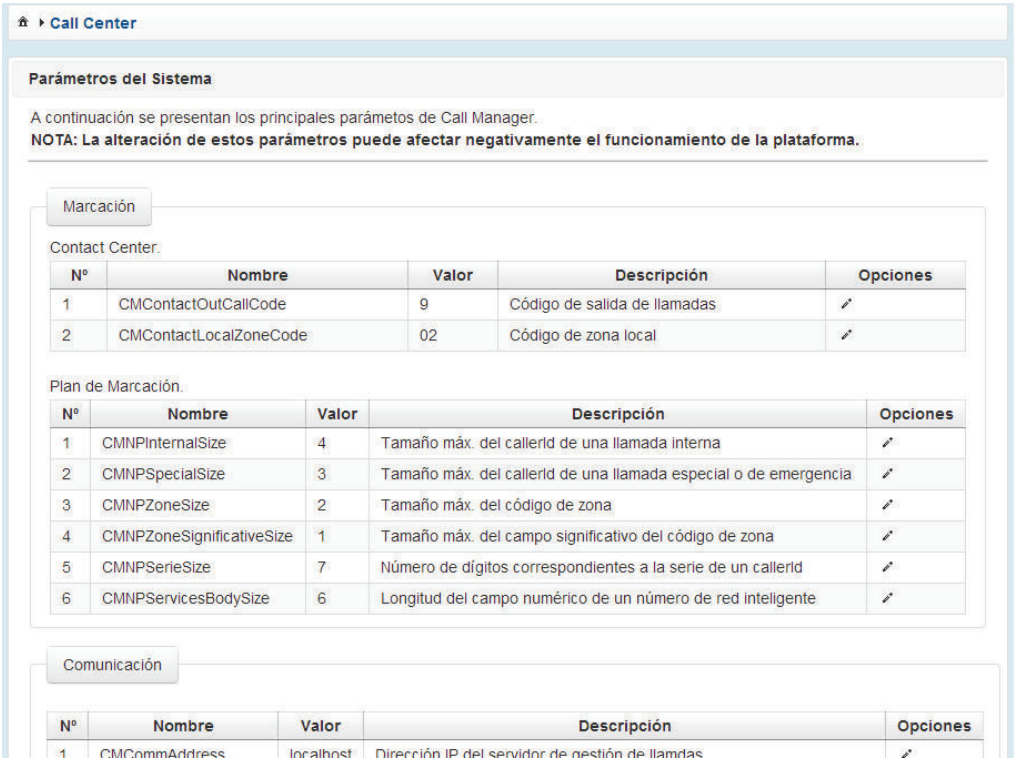


Figura 3.30 Página de Configuración de Variables del Sistema

g. **Integración con bases de datos.** Se realiza en la página *DBBIntegration.xhtml* mediante un asistente de configuración que emplea el bean *AdmDBBS*. Para ello de manera previa tiene que haber sido creado un recurso JDBC en el servidor de aplicaciones *GlassFish* que permita la interconexión con bases de datos externas (este procedimiento está detallado en la sección de ***Anexos F*** de este documento).

Figura 3.31 Asistente de Integración con un Repositorio de Información Externo

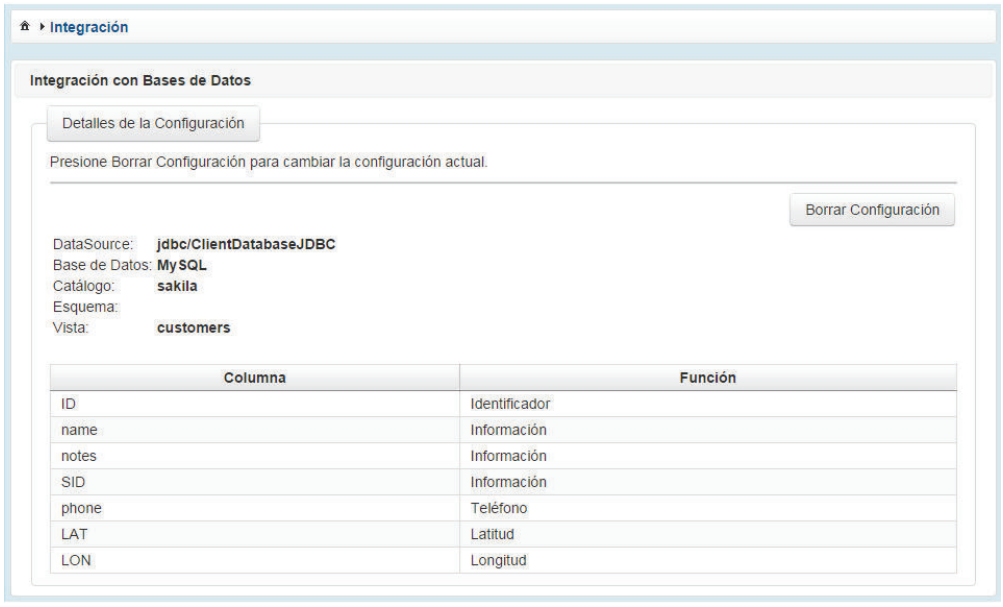


Figura 3.32 Parámetros de Interconexión Correctamente Configurados

h. **Ayuda.** Es una simple ventana emergente perteneciente a la plantilla de administración que describe cada una de las opciones del menú.
i. **Cierre de sesión.** Redirección a la página *logout.xhtml*.

3. **Supervisión.** La plantilla que se utilizará para la supervisión de la plataforma tiene por nombre *SupervTemplate.xhtml*.
   a. **Página principal.** La página *SuperADM.xhtml* es la encargada de mostrar en una tabla los agentes que se encuentran conectados a la plataforma, así como el estado de la línea telefónica asociada, siendo posible apreciar los detalles de las llamadas en curso. Adicionalmente, desde aquí el supervisor será capaz de enviar mensajes a los agentes en caso de considerarlo conveniente. En esta sección se utilizan los *beans AppLCMBean* y *AgentMonitor* principalmente.

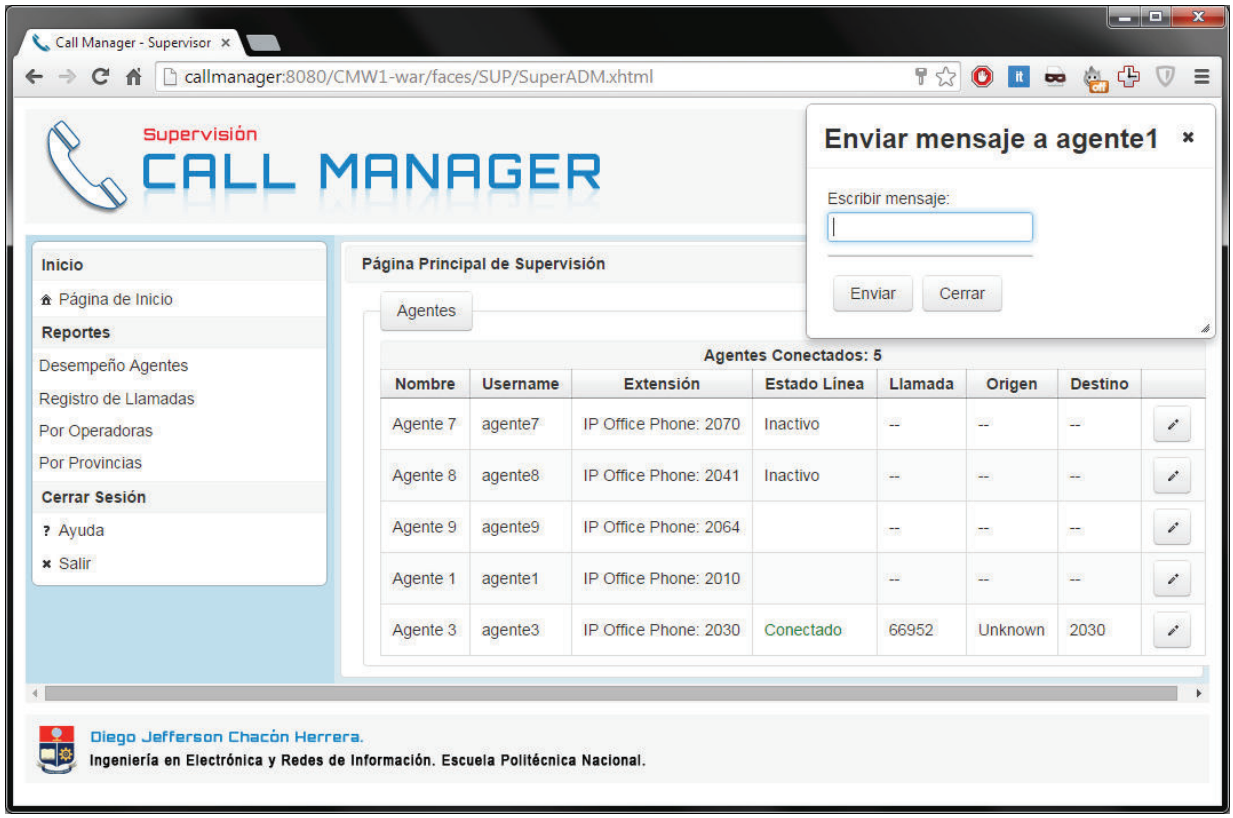


Figura 3.33 Página Principal de Supervisión

El diagrama de la figura 3.34 describe en mayor detalle el procedimiento seguido por la aplicación para el despliegue del estado en "tiempo real" de los agentes. Cada segundo, la página web realiza una solicitud al *bean AppLCMBean* (que es utilizado también por el administrador), que a su vez se interconecta con *MainControllerLocal* para recuperar el número total de agentes conectados. Si el número devuelto es mayor o igual a uno, a través de *AgentMonitor* nuevamente se solicita al componente de negocio principal la lista detallada de los usuarios para ser desplegados en la página web del supervisor.

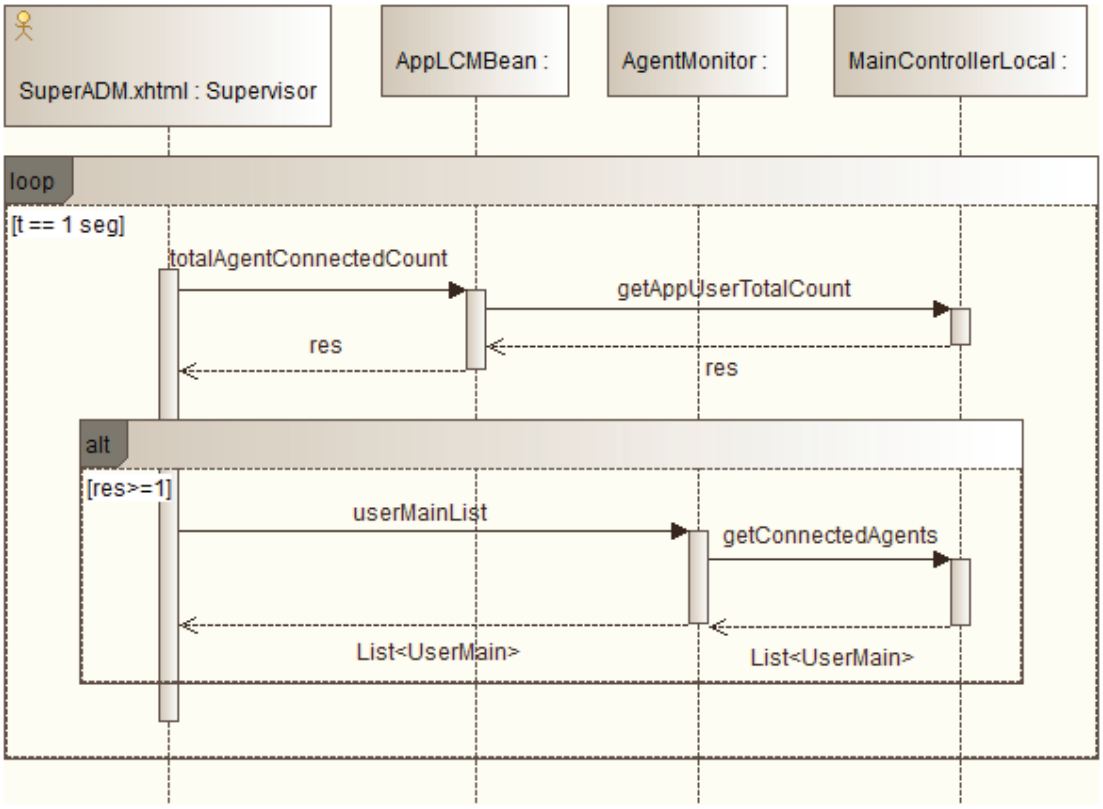


Figura 3.34 Monitoreo del Estado de los Agentes

En lo que respecta al envío de mensajes hacia los agentes de *call center*, simplemente se recoge cada cadena mediante un cuadro de texto, y desde el bean *AgentMonitor* se realiza una petición a *MainControllerLocal* del reenvío del texto al usuario correspondiente.

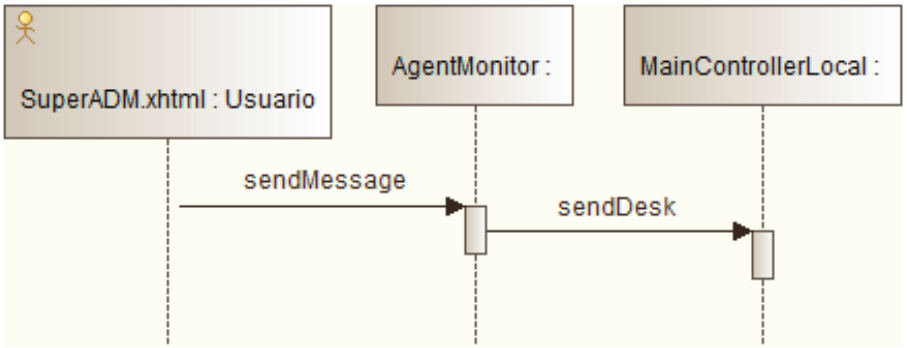


Figura 3.35 Envío de Mensajes Desde el Supervisor Hacia los Agentes

b. **Estadísticas.** La inserción de los parámetros de búsqueda en todas las páginas de esta sección es similar. Éstos están compuestos por un intervalo de tiempo definido en una fecha inicial y una fecha final, así como el tipo de llamadas (recibidas, realizadas, internas o externas).

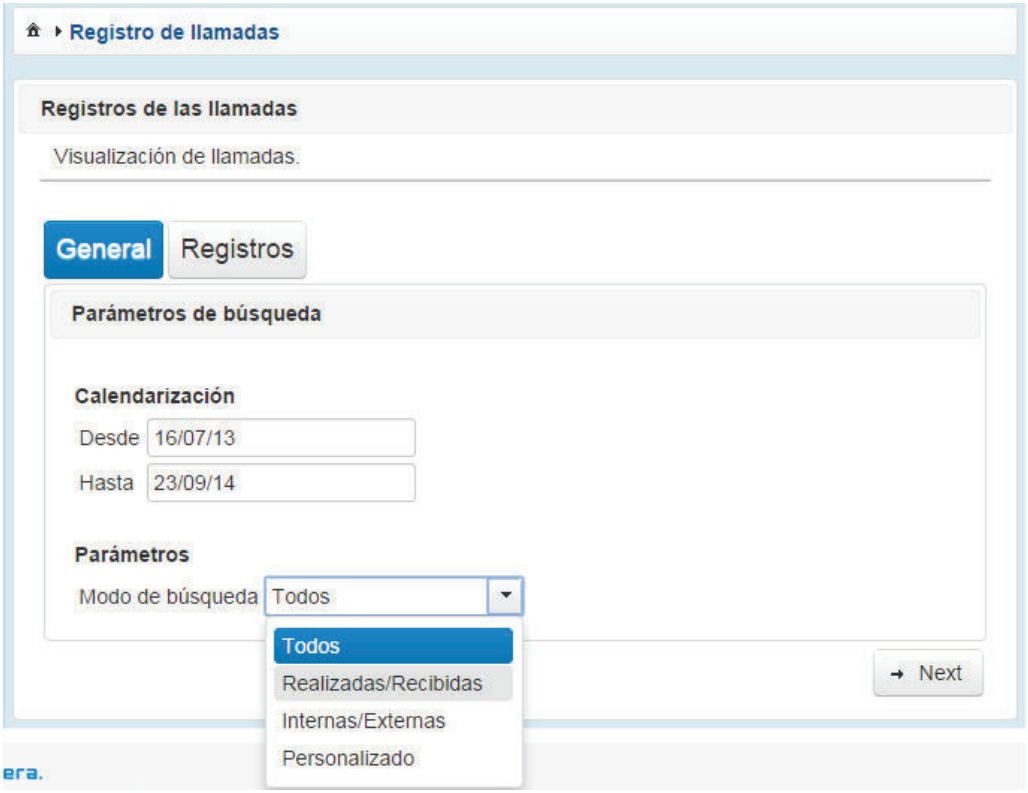


Figura 3.36 Establecimiento de Parámetros de Búsqueda de la Sección de Estadísticas

El procedimiento seguido para la búsqueda de información sigue el esquema definido en el diagrama de secuencias de la figura 3.37 en el que se ejemplifica la búsqueda de registros de llamadas. Será la página del supervisor, por tanto, la que provea las fechas de inicio y fin así como el modo de búsqueda al *bean* correspondiente (en este caso *RecordBeanV*) que a su vez utilizará las funciones predefinas en el paquete *entities* del componente de negocio para recuperar la información de la base de datos (en este caso particular, *CallRecordFacade*). El *bean* a su vez devuelve los datos a la página web para darle formato y presentarlos al supervisor.

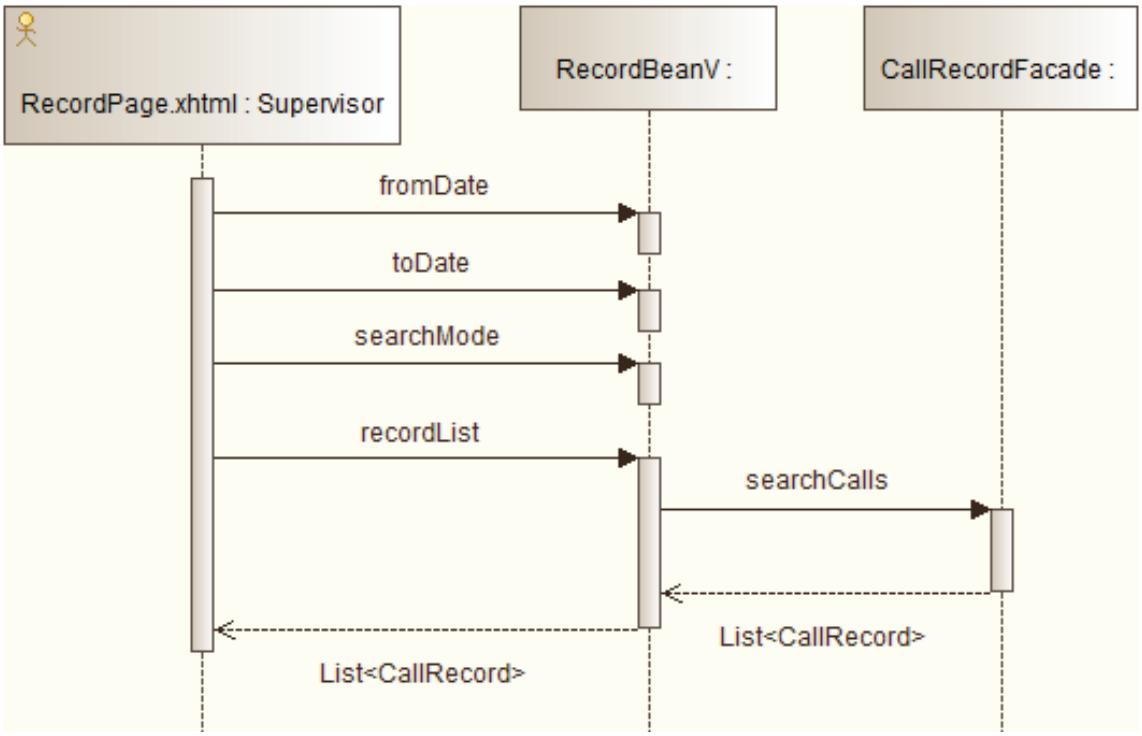


Figura 3.37 Diagrama de Secuencias de Búsqueda de Registros de Llamadas

i. **Desempeño de agentes.** Mediante un gráfico circular (o diagrama de "pastel") es posible comprobar el número de llamadas atendidas por cada uno de los agentes del *call center* en un espacio de tiempo determinado. La configuración de los

parámetros de búsqueda será similar en todas las páginas relacionadas con las estadísticas: En este caso, la página *AgentPer.xhtml* utiliza el *bean AgentPerformance* para tal acción tal y como se muestra a continuación:

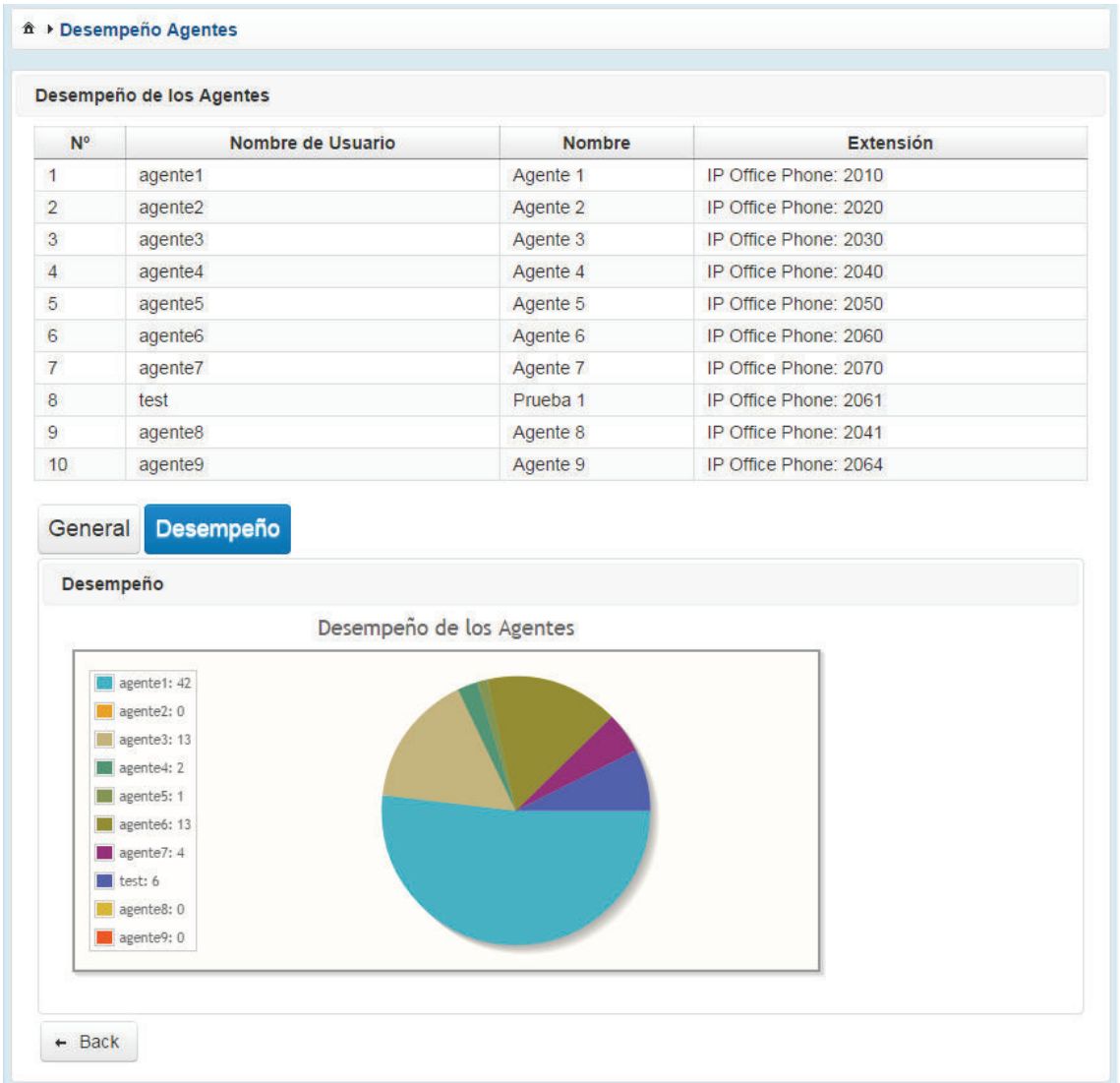


Figura 3.38 Estadística de Desempeño de los Agentes del *Call Center*

ii. **Llamadas.** En esta sección la página *CallRecordsPer.xhtml* utiliza el *bean CallRecordBeanV* para mostrar en una tabla todas las llamadas procesadas por el *call center*.

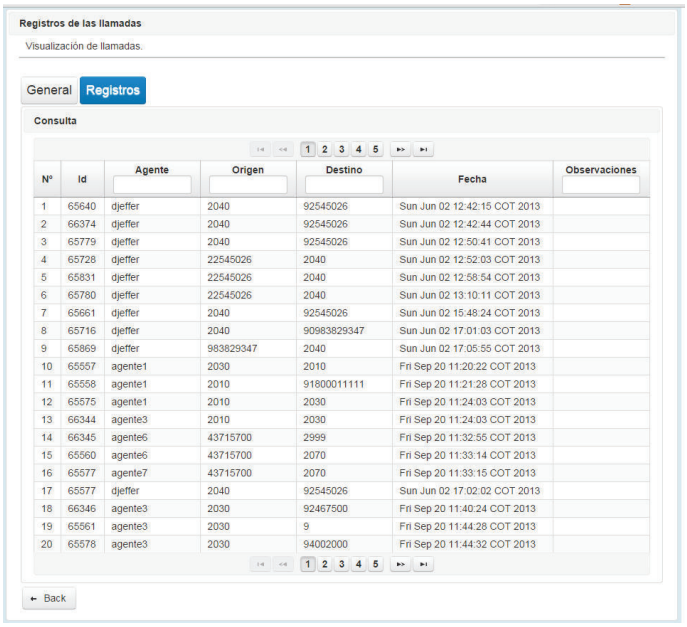


Figura 3.39 Registros de Llamadas Procesadas por el Sistema

iii. **Operadora.** Mediante un gráfico de barras es posible visualizar los operadores de servicios de telefonía implicados en las llamadas telefónicas. La página *CarrierPer.xhtml* utiliza el bean *OperatorBeanV* para ello. Aquí es de vital importancia contar con los planes de numeración previamente cargados en la BDD.

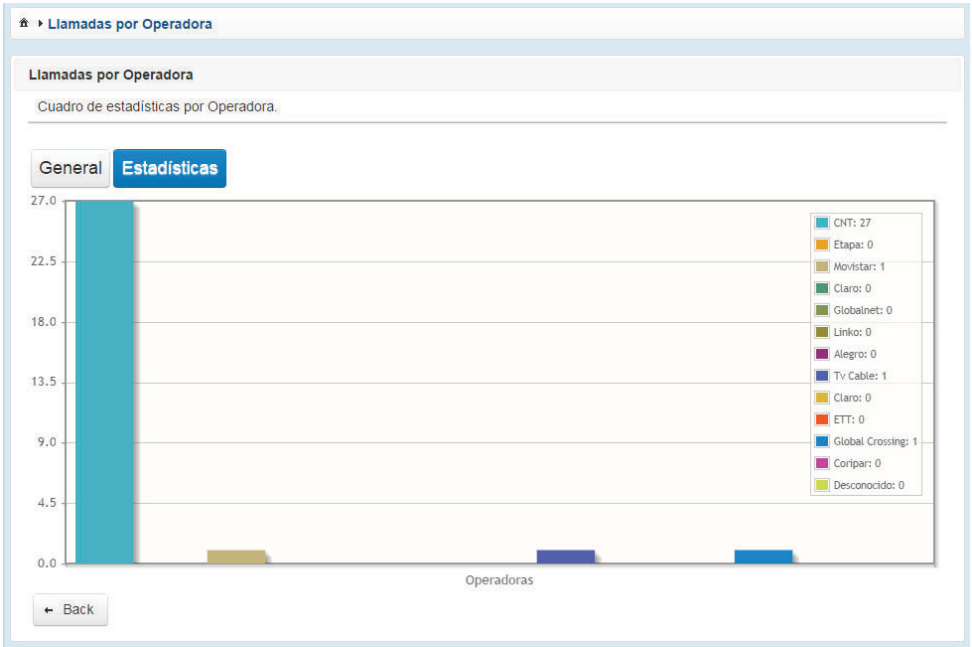


Figura 3.40 Estadística de Número de Llamadas por Operadora.

iv. **Provincia.** La página ProvincePer.xhtml utiliza el *bean ProvincePer* y librerías de *JavaScript* para mostrar en un mapa de calor la frecuencia de realización/recepción de llamadas desde números telefónicos fijos según la provincia.

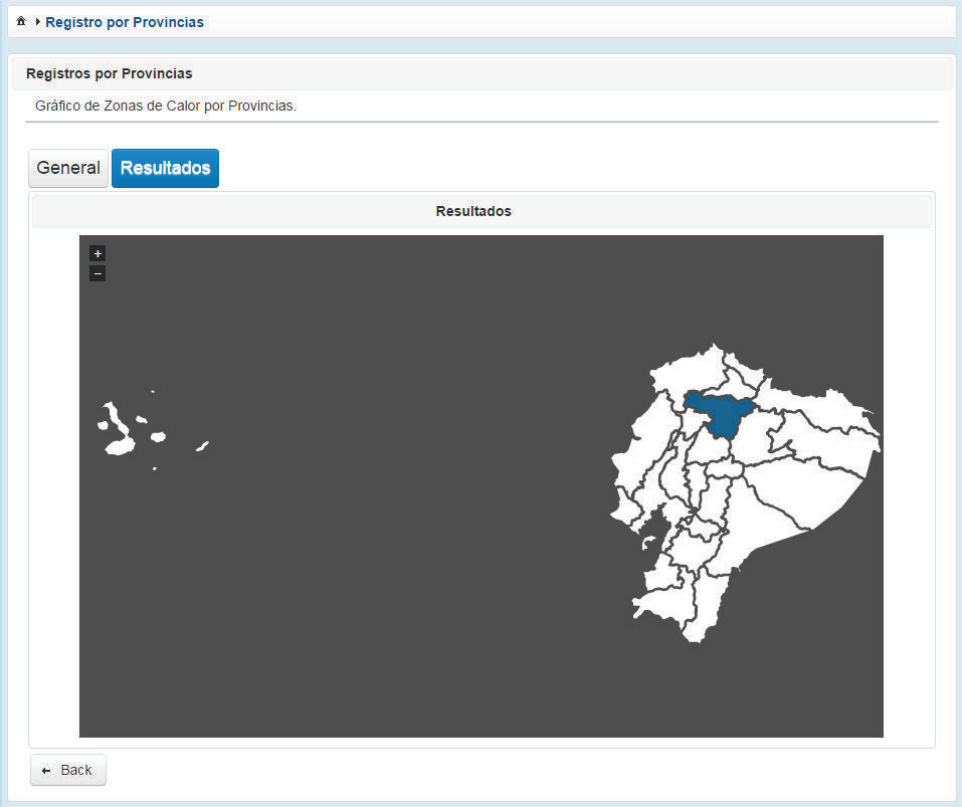


Figura 3.41 Gráfico de Zona de Calor por Provincia

v. **Ayuda.** Es una simple ventana emergente perteneciente a la plantilla de supervisión.

vi. **Cierre de sesión.** Redirección a la página *logout.hxtml.*

4. **Agente.** La plantilla de esta sección tiene por nombre *AgentTemplate.xhtml* de similares características a las demás.

   a. **Página principal.** El agente en esta página puede visualizar el historial de llamadas procesadas mediante un asistente donde es posible determinar parámetros de búsqueda. El *bean AgtCallHistory* es utilizado por la página *AgentMain.xhtml* para que esto sea así.

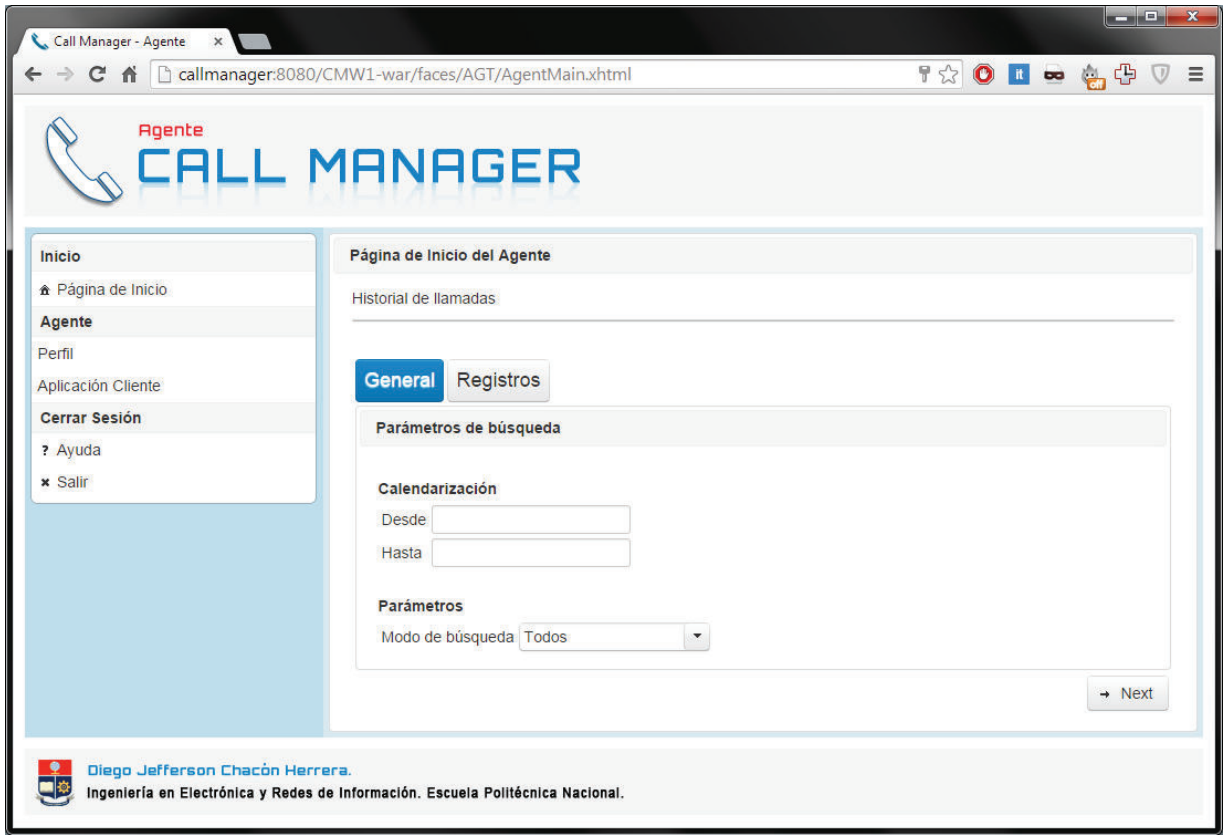


Figura 3.42 Página Principal del Agente

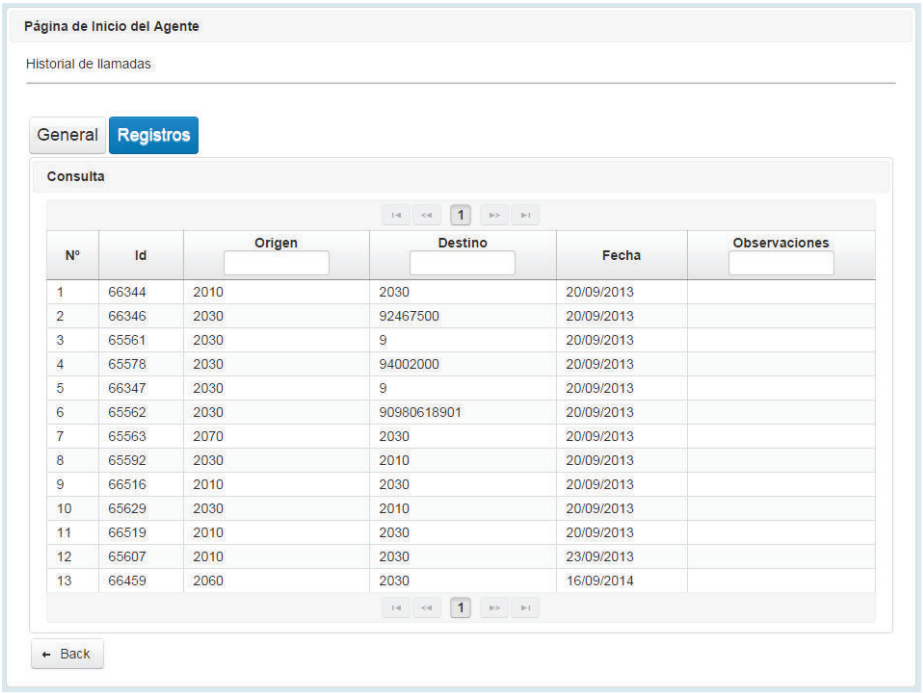


Figura 3.43 Historial de Llamadas del Agente

b. **Perfil.** La actualización del perfil del agente se realiza en la página *AgentProfile.xhtml* mediante el empleo del *bean AgtMainS*.

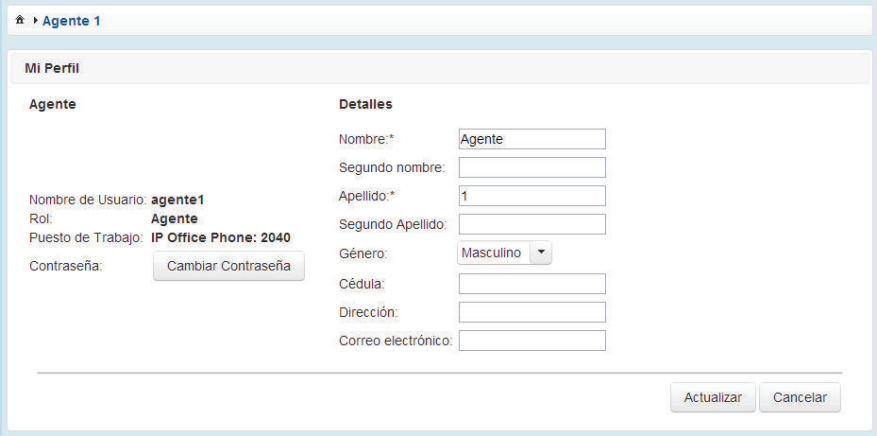


Figura 3.44 Actualización del Perfil del Agente

c. **Aplicación Cliente.** El aplicativo de escritorio también puede ser descargado desde esta página, *DesktopClient.xhtml,* que según lo descrito en la disciplina de análisis de esta fase (3.1.1.3, CA-DES) utiliza la tecnología *Java Web Start* para su despliegue. Para que esto sea posible ha sido necesario crear el archivo *CallManagerDesktop-webstart.jnlp* en el que se especifican las características de su distribución, así como un subdirectorio dedicado dentro del servidor de aplicaciones en el que se encuentran los ejecutables y bibliotecas del agente de escritorio, todos ellos autofirmados digitalmente.

Figura 3.45 Descarga del Aplicativo de Escritorio del Agente

Los detalles de la descarga e instalación del cliente de escritorio están descritos en el **Anexo D** al final de este documento**.**

d. **Ayuda.** Cuadro de diálogo descriptivo de la sección.
e. **Cierre de sesión.** Redirección a la página *logout.xhtml*.

5. **Página de cierre de sesión.** Página *logout.xhtml*.

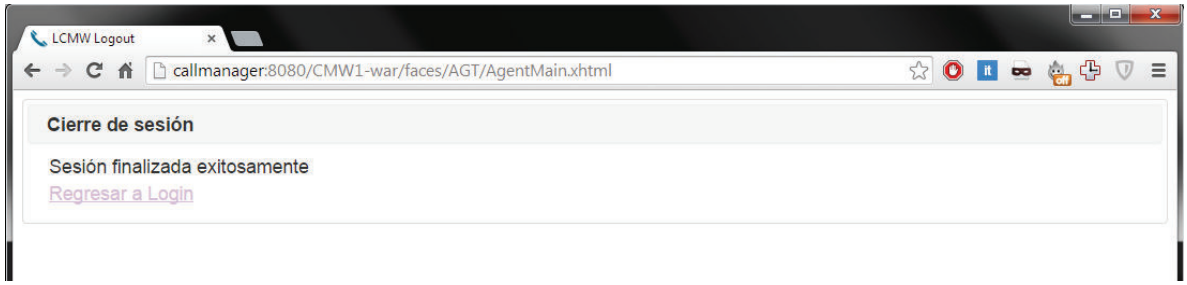


Figura 3.46 Página de Cierre de Sesión

Por otra parte, la implementación del esquema organizacional de las páginas web de este componente que han sido anteriormente descritas está, resumida en el diagrama 3.47 para un mejor entendimiento.

Finalmente, como resultado de la implementación del servidor principal se ha obtenido un archivo de extensión ***.ear*** (*Enterprise ARchive*) de nombre **CMW1.ear** que puede ser desplegado en cualquier servidor de aplicaciones que cumpla con las características definidas en los análisis de las fases de elaboración y construcción.

El procedimiento de instalación de este componente (el servidor principal) está detallado en el **Anexo C** al final de este documento.

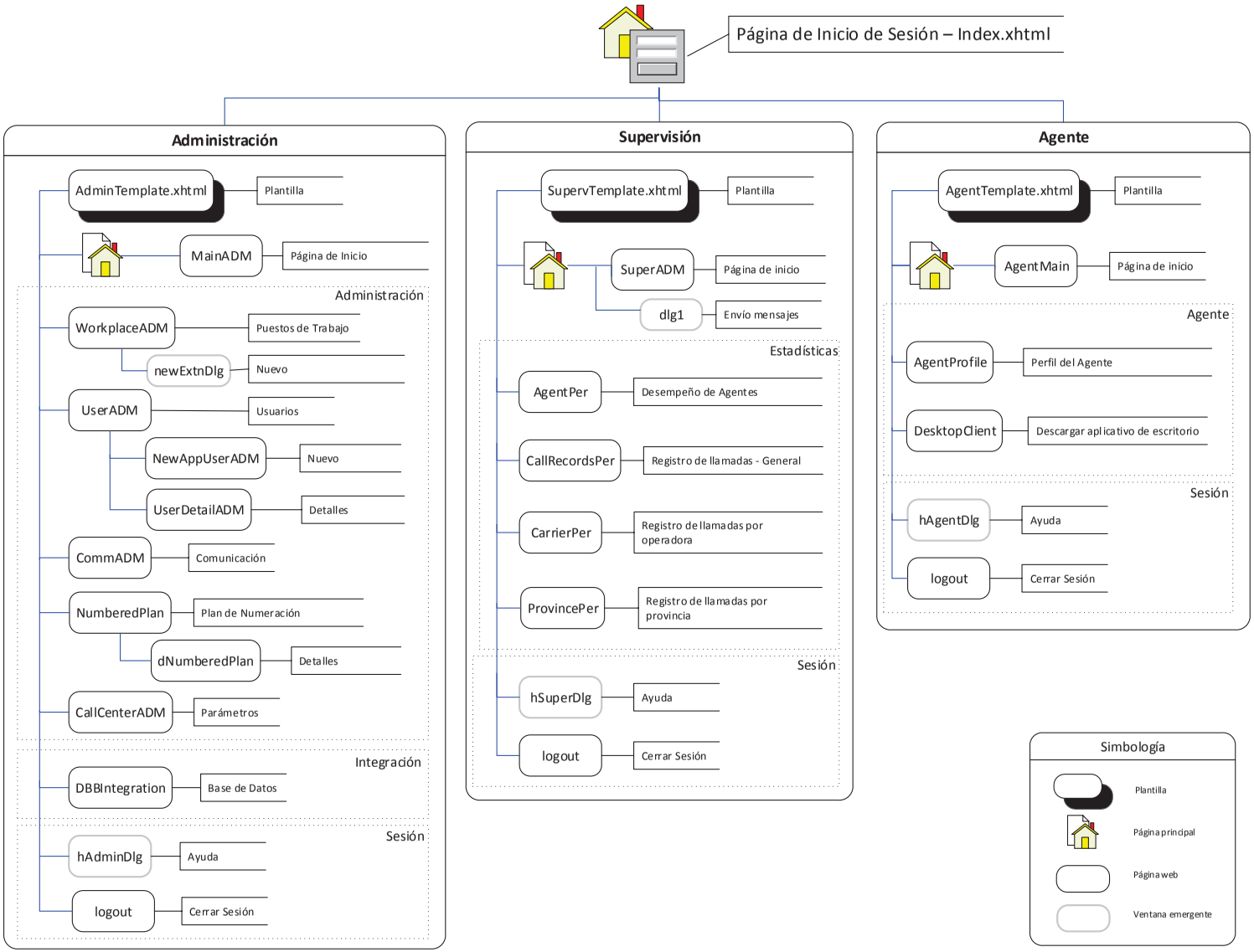


Figura 3.47 Diagrama Organizacional del Sitio Web de *Call Manager*

3.1.2.4. Cliente de Escritorio (CM-DES)

La creación de este módulo fue realizada de manera simultánea al desarrollo del módulo principal, habiéndose tratado en una primera etapa los esquemas de interconexión y comunicación entre clientes y servidor. Posteriormente se gestionaron los esquemas de autenticación y de envío/recepción de comandos y notificaciones, comprobando que los eventos relacionados con las líneas telefónicas sean apropiadamente reenviados al servidor CTI correspondiente. A continuación se trabajó en el perfeccionamiento de la interfaz gráfica en lo concerniente a la presentación de la información recibida y mapas de georreferenciación de llamadas, para finalizar con el almacenamiento de registros en la base de datos y esquemas de seguridad de bajo nivel.

El diagrama organizacional de clases de la figura 3.48 muestra la disposición en paquetes de cada uno de los elementos que componen el aplicativo del cliente y las relaciones establecidas entre los más importantes. Para visualizar los atributos y métodos pertenecientes a cada clase se recomienda visitar la sección de anexos presentada al final. Dentro del paquete *gui*, la clase más importante del aplicativo es *CallManagerDesktop*, punto central de procesamiento de todos los eventos generados por el sistema (a través de los comandos) y por el usuario (interacción mediante la interfaz gráfica). Con el propósito de que exista una única instancia de la aplicación en ejecución en un ordenador particular se ha implementado la interfaz *SingleInstanceListener*.

Por otro lado, tomando en cuenta que la gestión de las comunicaciones es realizada desde otro hilo de ejecución, se utilizó el patrón *observable/observer* para recuperar los mensajes recibidos desde el servidor principal de forma asincrónica. Por tal motivo esta clase implementa la interfaz *Observer*, en el que el procesamiento de las cadenas de texto es realizado en el método *update*. Adicionalmente, dentro del paquete *gui* se incluyen todos los cuadros de diálogo necesarios para la gestión de llamadas (*CallDialog*), sesión (*SessionDialog*), configuración del sistema (*ToolsConfigDialog*), historial de llamadas (*HistoryDialog*), entre otros; todos ellos herederos de la clase *JDialog*.

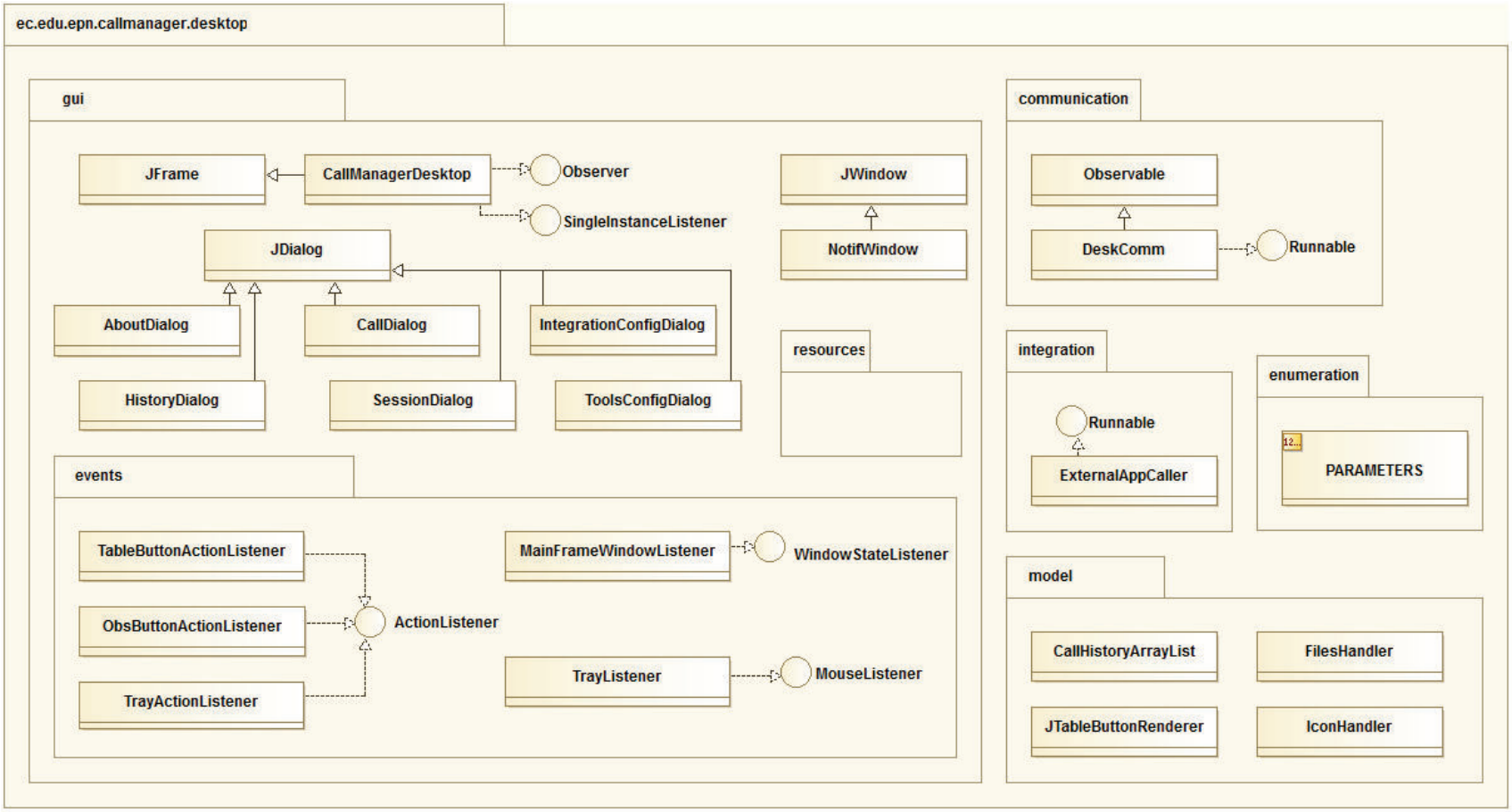


Figura 3.48 Diagrama de Clases del Cliente de Escritorio del Agente

Dentro del paquete *gui* también se incluye la clase *NotifWindow* que es la encargada de presentar en una ventana de *popup* (entiéndase emergente) la recepción de llamadas telefónicas. Además el subpaquete *events* contiene clases que implementan escuchadores (*listeners*) que permiten ejecutar procedimientos específicos ante acciones predeterminadas, como la reasignación de dimensiones de la interfaz principal (*MainFrameWindowListener*), la inserción de observaciones en el transcurso de una llamada (*ObsButtonActionListener*), la selección de una llamada específica (*TableButtonActionListener*) o el comportamiento del sistema cuando se minimice al área de notificación (*TrayActionListener y TrayListener*). Finalmente el subpaquete *resources* contiene los elementos gráficos e íconos utilizados en la presentación del aplicativo. El resultado final en lo concerniente a la interfaz gráfica tiene como base lo explicado en la sección 2.4.2.8.2 (EAD-DSK) y está mostrado en la figura 3.49.

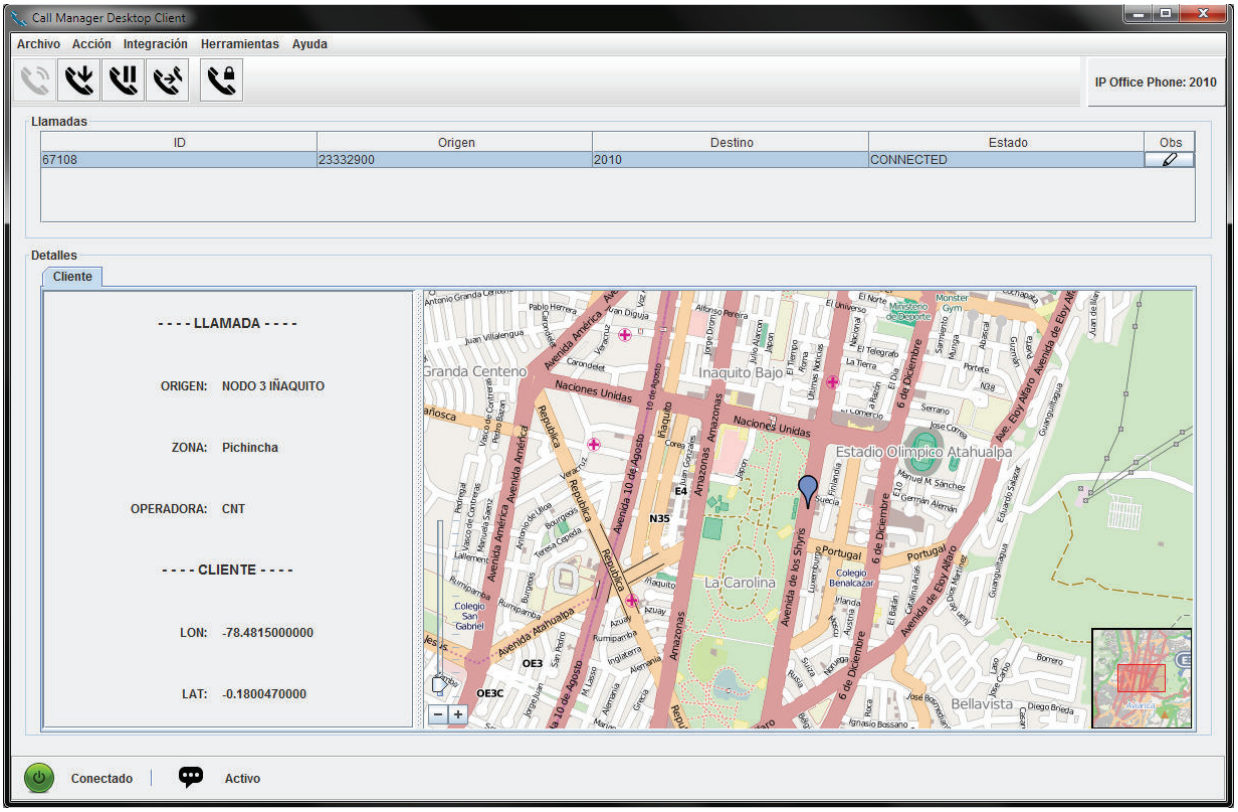


Figura 3.49 Aplicativo de Escritorio en el Transcurso de una Llamada

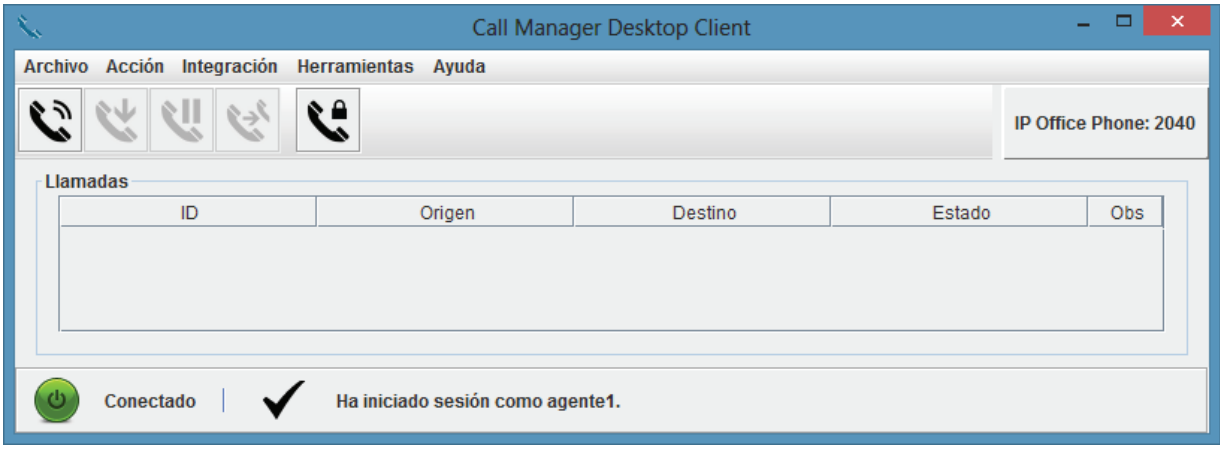


Figura 3.50 Ventana Principal del Aplicativo de Escritorio en Estado Inactivo

El paquete *communication* por otro lado, contiene la clase *DeskComm* como único elemento, la cual implementa la interfaz *Runnable* (para su ejecución desde otro hilo) y hereda de la clase *Observable* (para el envío de eventos asincrónicos al GUI). En cambio las clases ubicadas dentro del paquete *model* tienen como propósito dar formato a los registros del sistema (*CallHistoryArrayList*), manipular imágenes (*IconHandler*) o archivos de configuración *(FilesHandler*). En este sentido, el archivo ***config.conf*** almacena la dirección IP y puerto del servidor principal, e información adicional como las credenciales de acceso del agente y la posibilidad de iniciar sesión de forma automática tan pronto como se inicie el aplicativo.

Para profundizar en la recepción de los comandos se utilizará el diagrama de secuencias mostrado en la figura 3.51, que tiene como base lo especificado en la fase de elaboración 2.4.2.8.1 (EAD-DSK). Los mensajes enviados por el módulo central son recibidos en *DeskComm*, que a través de la invocación de los métodos *setChanged* y *notifyObservers* reenvía la cadena de texto correspondiente a *CallManagerDesktop* el cual de acuerdo a la información recibida procede a actualizar la interfaz gráfica.

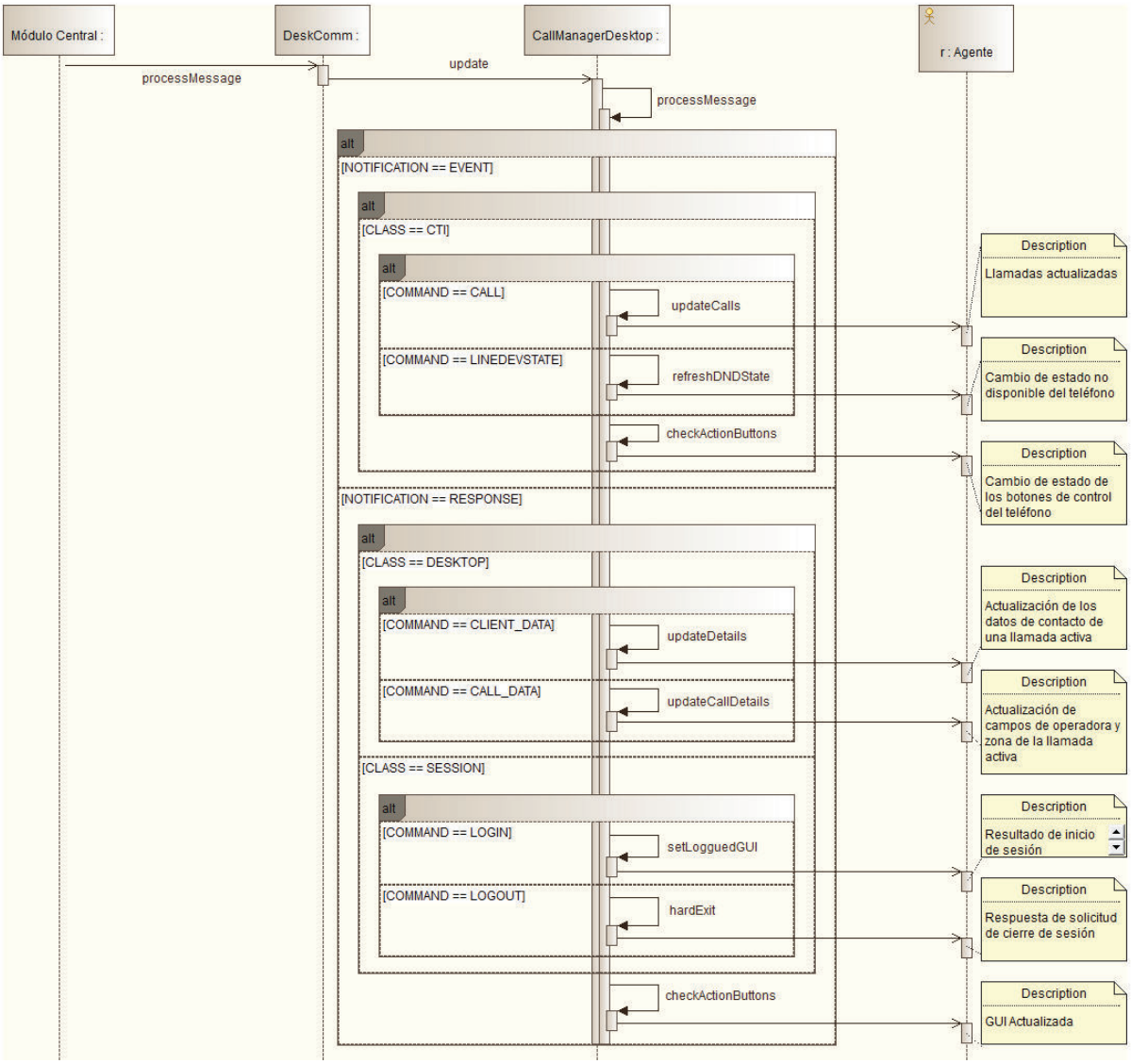


Figura 3.51 Procesamiento Detallado de Comandos en el Aplicativo de Escritorio

Para finalizar, el resultado de la creación de este módulo es el archivo Java ***CallManagerDesktopClient.jar*** que además requiere de las siguientes bibliotecas externas para su funcionamiento:

- *CallManagerCommons.jar* (descrita en la sección 3.1.2.2, CM-BC).
- *javaws.jar* (para que pueda ser distribuido desde el navegador web).
- *swing-worker.jar, swingx.jar, swingx-bean.jar, swingx-ws-2007_10_14.jar* (bibliotecas del conjunto *JXMapKit* utilizados para la georreferenciación de llamadas).

3.1.2.5. Gestión de Eventos (CM-EVN)

Si bien en las secciones anteriores han sido descritas a detalle la composición e implementación de cada uno de los módulos que componen este proyecto, es valioso y elemental profundizar en la interacción existente entre todos ellos, especialmente en lo relativo a la gestión de los eventos de las líneas telefónicas y los procedimientos de inicio y cierre de sesión, todo conforme a lo señalado en la sección 2.4.2.3 (EAD-ENV) de análisis y diseño de la fase de elaboración.

- **Gestión de Eventos Relacionados con el Sistema de Telefonía**

  Como ya se explicó con anterioridad, los eventos concernientes a las líneas telefónicas pueden tener dos orígenes: la central telefónica (ver la sección 2.4.2.3.1, EAD-ENV-IPO) o el agente de *call center* (sección 2.4.2.3.2, EAD-EVN-AGT).

  En primer lugar se describirá la implementación de los eventos generados por la central telefónica mediante el empleo del diagrama de secuencias de la figura 3.52. De forma general, cualquier evento concebido en la central telefónica respecto a una extensión, como la realización de una llamada, una transferencia, o la puesta en espera, desencadenará la detección del mismo en el servidor CTI a través del TAPI.

  De esta manera, el método de retorno *lineHandlerCallback* de la clase *CTapiHandler* del módulo CTI será el primero en detectar y recibir el incidente en cuestión. Al tratarse del cambio de estado de una línea telefónica (lo que se ha definido como un evento CTI) el servidor realiza una búsqueda de los usuarios locales cuya sesión esté activa para inmediatamente enviar un comando (entiéndase mensaje) al servidor principal mediante el método *Send* de la clase *CCommHandler* con la ayuda de *CProtocol*.

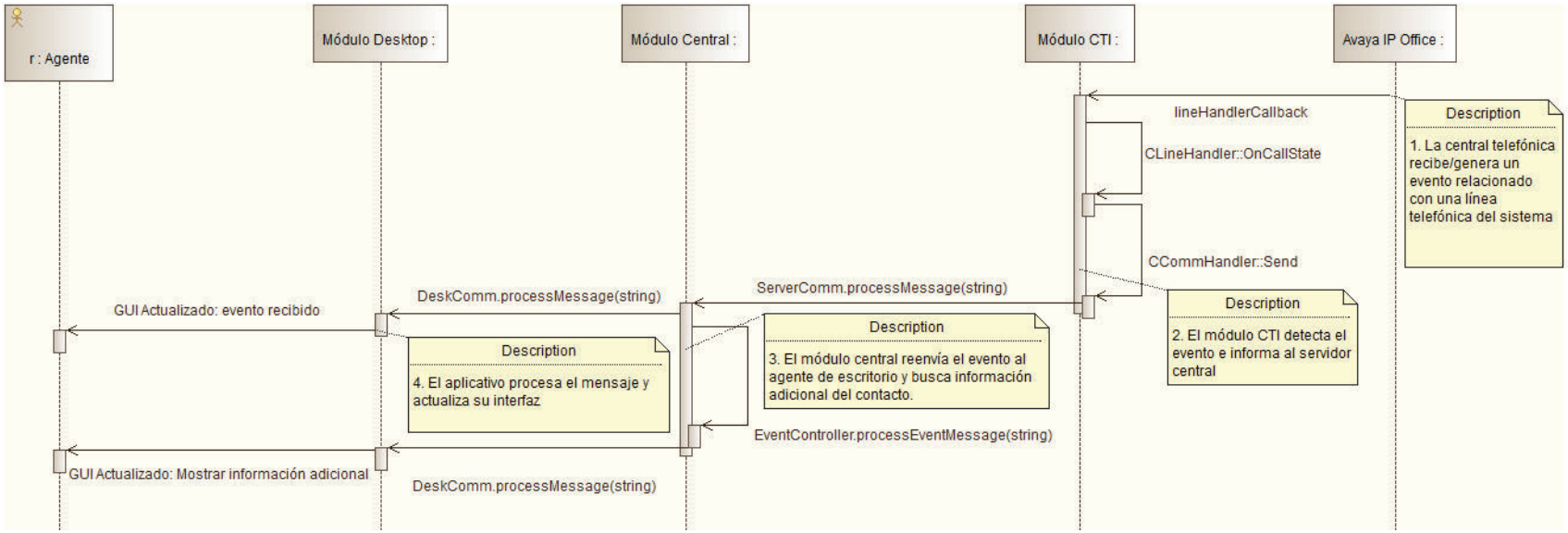


Figura 3.52 Procesamiento de Eventos Generados por la Central Telefónica

En el servidor principal el evento (mensaje) es recibido por la clase *ServerComm*, encargada de realizar dos tareas: la primera es reenviar inmediatamente el mensaje al agente mediante la interfaz *sendDesk* de *MainController*, mientras que la segunda es buscar información adicional de la llamada telefónica mediante el empleo de la función *processMessage* del mismo EJB que a su vez es transferido a *EventController* para su procesamiento. Debe notarse que en esta última parte se realiza el registro del evento en la base de datos local así como la comprobación de los planes de numeración, búsqueda de información de contacto y georreferenciación de la llamada.

Consecutivamente, el primer evento recibido por el cliente de escritorio (a través de *DeskComm*) es una notificación de tipo CTI (reenvío del evento desde el servidor principal), por lo que se utiliza una instancia de la clase *NotifWindow* para informar de este hecho al agente mediante una ventana de *popup* y se actualiza la interfaz gráfica conforme a ello. Los datos adicionales de la llamada, al llegar después, se incluyen de manera sucesiva (conforme a su llegada) en la pantalla principal para reflejar la información más reciente que pueda ser de ayuda para el agente. Considerando que puede haber más de una llamada en proceso, el aplicativo de escritorio toma medidas para determinar a qué llamada pertenecen los datos recibidos, principalmente mediante el empleo de un identificador de llamada único generado por la central telefónica que es posible obtener solamente al utilizar la versión 2.0 del TAPI proporcionado por Avaya.

Por otro lado, cuando los eventos son generados por el agente de *call center* (sentido inverso de la comunicación) el primer módulo de interacción es el aplicativo de escritorio, como se puede observar en la figura 3.53. La selección de los botones de control de la línea telefónica (como el de realización de llamadas, transferencia, cerrado o puesta en espera) origina la construcción de un mensaje de tipo solicitud que es enviado al servidor principal mediante el método *sendMessage* de *DeskComm*. De requerirse información adicional

(como el ingreso del número telefónico a llamar o transferir) primero se abre un cuadro de diálogo para la recolección de estos datos.

En el módulo principal la cadena de texto es recibida en el método *processMessage* de la instancia *DeskComm* particular relacionada con el agente en cuestión, que al ser detectada como una solicitud de cambio de estado de una línea telefónica a su vez se redirige al servidor CTI por medio de la clase *ServerComm*. Debe notarse que el registro de las solicitudes no se almacena el sistema de ninguna forma, puesto que el TAPI no garantiza que éstas sean cumplidas/ejecutadas por completo por motivos externos relacionados con el software desarrollado y exclusivos del sistema de telefonía.

Por último, el servidor CTI recoge la solicitud del agente en la instancia de la clase *CCommHandler* que con la ayuda de *CProtocol* determina que se requiere cambiar el estado de una línea telefónica, invocando los métodos de *CLineHandler* y *CTapiHandler* para el efecto que finalmente a través del TAPI instalado intentará comunicarse con la central Avaya IP Office.

Para una mejor comprensión de los dos tipos de eventos anteriores (tanto los generados por la central telefónica como los difundidos por el agente de escritorio) en el diagrama de secuencias mostrado en la figura 3.54 se presenta como ejemplo el procedimiento completo llevado a cabo por el sistema para la generación de una llamada telefónica. Está por demás decir que la implementación comienza con la solicitud de llamada generada por el agente y continúa con la detección de la llamada en el servidor CTI que luego se reenvía al mismo agente. Es importante notar que el mismo escenario ejemplificado es aplicable (con mínimos cambios) para eventos como la transferencia, cuelgue, puesta en espera una llamada o cambio del estado de disponibilidad del teléfono.

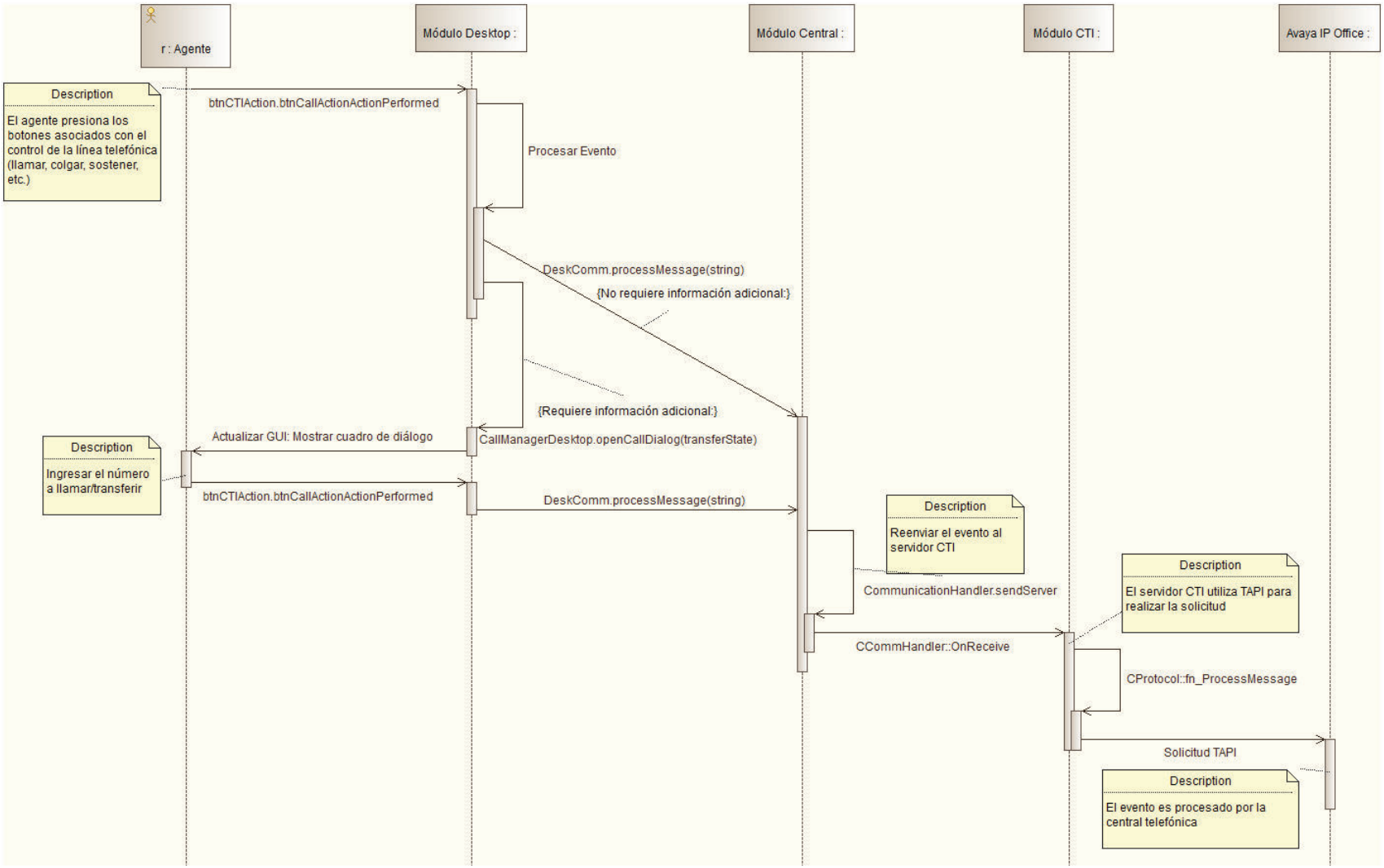


Figura 3.53 Implementación del Procesamiento de Eventos Generados por el Agente de *Call Center*

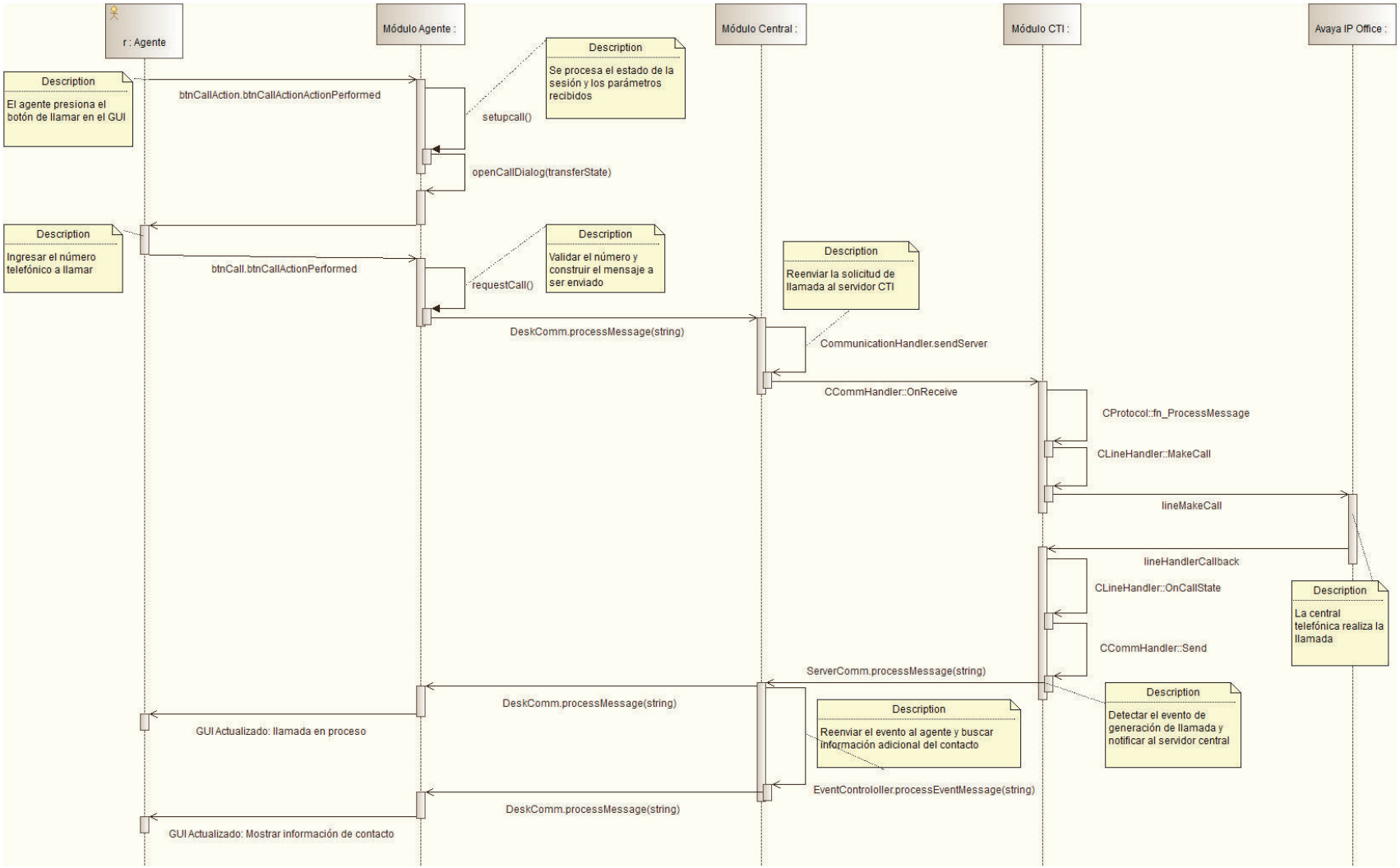


Figura 3.54 Ejemplo de Implementación de la Generación de una Llamada Telefónica en *Call Manager*

- **Inicio y Cierre de Sesión de los Agentes**

Esta sección tiene como propósito cubrir los detalles de implementación según lo planteado en el diseño de la sección 2.4.2.2 (EAD-SES) de la fase de elaboración con respecto al inicio y cierre de sesión de los agentes en el sistema.

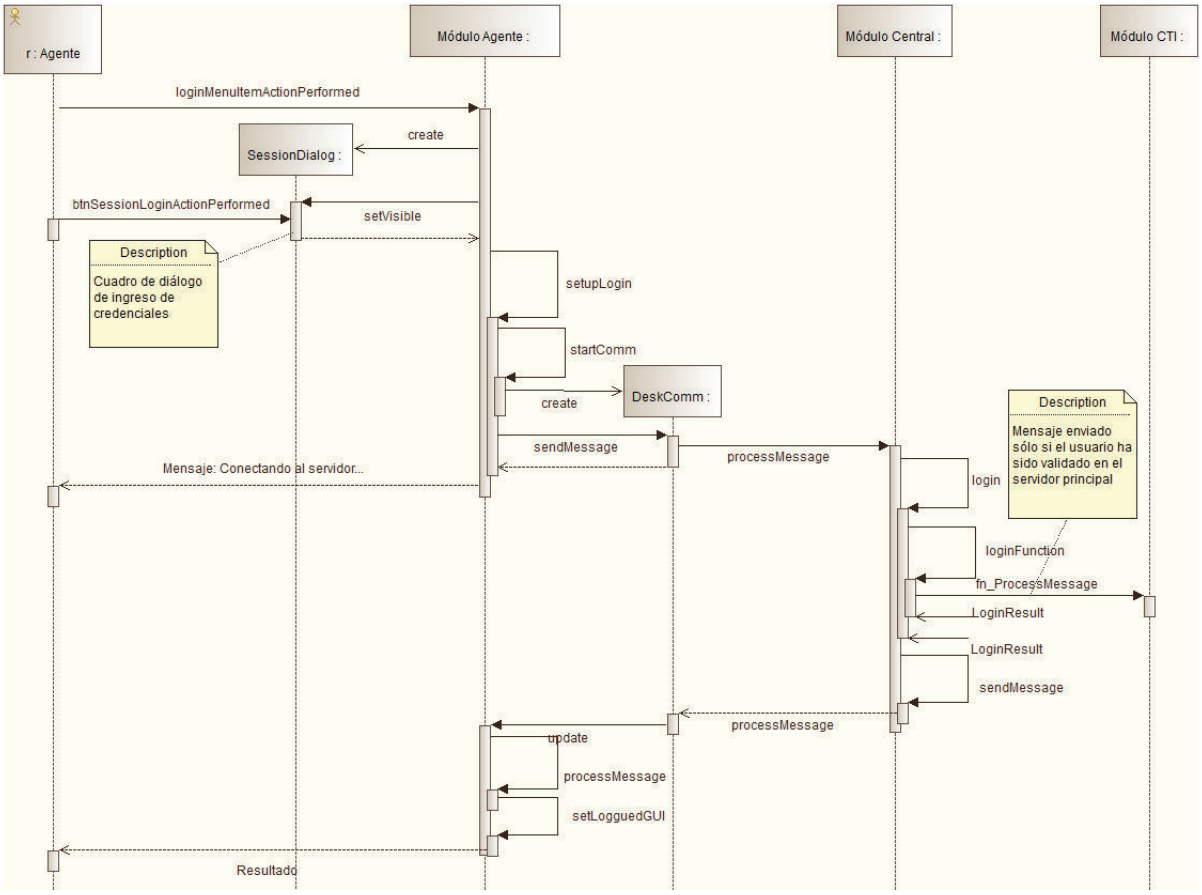


Figura 3.55 Diagrama de Secuencias de Inicio de Sesión de los Agentes

En primer lugar (ver figura 3.55), el agente de *call cente*r abre el aplicativo de escritorio y selecciona la opción de inicio de sesión mediante la elección de la opción correspondiente (desde la barra de menús o a través de la pulsación del atajo de teclado *Ctrl* + *S*). Inmediatamente se abre un cuadro de diálogo para la inserción de credenciales, las cuales son retransmitidas al servidor principal para su validación. Para ello primero es necesario establecer la comunicación entre el cliente de escritorio y el servidor, efectuado en la

función *setupLogin* de *CallManagerDesktop.* Posteriormente el servidor recibe la cadena de texto e invoca los métodos *login* y *loginFunction* del EJB *MainController* para autenticar al agente de acuerdo a la información almacenada en la base de datos local del aplicativo. En caso de obtenerse un resultado positivo, el servidor envía un mensaje al módulo CTI informándole que éste puede enviarle eventos de la línea telefónica perteneciente al agente recientemente autenticado. Al recibir este mensaje el módulo CTI comprueba el estado de la extensión que seguidamente será notificado como un evento CTI regular. Por último, el servidor principal despacha el resultado al aplicativo de escritorio que a continuación actualiza su interfaz conforme a ello.

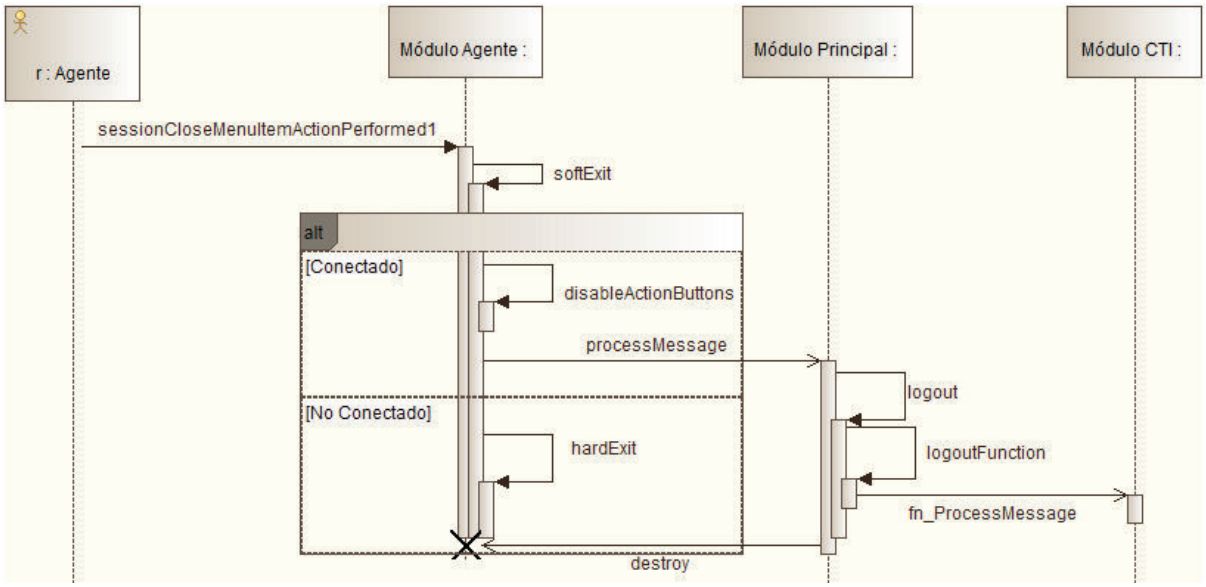


Figura 3.56 Diagrama de Secuencias de Cierre de Sesión de Agentes

En lo que respecta a la implementación del cierre de sesión (figura 3.56), el agente selecciona la opción del menú o el atajo de teclado *Ctrl + F*, que invocará el método *SoftExit* de *CallManagerDesktop* que será el responsable de seguir un procedimiento "limpieza" de recursos en todo el sistema. Si el agente está conectado al módulo central, primeramente se desactivarán todas las funcionalidades de la interfaz mediante el método *disableActionButtons* seguido del envío de una notificación del intento de desconexión del usuario. El servidor principal a su vez invoca los métodos *logout* y *logoutFunction*

desde los cuales libera los recursos asignados y además informa al servidor CTI que ya no es necesario el envío de eventos relacionados con la extensión del agente en cuestión. El cierre de sesión culmina con el envío de un último mensaje de confirmación desde el servidor principal que ocasiona el cierre del aplicativo de escritorio a través de la función *hardExit*, la cual también es invocada en el caso que el aplicativo por cualquier motivo no haya estado conectado previamente al sistema.

3.1.3. PRUEBAS DE LA FASE DE CONSTRUCCIÓN (CP)

De forma general, en la disciplina de pruebas de la fase de implementación se realizan todas las comprobaciones que determinen el adecuado funcionamiento e interacción de todos los componentes, programas y ejecutables resultantes de esta fase, así como de la verificación de la inclusión de los procedimientos, actividades y algoritmos descritos en la disciplina de diseño de la fase de elaboración. Tomando en cuenta la estrecha relación existente entre las disciplinas de implementación y pruebas, más aún cuando existen varios componentes que pueden estar distribuidos en distintas plataformas, está por demás aclarar que las comprobaciones del funcionamiento del sistema fueron realizadas en la construcción de cada elemento. En ellas se pudo determinar que el sistema funciona de forma estable con 20 agentes simultáneos, sugiriendo la posibilidad de soportar un mayor número de agentes *desktop* concurrentes.

Sin embargo, dentro del desarrollo de software, el término “pruebas” puede ser muy amplio y abarcar complejos sistemas de verificación de los productos. Con el propósito de no extender innecesariamente la longitud de este documento, el término “pruebas” se referirá a las “*pruebas de aceptación*” que serán descritas en la fase de transición y que permiten concluir el grado de alineamiento de la solución de acuerdo con los requisitos planteados en la fase de inicio; a diferencia de lo que podría ser el análisis, diseño y ejecución del banco de todos los posibles escenarios en los cuales la aplicación podría funcionar, o del establecimiento de parámetros de rendimiento como ancho de banda, memoria, procesamiento, número máximo de conexiones simultáneas, etc.; temas que no fueron planteados en el alcance de este proyecto.

3.2. FASE DE TRANSICIÓN (T)

En esta última fase del proyecto son realizadas varias actividades. Una de ellas es la gestión del despliegue de los componentes (instalación) que puede ser realizada mediante archivos ejecutables, scripts o *pluggins*. Otra es la verificación de cumplimiento del proyecto conforme a las necesidades de los usuarios mediante la realización de pruebas de aceptación. Además es aquí donde se recibe la retroalimentación de los usuarios finales con el propósito de corregir cualquier fallo no previsto y finalmente crear la documentación orientada al cliente como manuales y guías de administración y uso.

En lo que respecta a la creación de instaladores, la descripción de las disciplinas de análisis, diseño e implementación de aquéllos serán resumidas en la misma disciplina de despliegue por su poca relevancia con relación a las fases anteriores (en otras palabras, no serán descritos el diseño ni la implementación de los instaladores). El manual de administración de la plataforma así como también las guías de instalación de los componentes del sistema, por otro lado, están descritos en la sección de anexos al final de este documento.

3.2.1. DESPLIEGUE (TD)

Como resultado de la implementación de la fase anterior (construcción) fueron obtenidos archivos ejecutables, *scripts* y bibliotecas externas que deben ser instalados adecuadamente en los servidores (para los módulos CTI y principal) y los computadores de los agentes (aplicativos de escritorio). En este sentido, la disciplina de despliegue de la fase de transición describe las técnicas y los procedimientos necesarios para la instalación de dichos componentes en el escenario de evaluación de la solución (3.2.2.1, TP-EV).

3.2.1.1. Central Telefónica Avaya IP Office 500 (TD-IPO)

En el lugar donde fue realizado el despliegue de *Call Manager* se efectuó previamente la instalación y configuración de la plataforma de telefonía Avaya IP Office 500 V2, habiendo quedado perfectamente operativa. Cabe aclarar que el

diseño e instalación del sistema de telefonía excede el alcance de este proyecto y se sugiere investigar la documentación del fabricante en caso de ser necesario. Esta sección, por tanto, tiene como finalidad describir las configuraciones necesarias para el funcionamiento de sistemas de tipo CTI interconectadas con la central telefónica en cuestión.

De acuerdo a lo establecido en las disciplinas de análisis de la fase de inicio IA-AR-1 (sección 2.3.4.2.1) e IA-AA-IPO (2.3.4.3.2) el sistema de telefonía que utiliza IP Office V2 500 de Avaya consta de lo siguiente:

- Infraestructura de red (*networking*) y cableado estructurado.
- Chasis IP Office 500 V2 actualizado a la versión 9.0.200.860.
- Licenciamiento completo.
- Tarjetería y licenciamiento tanto de usuarios como de enlaces troncales conforme a las necesidades de la institución.
- Licenciamiento CTI Link Pro (ver figura 3.57).

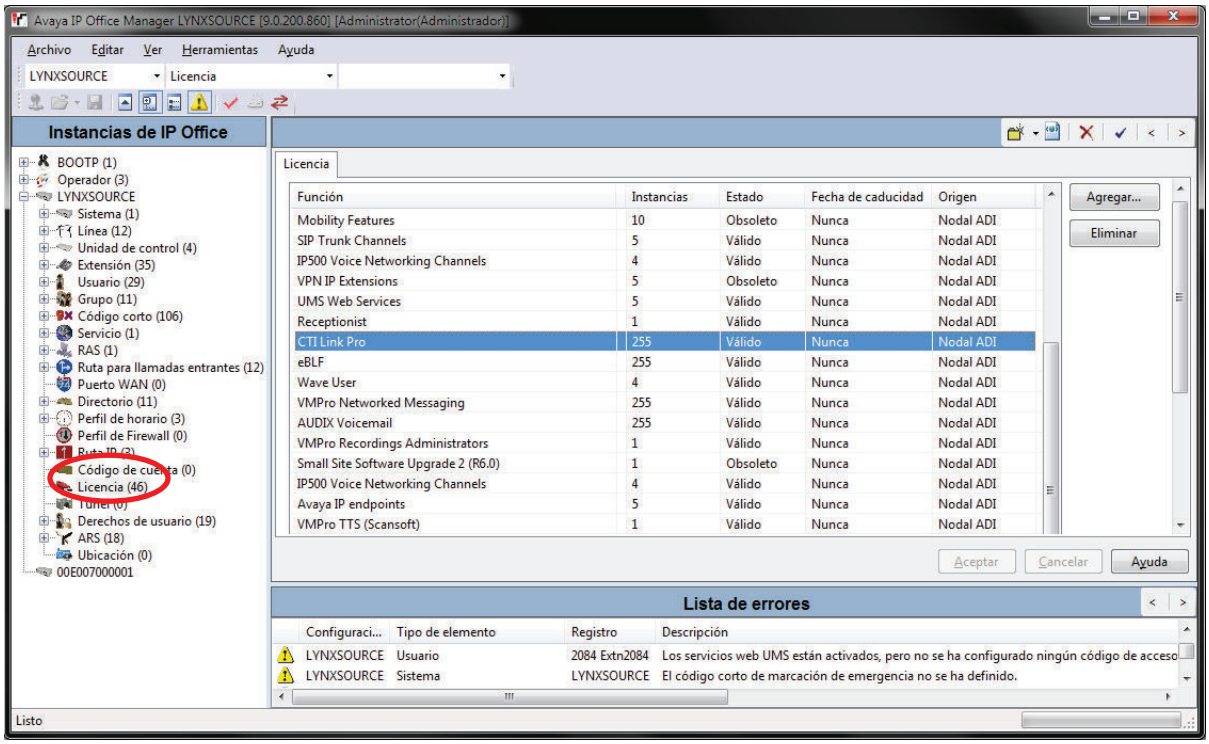


Figura 3.57 Licenciamiento CTI Link Pro Activo en IP Office

- Teléfonos físicos o lógicos (*softphones*) para los usuarios de la plataforma. En este escenario de despliegue los usuarios trabajan en un entorno mixto conformado por teléfonos IP, *softphones* y digitales, los cuales son manejados por TAPI de forma equivalente y sin excepciones respecto al tipo de teléfono.
- Configuración de permisos, códigos o planes de marcación que posibiliten la generación y recepción de llamadas telefónicas en el *call center*. Cabe recalcar que todas las configuraciones realizadas en la central telefónica fueron realizadas por el autor conforme a la documentación oficial del fabricante y conforme a la simulación de un entorno de *call center* genérico.
- Servicio de telefonía (acceso a la PSTN) a través del proveedor de servicios CNT con la funcionalidad de identificación de llamadas habilitada.

3.2.1.2. Servidor CTI (TD-CTI)

Como es de esperarse, para que este componente pueda funcionar de forma apropiada primero debe ser instalado y configurado el TAPI 2.1 de Avaya incluido en el paquete *User4_2_43*, cuyo procedimiento de instalación además del respectivo componente en cuestión está descrito en la sección **Anexo B**.

El módulo CTI requiere de la creación de un instalador que de acuerdo a lo planteado en la sección 3.1.2.1 (CM-CTI) tiene por finalidad no sólo colocar los archivos *CallManager.exe* y *CallManager.conf* en lugares apropiados, sino también crear un enlace de acceso directo en el directorio de arranque de Windows (*CallManager.lnk*) y recibir parámetros de instalación como el número de puerto de escucha para la interconexión con el servidor principal o la selección que autoriza el inicio automático de la aplicación en el arranque del sistema operativo. Conforme a lo anterior, se ha creado el proyecto ***LCMSetupDeployment*** en el IDE Windows Visual Studio 2008 que ha dado como resultado el paquete de Windows *Installer* ***CallManagerSetup.msi*** que puede ser instalado en plataformas de 32 bits.

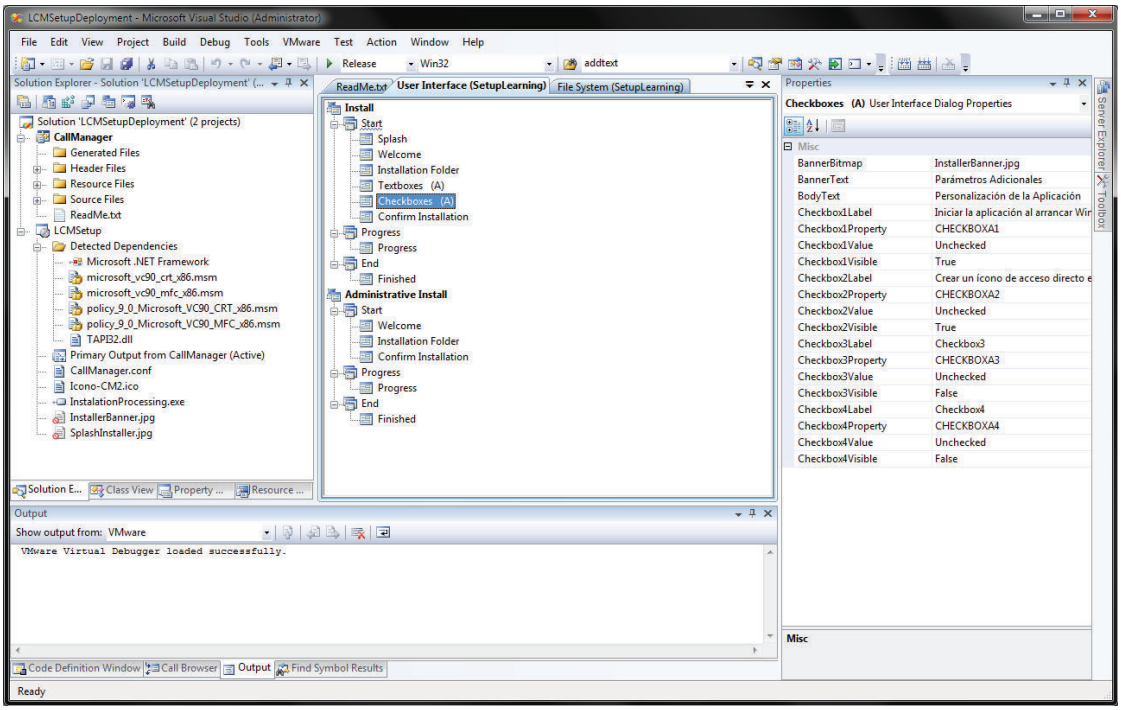


Figura 3.58 Creación del Paquete Instalador del Servidor CTI

De esta manera es posible determinar las características y requerimientos finales del componente CTI de *Call Manager*, definidos a continuación:

- Sistema Operativo Windows Vista o superior. También es posible instalar sobre Windows Server 2008 R2 o superior.
- .NET Framework 3.5 o superior.
- Windows *Installer* 3.1.
- Credenciales de interconexión con la central telefónica configuradas en la instalación del controlador TAPI de Avaya.
- Excepción en el *firewall* del sistema operativo que permita la interconexión con el servidor principal a través del puerto 47915 (por defecto).

Debido a que los parámetros de operación del aplicativo son configurados en las instalaciones tanto del controlador TAPI como del componente CTI mismo, no son necesarias configuraciones adicionales para su funcionamiento (es decir, el programa queda listo para ser utilizado después de la instalación). Por otro lado, la desinstalación de éste se puede realizar desde el panel de control de Windows.

3.2.1.3. Servidor Principal (TD-CORE)

Conforme a lo obtenido en la disciplina de despliegue (CM-CORE) el resultado de la implementación del servidor principal es un archivo de extensión *.**ear*** (Enterprise ARchive) denominado ***CMW1.ear*** que debe ser desplegado en el servidor de aplicaciones, por lo que no hay necesidad de crear instaladores de ningún tipo. Sin embargo, para que el archivo en cuestión pueda ser correctamente cargado en *GlassFish* primero deben ser realizadas ciertas tareas tanto en el gestor de base de datos como en el servidor de aplicaciones. El procedimiento de instalación a detalle de este componente se describe en la sección **Anexo C**.

Tomando en cuenta la naturaleza multiplataforma y distribuida de este componente, la base de datos local puede estar físicamente localizada en otro equipo que no sea el mismo servidor principal siempre y cuando ésta cumpla con los requisitos planteados al final de esta sección. El primer paso del despliegue del servidor principal, por tanto, recae en la creación e inicialización de la base de datos local que debe tener por nombre ***CallManagerDB.*** Dicha base debe tener asociado un usuario con privilegios de administración llamado ***managerUser*** cuyas credenciales son utilizadas en la interconexión desde el servidor de aplicaciones. Este usuario debe ser capaz de ejecutar los siguientes *scripts*:

- ***Tablas.sql.*** Creación de las tablas de la base de datos.
- ***InsercionDatos.sql.*** Creación de parámetros esenciales y por defecto del sistema.

El siguiente paso consiste en la instalación del servidor de aplicaciones *GlassFish* (también detallado en el anexo C) así como de la configuración de los *pools* de conexiones tanto de la base de datos local (PostgreSQL) como de la base de datos externa (en caso de ser necesario). Una vez comprobada la interconexión entre las BDDs y el servidor de aplicaciones es posible continuar con el despliegue del archivo *CMW1.ear* desde la consola de administración de *GlassFish*.

Como resultado de esta fase se han establecido los siguientes requisitos finales para la instalación y despliegue del servidor principal:

- Sistema de gestión de base de datos PostgreSQL 9.2 o superior para la creación de la base de datos local de la solución.
- Plataforma Java JDK 7 Update 3 o superior según la versión requerida por el servidor de aplicaciones.
- Servidor de aplicaciones *GlassFish Open Source Edition* versión 3.1.2 o superior.
- Los requerimientos de sistema operativo y hardware están sujetos a los requisitos impuestos tanto por el JDK[18] como por el servidor de aplicaciones[19] antes señalados. De forma general es posible utilizar equipos Windows, Solaris, Linux y MacOS de 64 bits.
- Controladores de los JDBCs para interconexión con bases de datos externas:
    - PostgreSQL: *postgresql-9.2-1002.jdbc4.jar* o más reciente.
    - MySQL: *mysql_connector_java_5.X.XX_bin.jar* o posterior.
    - Microsoft SQL Server: *sqljdbc4.jar*.
- Configuración de políticas en el cortafuegos que habiliten los siguientes puertos TCP/IP utilizados por *Call Manager*:

Puerto	Propósito	Sentido
80	Acceso a *Call Manager Web*.	Entrante
4848	Consola de administración de *GlassFish*.	Entrante
47915	Interconexión con el servidor CTI.	Saliente
47925	Escucha de clientes *desktop*.	Entrante
5432 (defecto)	Interconexión con la base de datos local *PostgreSQL*.	Saliente
3306, 1433	Puertos por defecto para interconexión con BDD Externas (*MySQL* y *SQL Server*)	Saliente

Tabla 3.4 Puertos Utilizados por *Call Manager*

[18] *http://www.oracle.com/technetwork/java/javase/config-417990.html*
[19] *http://docs.oracle.com/cd/E26576_01/doc.312/e24939/release-notes.htm*

3.2.1.4. Agente de Escritorio (TD-DSK)

Si bien los usuarios del aplicativo de escritorio podrían ejecutar directamente el archivo ***CallManagerDesktopClient.jar*** para su uso, lo más apropiado es proporcionar medios que faciliten tanto la instalación como la remoción de los componentes que conforman este programa. En tal virtud, y de acuerdo a lo planteado en la sección 3.1.1.3 (CA-DES), en este proyecto se han considerado como válidas dos alternativas de instalación de *Call Manager Desktop* cuyos procedimientos de instalación están debidamente detallados en el **Anexo D**.

3.2.1.4.1. Archivo Instalador para Sistemas Operativos Windows

Tomando en cuenta la popularidad de los sistemas operativos de Microsoft se ha visto apropiado crear un instalador dedicado que siga los esquemas tradicionales de despliegue de software en dichas plataformas. Para ello ha sido creado el proyecto ***CallManagerDesktopSetup*** en el IDE Microsoft Windows Visual Studio 2008 cuyo resultado es el paquete de Windows Installer ***CallManagerDesktopSetup.msi***. Es importante notar que el archivo principal de extensión *.jar* primero tuvo que ser encapsulado como un archivo ejecutable de Windows (*.exe*) mediante el empleo de la aplicación ***Launch4j***. Además, todos los archivos del aplicativo están localizados en la carpeta *"Call Manager Desktop"* que se encuentra ubicada el directorio ***AppData*** de cada usuario de tal forma que los archivos de configuración puedan ser independientes, haciendo posible el almacenamiento de las credenciales de los agentes para que la aplicación pueda iniciar sesión de manera automática. Como resultado, los siguientes requisitos finales han sido determinados:

- Java Runtime Environment (JRE) 7 Update 3 o superior.
- Windows Installer 3.1 o superior.
- Configuración de políticas en el cortafuegos que habiliten el tráfico por el puerto 47925 (por defecto).
- Conexión a internet para la funcionalidad de georreferenciación de llamadas.

Por otro lado, en lo correspondiente a la desinstalación del programa debe seguirse en procedimiento estándar de remoción de aplicaciones en Windows mediante la sección *Programas y características* del Panel de Control.

3.2.1.4.2. Empleo de Java WebStart para todo Tipo de Sistemas

La instalación de *Call Manager Desktop* puede ser realizada vía web mediante el empleo de la tecnología Java WebStart en todos los sistemas operativos compatibles con Java. Sin embargo, el empleo de este método implica establecer varias restricciones y cuidados que deben ser tomados en cuenta para que pueda ser utilizado de forma práctica en las instituciones.

Uno de los primeros puntos a tomar en cuenta es que Java debe ser actualizado a su última versión en todos los equipos de los clientes. Otro punto importante consiste en que todos los clientes deben ser capaces de resolver el nombre de host por defecto ***callmanager*** en la dirección IP del servidor principal para que el archivo ***.jnlp*** pueda ser leído sin errores. Esta restricción es propia de la tecnología (en la que tiene que apuntarse a un URL predeterminado) y que debe ser cuidadosamente advertida por los administradores para su adecuado funcionamiento. En el caso que fuera necesario resolver otro nombre que no sea *callmanager* deberá cambiarse el URL correspondiente al campo *codebase* del archivo *CallManagerDesktop-webstart.jnlp* que se encuentra localizado en el directorio "*…/domain1/applications/CMW1/CMW1-war_war/resources/webstart/"* del servidor de aplicaciones *GlassFish*.

Adicionalmente, dentro del mismo directorio anterior, todos los elementos utilizados por el cliente de escritorio (es decir, todos los archivos *.jar* que conforman *CallManagerDesktop*) deben ser firmados digitalmente por motivos de seguridad mediante el empleo de los comandos *keytool* y *jarsigner* de Java para que éstos puedan ser descargados desde los navegadores web de los clientes. La firma de los archivos puede utilizar claves privadas o certificados de confianza según las conveniencias de las instituciones interesadas. En lo que respecta a este proyecto se han utilizado claves privadas de tal forma que la descarga del programa muestra una

advertencia de seguridad ya que los archivos están autofirmados. En la figura 3.59 se muestran los comandos utilizados para el efecto.

```
keytool -genkeypair -dname "cn=CALLMANAGER, ou=FIEE, o=EPN, c=EC" -alias CM_KEYPAIR -keypass Epn123 -keystore CM_KS -storepass Epn1234 -validity 360 -keyalg RSA -keysize 1024

jarsigner -keystore CM_KS -storepass Epn1234 -keypass Epn123 CallManagerDesktopClient.jar CM_KEYPAIR

jarsigner -keystore CM_KS -storepass Epn1234 -keypass Epn123 lib/CallManagerCommons.jar CM_KEYPAIR

jarsigner -keystore CM_KS -storepass Epn1234 -keypass Epn123 lib/javaws.jar CM_KEYPAIR

jarsigner -keystore CM_KS -storepass Epn1234 -keypass Epn123 lib/swing-worker.jar CM_KEYPAIR

jarsigner -keystore CM_KS -storepass Epn1234 -keypass Epn123 lib/swingx.jar CM_KEYPAIR

jarsigner -keystore CM_KS -storepass Epn1234 -keypass Epn123 lib/swingx-bean.jar CM_KEYPAIR

jarsigner -keystore CM_KS -storepass Epn1234 -keypass Epn123 lib/swingx-ws-2007_10_14.jar CM_KEYPAIR
```

Figura 3.59 Comandos Utilizados en la Firma Digital de *Call Manager Desktop*

Finalmente es necesario configurar la lista de sitios seguros de la pestaña *Seguridad* del *Panel de Control de Java* de cada uno de los agentes para incluir la URL del servidor de aplicaciones *GlassFish* y que de esta forma sea posible descargar los archivos firmados del programa. Se reitera que este procedimiento está descrito en la sección anexo D para más detalles.

En resumen, para la instalación de *Call Manager Desktop* mediante el empleo de Java WebStart deben cumplirse los siguientes requisitos:

- El cliente de Java del agente debe estar actualizado a la última versión.
- El computador del cliente debe ser capaz de resolver el nombre de host *callmanager* (por defecto) correspondiente al servidor principal.

- El URL del servidor de aplicaciones debe estar incluido en la lista de excepciones de sitios seguros del Panel de Control de Java del agente.
- Los archivos que conforman el aplicativo deben estar firmados digitalmente en el servidor.
- Configuración de políticas en el cortafuegos que habiliten el tráfico por el puerto 47925 (por defecto).
- Conexión a internet para la funcionalidad de georreferenciación de llamadas.

Por otro lado, la desinstalación de *Call Manager Desktop* mediante este método es realizada desde la pestaña *General→ Archivos temporales de Internet* del Panel de Control de Java de cada usuario.

3.2.2. PRUEBAS DE LA FASE DE TRANSICIÓN (TP)

Como ya fue explicado anteriormente, las pruebas de esta fase se diferencian de las realizadas en la fase de construcción en que estas últimas abarcan la comprobación de cada uno de los elementos, bibliotecas, archivos y plataformas de una manera funcional, mientras que las primeras pretenden establecer el nivel de cumplimiento de los requisitos primarios y secundarios proyectados en el capítulo 2.

Para comenzar será obligatorio diseñar métodos que permitan evaluar el cumplimiento de los casos de uso de la fase de elaboración.

3.2.2.1. Escenario de Evaluación de la Solución (TP-EV)

El sistema fue creado (diseñado y desarrollado) en Lynxsource Cía. Ltda., empresa localizada en la ciudad de Quito, orientada a brindar soluciones tecnológicas especialmente en las áreas de comunicaciones unificadas, *contact center* y *networking*. Sin embargo, las pruebas de aceptación fueron realizadas en el Laboratorio de Redes II del edificio de la Facultad de Ingeniería Eléctrica y Electrónica de Escuela Politécnica Nacional, en la ciudad de Quito. La red para la ejecución de pruebas se organizó en varias subredes separadas en VLANs de acuerdo a los servicios que prestan conforme a lo dispuesto en la tabla 3.5.

Subred	Dirección IP de Subred	Máscara
Datos	172.16.50.0	255.255.255.0
Voz	172.15.40.0	255.255.255.0
Servidores (pruebas)	192.168.1.0	255.255.255.0

Tabla 3.5 Subredes Utilizadas por *Call Manager* en el Escenario de Pruebas[20]

Como es posible apreciar en la figura 3.60, los servidores CTI y principal que pertenecen a la VLAN de pruebas poseen las direcciones IP 192.168.1.253 y 192.168.1.252 respectivamente. La central telefónica Avaya IP Office (192.168.40.251) y los teléfonos IP Avaya (192.168.40.X) pertenecen a la subred de voz de la institución. Finalmente, los computadores de los agentes se conectan a la red de datos (192.168.50.X) a través de los teléfonos IP.

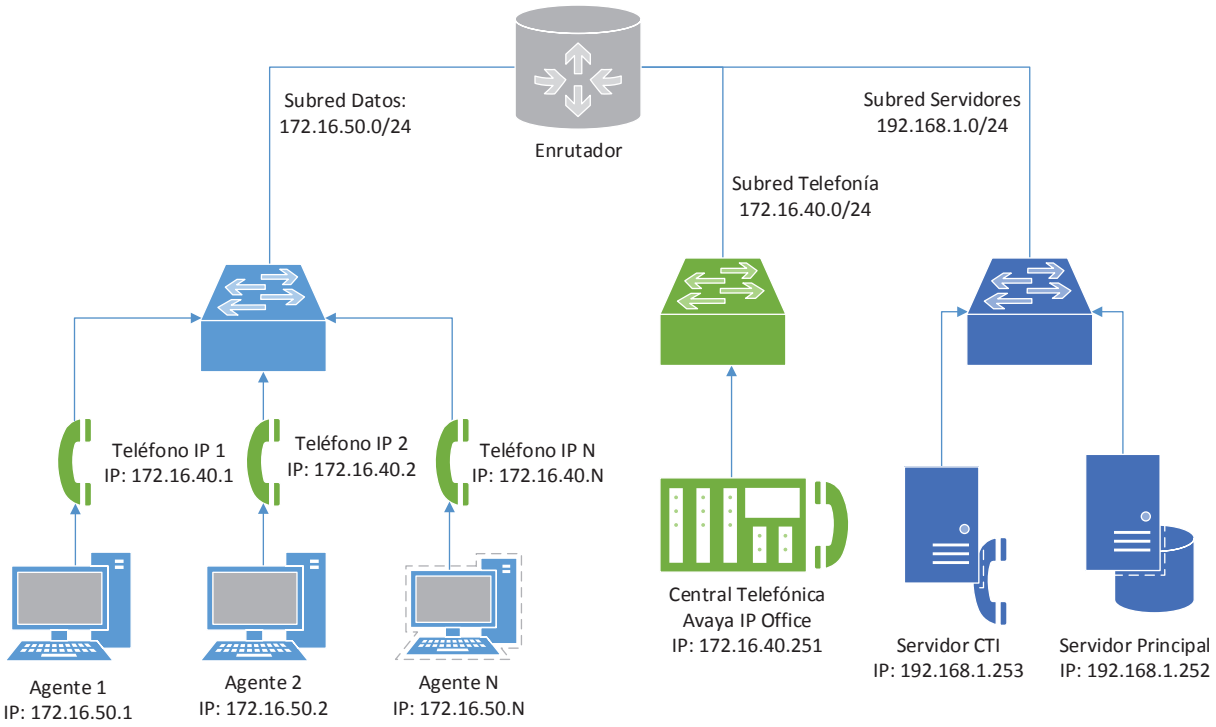


Figura 3.60 Escenario de Despliegue de la Solución

[20] Es importante aclarar que no han sido descritos los identificadores de VLAN (VID), direcciones IP de las puertas de enlace, o direccionamiento IP de administración de los *switches* de capa 2 (acceso), ya que no son relevantes para el escenario de pruebas de la aplicación a ser evaluada.

El equipo encargado de la interconexión entre las subredes es un único enrutador principal en el que están configuradas políticas y reglas que permitan el intercambio de información entre los componentes implicados sin restricciones de ningún tipo.

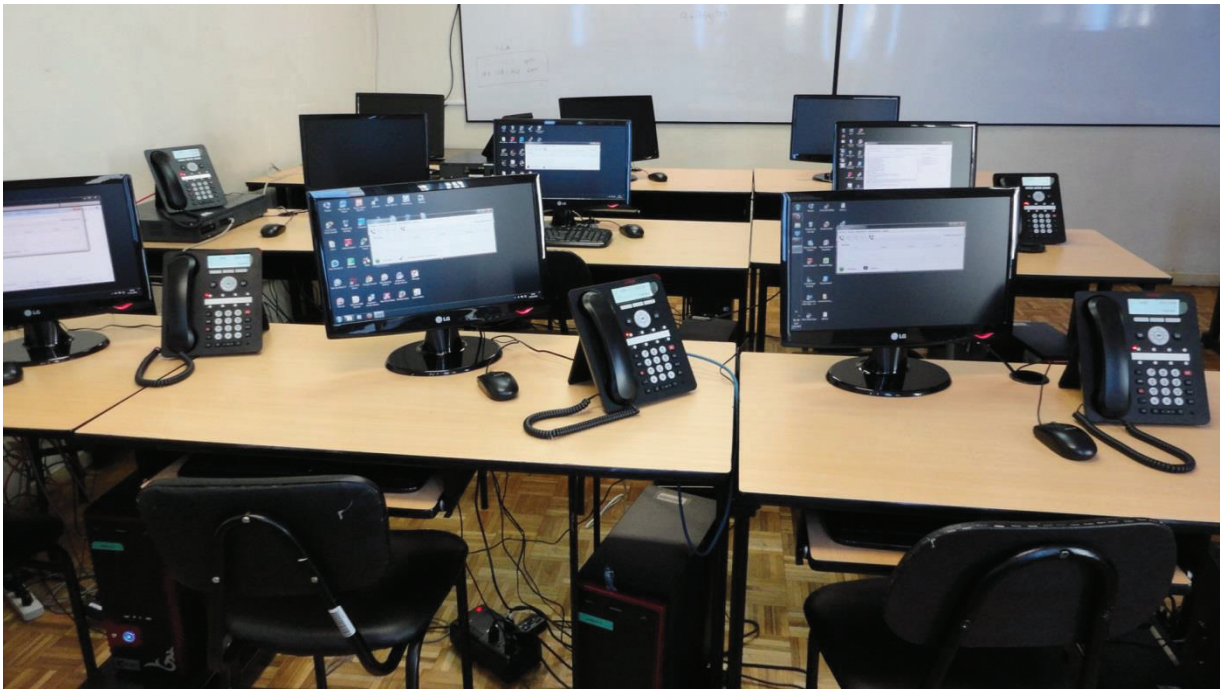


Figura 3.61 Escenario de Pruebas en la EPN

Específicamente, las pruebas se ejecutaron con 5 agentes, cada uno de los cuales disponía de un computador y su correspondiente teléfono IP Avaya 1608i, como se muestra en la figura 3.61. Entre las pruebas ejecutadas se recibieron llamadas entrantes (desde la PSTN e internas) y se realizaron llamadas salientes (hacia la PSTN y hacia otras extensiones).

En el caso de tratarse de números telefónicos fijos, se comprobó el funcionamiento del agente de escritorio y el correspondiente despliegue de la georreferenciación de las llamadas, como muestra la figura 3.62.

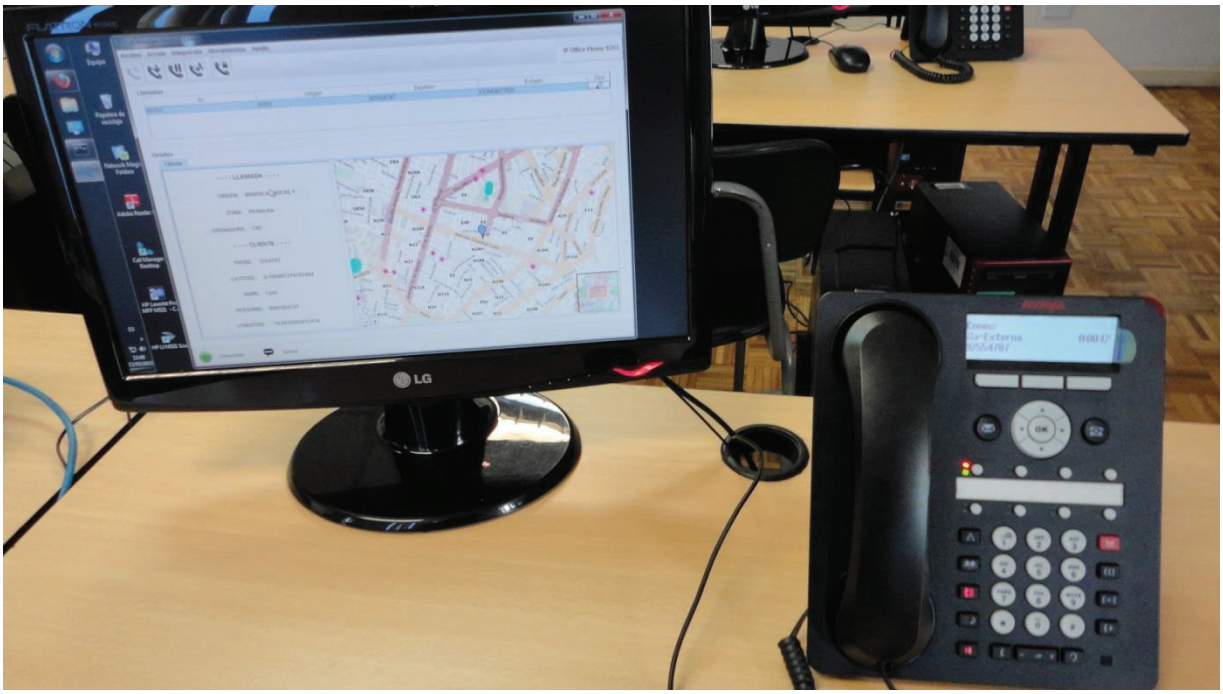


Figura 3.62 El Agente Recibe una Llamada Entrante

Las secciones TP-CR (3.2.2.2) y TP-CU (3.2.2.3) describen detalladamente las pruebas de aceptación realizadas.

Cabe aclarar que tanto el servidor principal como los computadores de los agentes poseen conexión a Internet con el propósito de obtener las coordenadas y mapas geográficos de los números telefónicos fijos para la localización de las llamadas. Con respecto a este último punto, adicionalmente han sido creadas tres bases de datos de prueba en los gestores PostgreSQL, MySQL Server y SQL Server 2008 desde los cuales se recuperarán coordenadas predefinidas asociadas a números telefónicos fijos con el propósito de comprobar la integración con bases de datos.

Finalmente, conforme a lo planteado en la sección 3.2.1.1 (TD-IPO) la central telefónica Avaya IP Office 500 V2 fue configurada por el autor conforme a las recomendaciones y lineamientos del fabricante. Dicha central dispone del licenciamiento apropiado conforme a los requisitos descritos en la disciplina de despliegue. Adicionalmente, este equipo se interconecta a la PSTN a través de una

troncal SIP que apunta a la central telefónica interna marca SIEMENS de la EPN. Para ello fue necesario configurar un *firewall* en un computador que enmascare mediante NAT estático a la central telefónica IP Office en la red interna de la universidad.

Figura 3.63 Central Telefónica Utilizada en las Pruebas

3.2.2.2. Cumplimiento General de Requisitos Primarios y Secundarios (TP-CR)

En la fase de inicio, específicamente en la disciplina de análisis (IA-CA) fueron examinados y evaluados los requisitos para la elaboración del proyecto de acuerdo a su importancia, obteniendo como resultado dos categorías divididas en requisitos primarios y requisitos secundarios. En esta sección el cumplimiento de dichos requerimientos son evaluados con el ánimo de establecer el grado de fidelidad del producto final conforme a ellos.

Así, en primer lugar se dará un vistazo general de los requisitos primarios que se encuentran en la tabla 3.6.

Código Prueba	Requisito Primario	Código Requisito	Verificación
TP-CR-P1	Utilización de TAPI de IP Office	IR-RT-T1	Cumplido
TP-CR-P2	Gestión y control de llamadas telefónicas	IR-RF-O1	Cumplido
TP-CR-P3	Despliegue de información de los contactos en una llamada	IR-RF-O2	Parcial
TP-CR-P4	Monitoreo y supervisión de agentes	IR-RF-O6	Cumplido
TP-CR-P5	Gestión de reportes y/o estadísticas	IR-RF-O7	Cumplido
TP-CR-P6	Gestionar usuarios, servicios y comunicaciones	IR-RF-O8	Cumplido
TP-CR-P7	Multiplataforma	IR-RF-O5	Parcial
TP-CR-P8	Económico	IR-NF-F1	Cumplido
TP-CR-P9	Gestión e ingreso de observaciones de las llamadas	IR-RF-O3	Cumplido
TP-CR-P10	Detalle del número telefónico de contacto en una llamada externa	IR-RF-O4	Cumplido

Tabla 3.6 Cumplimiento General de Requisitos Primarios

Tomando en cuenta que la arquitectura del proyecto tiene como base el empleo de la central telefónica Avaya IP Office 500 está por demás notar que el requisito TP-CR-P1 ha sido cumplido. En lo que respecta a la gestión y control de llamadas telefónicas (TP-CR-P2) ha sido posible generar, colgar, poner en espera, y transferir llamadas telefónicas desde el cliente de escritorio *CallManagerDesktop* con un mínimo de inconvenientes. La prueba correspondiente al requisito TP-CR-P3 relacionado con el despliegue de información de los contactos en una llamada tiene mucho que ver con la posibilidad de que las instituciones interesadas sean capaces de ajustarse a las limitaciones de *Call Manager* definidas en las secciones IA-AR-2, EAD-INF y anexo F de este documento. En tal virtud, se ha considerado que este requisito ha sido cumplido de forma parcial no sólo por la complejidad inherente en

su ámbito global, sino también por la limitada predisposición de las instituciones en cumplir con los lineamientos que permitan tal tipo de funcionalidad.

En cuanto a los requisitos TP-CR-P4 y TP-CR-P5, los supervisores del sistema son capaces de monitorear en tiempo real la actividad de las líneas telefónicas de los agentes así como también de visualizar estadísticas de operación del *call center* desde el mismo aplicativo web, razón por la cual tales requisitos han sido cumplidos a cabalidad. Sin embargo, debe notarse que por razones de tiempo no es posible crear ni exportar reportes de dichas estadísticas; funcionalidades que definitivamente pueden ser añadidas de ser el caso si se realizan las modificaciones pertinentes. La gestión de usuarios, servicios y comunicaciones es realizada por el administrador de *Call Manager* desde un navegador web, cumpliendo así el requerimiento TP-CR-P6.

Por otro lado, el requisito TP-CR-P7, multiplataforma, tiene evaluado un cumplimiento parcial debido a que el empleo de TAPI obligatoriamente limita la infraestructura de la solución al empleo de al menos un equipo que tenga como sistema operativo una versión reciente de Microsoft Windows. Debe notarse que tal limitación está analizada en la sección IA-AR-4 y que de ninguna manera interfiere con el funcionamiento multiplataforma de cualquier otro componente de *Call Manager*. A excepción de TAPI, el hecho de que los sistemas operativos, base de datos, y servidor de aplicaciones puedan ser obtenidos y empleados de forma gratuita para ejecutar *Call Manager*, hace de la solución una alternativa económica abarcando así el requisito TP-CR-P8. Tomando en cuenta que los agentes pueden ingresar en un cuadro de diálogo observaciones en el transcurso de una llamada para posteriormente ser visualizados por el supervisor hace de TP-CR-P9 un requerimiento satisfecho por el proyecto.

El último requisito primario, TP-CR-P10 atribuido a la observación de detalles del número telefónico de contacto fue minuciosamente analizado en las secciones IA-AR-3, EAD-NUM, y anexo E de este documento, haciendo de éste un objetivo cumplido por el sistema. Cabe aclarar, sin embargo, que la adecuada operación de

esta funcionalidad está optimizada para la provincia de Pichincha y únicamente para números telefónicos fijos de acuerdo a lo establecido en el alcance de este proyecto.

Por otra parte, en lo concerniente a los requisitos secundarios la tabla 3.7 muestra el grado general de cumplimiento de *Call Manager*.

Código Prueba	Requisito Secundario	Código Requisito	Verificación
TP-CR-S1	Tolerante a fallos	IR-RF-P5	Parcial
TP-CR-S2	Integración con aplicaciones externas	IR-RF-P1	Básico
TP-CR-S3	Escalable	IR-RF-P3	Parcial
TP-CR-S4	Estable	IR-RF-P6	Cumplido
TP-CR-S5	Flexible	IR-RF-P4	Cumplido
TP-CR-S6	Sencillo	IR-NF-F3	Cumplido
TP-CR-S7	Seguro	IR-NF-F4	Básico
TP-CR-S8	Atractivo	IR-NF-F2	Cumplido
TP-CR-S9	Fácil distribución	IR-RF-P2	Cumplido

Tabla 3.7 Cumplimiento General de Requisitos Secundarios

En primer lugar el cumplimiento del requisito TP-CR-S1, tolerancia a fallos, es evaluado como parcial debido a la arquitectura distribuida de la solución que implica un aumento del número de puntos de falla en el sistema, entre los cuales están: desconexión del servidor CTI (descrito y resuelto en la sección CM-CORE-Bss), desconexión de la red de un agente particular (controlado por la excepción de Java), desconexión de *Call Manager* de la base de datos local o externa, desconexión de TAPI respecto a la central telefónica, información de georreferenciación desconocida o errónea, entre otros. En lo que respecta a la integración con aplicaciones externas (requisito TP-CR-S2), en el cliente de escritorio fue añadida una única opción que permite abrir archivos ejecutables de Windows (.exe) que reciban parámetros para de esta manera transferir el número telefónico de la comunicación a otro sistema que sea capaz de buscar la información de los contactos. Cabe aclarar que esta

funcionalidad fue incluida únicamente con propósitos académicos que en la práctica no podría ser suficiente para cubrir todos los escenarios de integración con aplicaciones externas. Entre otras alternativas posibles estaría el empleo del *framework Java Naming Interface* (JNI) para interactuar con programas escritos en C o C++ ó la utilización de JavaFX para contener páginas web dentro del mismo aplicativo de escritorio.

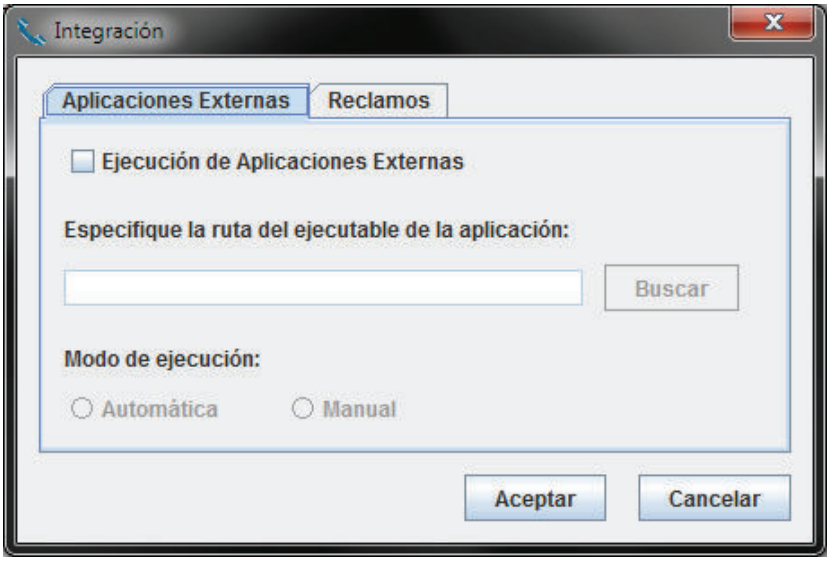


Figura 3.64 Configuración de Integración Básica con Aplicaciones Externas

En lo que respecta al requisito TP-CR-S3, escalable, éste recibió una evaluación parcial debido a que desde un punto de vista técnico, si bien el esquema distribuido de la solución hace posible añadir o remover características y funcionalidades según se convenga, todas éstas deben ser realizadas de forma manual y conforme a las posibilidades que el JDK provea. En otras palabras, el sistema es escalable acorde al portafolio de características del *framework* subyacente. Por otro lado, desde el punto de vista relacionado con el crecimiento del número de agentes o supervisores, las pruebas de campo realizadas con un número de diez agentes de escritorio demostró que el número de clientes puede ser aumentado o reducido sin problemas y que funciona con normalidad. Además de lo anterior, el hecho de que los usuarios hayan sido capaces de acceder e interactuar con el sistema por extensivos periodos de tiempo confirma el cumplimiento del requisito TP-CR-S4 (estable).

Adicionalmente, debido a que tanto el servidor principal como los clientes de escritorio fueron desarrollados con una perspectiva multiplataforma, el requisito TP-CR-S5 (flexible) es verificado positivamente. Además, tomando en cuenta que TAPI es una interfaz de programación es posible que otras centrales telefónicas distintas a IP Office compatibles con esta tecnología puedan funcionar con *Call Manager* con mínimos cambios, dando aún más validez a este requisito. Entre los fabricantes que tienen hardware compatible están Alcatel, Cisco, Panasonic, Toshiba, entre otros.

El requisito secundario TP-CR-S6, sencillo, ha sido catalogado como cumplido debido a que en las fases de elaboración e implementación se puso un adecuado énfasis en mantener un sistema fácil de utilizar así como también se prestó atención a la retroalimentación de los usuarios en la fase de transición. Por tal razón, la utilización del aplicativo de escritorio es muy intuitiva y similar a cualquier *softphone* mientras que el aplicativo web mantiene un esquema organizacional coherente y simple, motivos por los cuales el requisito TP-CR-S8 se evaluó positivamente.

Por otro lado, el requisito secundario TP-CR-S7 se consideró parcialmente cumplido ya que implementar esquemas de seguridad en todo el proyecto excede enormemente el alcance del mismo. Sin embargo, fueron tomadas medidas que previnieran ciertos problemas como son: encriptación de los mensajes de red enviados entre el servidor principal y los agentes de escritorio, encriptación de contraseñas de agentes, supervisores y administradores tanto en la base de datos local como en los archivos de configuración y firma digital del cliente de escritorio para la distribución del aplicativo de escritorio vía *Java WebStart*. Si bien las mejoras respecto a la seguridad se encuentran en el capítulo de recomendaciones, se pueden mencionar características de acceso HTTP seguro (HTTPS), encriptación de bases de datos, utilización de servidores AAA para administradores, agentes y supervisores, entre otros.

Finalmente, conforme a lo planteado en el requisito TP-CR-S9 (fácil distribución), fueron creados archivos instaladores de Windows tanto para el servidor CTI (ver TD-CTI) como para el agente de escritorio (ver TD-DSK). Adicionalmente, los agentes

pueden descargar fácilmente el programa *Call Manager Desktop* desde el navegador web mediante el empleo de *Java WebStart* desde cualquier sistema operativo. Estas razones hacen posible que el último requisito sea cumplido efectivamente.

3.2.2.3. Cumplimiento de Casos de Uso de Funcionalidades (TP-CU)

Tomando en cuenta que el refinamiento de los casos de uso de funcionalidades de la sección 2.4.1.2 (ER-RCU-F) tuvo como propósito establecer la interacción funcional de la plataforma con los usuarios (y viceversa), es necesaria una evaluación más detallada respecto al cumplimiento de los requisitos de nivel operativo del sistema *Call Manager*. En este sentido, la tabla 3.8 muestra un resumen del seguimiento de las ejecuciones de los flujos normales, alternos y excepciones de los casos de uso de la fase de elaboración.

Perfil	Código Prueba	Caso de Uso	Código C.U.	Cumplimiento Flujo		
				Normal	Alterno	Excep.
Global	TP-CU-1	Autenticar Usuario	ER-RCU-F-G1	Sí	Sí	-
	TP-CU-2	Cierre de Sesión	ER-RCU-F-G2	Sí	Sí	-
Agente	TP-CU-3	Realizar llamadas	ER-RCU-F-A1	Sí	Parcial	Parcial
	TP-CU-4	Colgar llamadas	ER-RCU-F-A2	Sí	No	Sí
	TP-CU-5	Contestar llamadas	ER-RCU-F-A3	Parcial	Sí	Sí
	TP-CU-6	Retener (sostener) llamadas	ER-RCU-F-A4	Sí	Sí	Sí
	TP-CU-7	Recuperar llamadas	ER-RCU-F-A5	Sí	-	Sí
	TP-CU-8	Transferir llamadas	ER-RCU-F-A6	Sí	Sí	Sí
	TP-CU-9		ER-RCU-	Sí	-	Parcial

		Recepción de eventos relacionados con la línea telefónica.	F-A7			
	TP-CU-10	Ingresar observaciones	ER-RCU-F-A8	Sí	Sí	Sí
Supervisor	TP-CU-11	Monitorear agentes	ER-RCU-F-S1	Sí	Sí	Parcial
	TP-CU-12	Envío de mensajes a los agentes	ER-RCU-F-S2	Sí	Sí	Sí
	TP-CU-13	Visualización de registros, reportes o estadísticas.	ER-RCU-F-S3	Sí	Sí	Parcial
Administrador	TP-CU-14	Administración de Puestos de Trabajo.	ER-RCU-F-A1	Sí	Sí	Parcial
	TP-CU-15	Administración de Usuarios.	ER-RCU-F-A2	Sí	Sí	Parcial
	TP-CU-16	Administración de Comunicaciones	ER-RCU-F-A3	Sí	Sí	-
	TP-CU-17	Administración de Planes de Numeración	ER-RCU-F-A4	Sí	Sí	Sí
	TP-CU-18	Administración de Parámetros del Sistema	ER-RCU-F-A5	Sí	Sí	Parcial
	TP-CU-19	Administración de Interconexión con Base de Datos	ER-RCU-F-A6	Sí	Sí	Sí

Tabla 3.8 Cumplimiento de Casos de Uso de Funcionalidades

De manera general, el flujo normal de eventos de todos los casos de uso de funcionalidades es cumplido por *Call Manager* conforme a lo esperado, con la excepción de la prueba TP-CU-5 (contestar llamadas), en la que se pudo comprobar que sólo es posible contestar una llamada a través del altavoz del teléfono (*speaker*)

para equipos sólo de la marca Avaya. Esta limitación es propia del controlador TAPI del fabricante, lo que implica que sólo es posible contestar llamadas telefónicas mediante el altavoz utilizando *Call Manager* en equipos compatibles con la central telefónica en cuestión.

Por otro lado, la prueba TP-CU-3, relacionada con la realización de llamadas, tiene cumplimientos parciales en los flujos alternos y excepciones debido a que no fueron considerados todos los casos de verificación de números telefónicos válidos que sean compatibles con los planes de numeración configurados en la central telefónica. Dicho de otra forma, la variabilidad del formato de marcación definido por el administrador del sistema de telefonía en cada organización es el motivo por el cual se dificulta establecer un sistema general de verificación de números válidos en el aplicativo de escritorio de *Call Manager*.

Adicionalmente, no fue posible reproducir un escenario en el que ocurriera el flujo alterno para la prueba TP-CU-4, colgar llamadas, razón por la cual tiene un estado de no cumplimiento. Además, el flujo de excepciones de los casos de uso TP-CR-9, 11, 13, 14, 15 y 18 fue evaluado como parcial debido a que no fueron incluidos todos los escenarios de desconexión de los componentes del sistema según lo descrito en la sección anterior para la prueba general TP-CR-S1.

Finalmente, y de acuerdo a la información desprendida de las pruebas realizadas, se puede afirmar que el sistema de gestión de llamadas *Call Manager* cumple positivamente con los objetivos planteados en las fases de inicio y elaboración, haciendo de ésta una herramienta eficaz que pueda ser utilizada en los *call centers* de las instituciones interesadas.

CAPÍTULO IV

4. ANÁLISIS COMPARATIVO

4.1. COSTO DEL PROYECTO

En esta sección se realizará un análisis cuantitativo aproximado del costo de desarrollo del Proyecto de Titulación para posteriormente compararlo con alternativas funcionalmente similares en el mercado. Cabe aclarar que en este apartado no se realizará un exhaustivo análisis financiero, ya que no es el tópico principal del Proyecto ni tampoco pretende exceder el alcance inicial definido en la aprobación del mismo. Por lo anterior, los valores aquí presentados solamente sirven como referencia para determinar inicialmente el grado de comercialización del software como producto para pequeñas y medianas empresas en el país.

Tomando en cuenta lo mencionado, para la determinación del coste del software creado se tomarán en cuenta dos enfoques distintos: el primero tendrá una perspectiva teórica desde el punto de vista de un supuesto gerente del proyecto que efectuaría una estimación aproximada de éste en la fase de inicio, específicamente en el análisis de riesgos. El segundo, en cambio, describirá el costo real de la aplicación al haberse finalizado su respectivo desarrollo tomando en cuenta los recursos prácticos invertidos en el mismo.

4.1.1. ENFOQUE TEÓRICO: ESTIMACIÓN MEDIANTE EL MODELO CONSTRUCTIVO DE COSTOS [18]

De entre las varias alternativas recomendadas para la estimación de costos de software, el modelo COCOMO I (*COnstructive COst MOdel*) se destaca por ser una opción sencilla y ampliamente utilizada, propuesta en el año 1981 por Barry W. Bohem. Este modelo utiliza una base empírica basada en el estudio de 63 proyectos de software, desarrollados principalmente mediante el empleo de un proceso en cascada para aplicaciones que oscilaban entre 2.000 y 100.000 líneas de código. Como resultado, fueron publicados tres submodelos, cada uno de los cuales está

dirigido a entornos que requieran distintos niveles de detalle según su complejidad, pudiendo ser: básico, intermedio y detallado. En lo que respecta a este proyecto, el modelo COCOMO Básico es más que suficiente para definir de forma general el coste del sistema; las ecuaciones correspondientes a este modelo son las siguientes:

1. $E = a \times KLOC^b$
2. $T_{dev} = c \times E^d$
3. $P = E/T_{dev}$

Donde se define:

- *E* es el esfuerzo requerido por persona al mes.
- *KLOC* es el número (en miles) de líneas de código estimadas.
- T_{dev} es el tiempo de desarrollo del proyecto en meses.
- *P* es el número de personas requeridas.
- Las constantes *a, b, c,* y *d* se presentan en la tabla 4.1.

Adicionalmente, COCOMO define varios modos según los cuales las constantes *a, b, c* y *d* varían conforme a los proyectos. En este documento se utilizará la tabla de constantes correspondiente al modo *orgánico* enfocado a un pequeño grupo de programadores y con un tamaño del software de hasta decenas de miles de líneas de código conforme a la tabla 4.1. Los modos semi-orgánico y empotrado están dirigidos a aplicaciones más complejas en las que los equipos de desarrollo tienen experiencia limitada o donde las aplicaciones están sujetas a reglas y restricciones específicas, que si bien comparten características con el software desarrollado se han considerado menos apropiadas.

MODO	a	b	c	d
Orgánico	2.40	1.05	2.50	0.38
Semi-orgánico	3.00	1.12	2.50	0.35
Empotrado	3.60	1.20	2.50	0.32

Tabla 4.1 Valores de Constantes para los Modos del Modelo COCOMO [18]

En tal sentido, los resultados de reemplazar los valores en las ecuaciones son los siguientes:

1. $E = (2.40) * (10)^{1.05} = 26.928\ persona/mes$
2. $T_{dev} = (2.50) * (26.928)^{0.38} = 8.738\ meses$
3. $P = 26.928/8.738 = 3.08\ personas$

El valor del campo KLOC ha sido establecido de manera empírica tomando en cuenta que se trata de un sistema distribuido multilenguaje, por lo que un estimado de 10.000 (diez mil) líneas de código es bastante probable en este escenario. Como resultado, el método COCOMO sugiere el empleo de 3 personas durante un lapso de 8 meses, según lo cual se establecerá la tabla final 4.2 de determinación del costo estimado del proyecto desde la sección de análisis de riesgos de la fase de inicio.

Concepto	**Costo Mensual (USD)**	**Instancias**	**Tiempo (meses)**	**Costo Final (USD)**
Salario mensual programador	$1.000	3	8	$24.000
Recursos logísticos (servicios básicos, internet, etc.)	$72	1	8	$576
Arriendo	$250	1	8	$2.000
Otros (manuales, libros, gastos de oficina, etc.)	-	-	-	$2.000
			Total sin IVA	**$28.576**
			TOTAL con IVA (12%)	**$32.005,12**

Tabla 4.2 Costo Estimado del Proyecto – Modelo COCOMO

Debe notarse que en la columna *Concepto* únicamente han sido tomados en cuenta los principales factores inherentes en la elaboración de *Call Manager,* sin tomar en cuenta costos adicionales externos como la adquisición del sistema de telefonía Avaya IP Office, licenciamiento específico CTI o de usuarios, o costos administrativos, infraestructura física, equipos de trabajo, patentes y publicidad del producto. Estos elementos forman en sí parte de la gestión de las empresas

desarrolladoras de software cuyo dominio no será incluido en este análisis ya que exceden el alcance del proyecto.

Sin embargo, el valor total de $32.005,12 definitivamente serviría como un punto de referencia válido para administradores e inversores, el cual ayudaría a determinar la probabilidad de proseguir con las fases siguientes del proceso unificado; es decir, sería uno de los ejes que daría luz verde a las fases de elaboración, construcción y transición de *Call Manager*.

4.1.2. ENFOQUE PRÁCTICO: COSTOS FIJOS

Si bien en el apartado anterior se determinó de manera aproximada el costo que tendría el desarrollo de la aplicación conforme a los métodos y recomendaciones de estimación de costos de software, la realidad es que el sistema fue creado por un solo programador (el autor) y en condiciones particulares que no pueden ser obviadas por las empresas de desarrollo de software. Es por ello que en este apartado se detallará de forma cuantitativa y con base en la experiencia real el costo final del proyecto.

Componente *Call Manager*	Nº Líneas Totales	% Certidumbre	Nº Líneas Efectivas
CTI	4.417	50	2.209
Commons	1.464	45	658,8
Desktop	4.929	50	2.464,5
CMW1-ejb	7.576	50	3.788
CMW1-war	7.032	80	5.625,6
		Total	**14.745,4**

Tabla 4.3 Número de Líneas de Código Efectivas de *Call Manager*

Para empezar, el número de líneas de código totales escritas fue determinado con el ánimo de comprobar el grado de certeza con el campo KLOC planteado en el modelo COCOMO. La tabla 4.3 describe el número aproximado de líneas que componen *Call Manager*. Se entiende por líneas de código efectivas a aquellas que tienen relación

directa con el funcionamiento de la aplicación y que han sido escritas por el programador. Esto quiere decir que los espacios en blanco, los comentarios, y el contenido autogenerado no debe ser incluido en el conteo. Por tal razón, y dado el extenso número de líneas de código de *Call Manager*, se ha decidido establecer un porcentaje de certidumbre para cada módulo del sistema cuyo propósito es determinar de manera aproximada el número total de líneas de código efectivas en cuestión. Al final se ha determinado un aproximado de 14.745 líneas efectivas que sobrepasan a las inicialmente planteadas en el enfoque teórico (10.000).

En segundo lugar se detallará el costo real de la aplicación tomando en cuenta que tiempo de desarrollo de la misma fue de **15** meses desde la selección de la metodología en la fase de inicio. La tabla 4.4 describe a detalle los gastos incurridos en la creación del sistema propuesto.

Concepto	Costo Mensual (USD)	Tiempo (meses)	Costo Final (USD)
Salario mensual programador	$1.000	15	$15.000
Costo servicios básicos (agua, luz eléctrica, teléfono)	$30	15	$450
Internet	$25	15	$375
Papelería	$10	15	$150
Otros	$10	15	$150
		Total sin IVA	**$16.125**
		Total con IVA (12%)	**$18.060**

Tabla 4.4 Costos Fijos de Elaboración de *Call Manager*

Debe notarse que la tabla no incluye gastos de alquiler o limpieza debido a que el sistema se realizó en el hogar del autor, las instalaciones de Lynxsource Cía. Ltda., y los laboratorios de la Escuela Politécnica Nacional. De igual manera, la central telefónica, sus respectivas licencias y la documentación de referencia fueron

proporcionadas por la empresa patrocinadora antes mencionada (Lynxsource) sin ningún cargo adicional para el autor, razón por la cual estas variables no fueron incluidas en el análisis de costos de la solución.

Ahora bien, el siguiente paso consiste en hacer de *Call Manager* una solución comercial. Tomando en cuenta un margen de rentabilidad del 15% sobre el costo de elaboración del proyecto, el monto final asciende a **$20.769**. Este último valor es el que será utilizado para determinar el costo que será asociado a las instituciones compradoras de la solución. Para ello, se tomará en cuenta una base de potenciales clientes de 8 instituciones, lo que a su vez representa un valor de **$2.596,125** por cada organización que desee instalar *Call Manager* como su plataforma de gestión de *call center* para Avaya IP Office. El análisis comparativo de la sección 4.3 de este capítulo determinaría que en el mercado ecuatoriano no existen soluciones de georreferenciación de llamadas en el país, por lo que el número de clientes potenciales podría ser mayor a los ocho inicialmente propuestos, lo que resultaría en un precio por institución menor al anteriormente señalado.

Es importante aclarar que el costo del software por institución calculado no incluye el costo del hardware subyacente en el que opera, ya que por lo general las instituciones deciden adquirir dicha infraestructura con proveedores aliados o incluso reutilizar hardware existente subutilizado.

4.2. OTRAS SOLUCIONES DE GESTIÓN DE *CALL CENTERS*

El reto al que se enfrenta *Call Manager* en lo que se refiere a una evaluación técnica y económica está determinado por el conjunto de funcionalidades comunes con respecto a otras alternativas comerciales. Luego de una extensiva búsqueda de aplicaciones comerciales de geolocalización de llamadas telefónicas en el país se ha llegado a la conclusión de que únicamente es posible encontrar sistemas propietarios orientados a sistemas de emergencia, especialmente dirigidos a la provincia del Azuay. Esto significa que existe una oportunidad de mercado enorme que puede ser aprovechada por *Call Manager* como el primer intento comercial de

georreferenciación de llamadas telefónicas para pequeñas y medianas empresas en el Ecuador.

Sin embargo, es importante señalar que *Call Manager* también incluye características que pueden ser encontradas en otros sistemas de gestión de *Call Centers*, cuya comparación permite establecer una línea base de complejidad, calidad, desempeño, y conveniencia que le permitirá definir sus fortalezas y debilidades con respecto a otras soluciones. En tal virtud, los sistemas de gestión de *call centers* utilizados para la evaluación comparativa de *Call Manager* serán *Avaya IP Office Contact Center* y *Evolution Contact Center*.

4.2.1. IP OFFICE *CONTACT CENTER*

Provisto por la empresa Avaya, esta solución engloba una gran cantidad de funcionalidades de gestión de *call centers* con el afán de ofrecer un amplio conjunto de interacciones entre los agentes y los clientes de las instituciones. Al ser desarrollado y mantenido por la empresa Avaya, la aplicación tiene una total compatibilidad con la plataforma de comunicaciones IP Office V2 500 utilizada en este proyecto.

Esta solución está orientada a pequeñas y medianas empresas que requieren canales de comunicación de voz, correo electrónico, chat vía web, así como también administrar por completo el ciclo de vida de interacción con los clientes. Entre las principales características de la herramienta están [19]:

- **Flexible distribución de medios.** Optimiza los procesos de negocios mediante la distribución de contactos hacia los empleados basados en procesos previamente definidos.
- **Integra soluciones multicanal para video, e-mail y chat vía web.** De esta forma los clientes son capaces de elegir el canal de comunicación que consideren más conveniente.
- **Priorización de clientes.** Con base en las estrategias de negocios es posible establecer prioridades de atención de los clientes.

- **Enrutamiento basados en habilidades.** Las llamadas son enrutadas al empleado que está mejor calificado para atender la comunicación con base en el tipo de canal, pericia y experiencia.
- **Soluciones de auto-servicio.** El cliente es capaz de trabajar sus inquietudes por sí mismo reduciendo la carga de trabajo en los empleados.
- **Campañas de llamadas salientes y *telemarketing*.** Proporciona estrategias de generación de llamadas telefónicas o de correo electrónico masivo para la provisión de oportunidades de mercado.
- **Monitoreo en línea.** Provee información en tiempo real de las operaciones del *call center*.
- **Reportes históricos.** Proporciona información de operación del *call center* sobre prolongados períodos de tiempo.
- **Respuesta de voz interactiva (IVR).** Sistemas automatizados de voz.
- **Grabación de llamadas.** Funcionalidad utilizada para propósitos de entrenamiento o de resolución de conflictos.

Es importante notar que esta solución es adquirida de forma separada respecto a la plataforma básica de comunicaciones Avaya IP Office, y que para su despliegue son necesarios ciertos requisitos iniciales (conforme al **Anexo G**):

- **IP Office *Contact Center***
    - *Sistema Operativo*: Microsoft ® Windows Server 2008 R2 Standard 64–bit Edition SP1, con licencia válida. - Microsoft ® Windows Server 2012 R2 Standard 64–bit Edition, con licencia válida.
    - *Procesador*: Intel Xeon E3 Quadcore de 3.1 GHz
    - *Memoria*: RAM 8 GB
    - *Disco Duro*: 250 GB
    - *Media*: DVD-ROM drive
    - *Red*: 1 tarjeta de red de 1 GB
    - También es Soportado en un entorno de servicio virtualizado como VMware con las características anteriores.

- **Contact Recorded (Sistema de Grabación)**
  - *Sistema Operativo*: Microsoft ® Windows Server 2008 R2 Standard 64–bit Edition SP1, con licencia válida ó Microsoft ® Windows Server 2012 R2 Standard 64–bit Edition, con licencia válida.
  - *Procesador*: Intel Dual Core de 2 GHz
  - *Memoria*: RAM 4 GB
  - *Disco Duro*: 250 GB
  - *Media*: DVD-ROM drive
  - *Red*: 1 tarjeta de red de 1 GB
  - También es Soportado en un entorno de servicio virtualizado como VMware con las características anteriores.

- **Estaciones de trabajo (Agentes/Supervisor)**
  - *Sistema Operativo*: Microsoft® Windows 7, o Windows 8.1
  - *Procesador*: Intel Pentium 4, 2.2 GHz o superior.
  - *Memoria*: RAM 4 GB
  - *Disco Duro*: 20 GB
  - *Buscador*: Microsoft ® Internet Explorer 8.x o superior; Mozilla Firefox 3.6 o superior.
  - *Red*: 1 tarjeta de Red

Adicionalmente, las recomendaciones publicadas por el fabricante exponen la utilización de teléfonos de características avanzadas (serie IP Phone 9608G) y diademas con cancelación de ruido en cada uno de los agentes/supervisores que componen el *call center*.

Finalmente, la información a detalle de las licencias y los equipos terminales para un *call center* de cinco agentes está indicada en la proforma del sistema Avaya IP Office *Contact Center* Multicanal en el Anexo G de este Proyecto de Titulación.

4.2.2. *EVOLUTION CALL CENTER*

"*Evolution es la solución de software diseñada para la automatización y gestión eficiente de los Contact Centers, con la relación precio/rendimiento más competitiva de la industria*." [E-10]

Esta solución tiene respaldo de Avaya ya que *Evolution* es *partner* del programa *Avaya Devconnect*, el cual ofrece a sus miembros soporte en el desarrollo de aplicaciones para productos de la marca Avaya, así como también facilita pruebas de aceptación y compatibilidad de productos desarrollados por terceros.

Entre las características principales de *Evolution Call Center* están [E-10]:

- **Operación y Gestión de *Call Centers*.** Las prestaciones de Evolution son presentadas a los usuarios de una manera sencilla e intuitiva, minimizando el tiempo de implementación y aprendizaje de los usuarios, incrementando así la productividad de los agentes.
- **Gestión de llamadas entrantes.** Que incluye la grabación de llamadas, IVR, supervisión, informes e integración con herramientas como CRMs, ERPs, etc.
- **Enrutamiento Dinámico de Negocios.** Distribución de interacciones multicanal conforme a los términos de negocio y enrutamiento basados en habilidades.
- **Marcación Saliente.** Generación de campañas de llamadas salientes de forma automática.
- **Multimedia.** Permite la construcción de aplicaciones para la atención en canales múltiples de voz, correos electrónicos, *tweets*, *posts* en Facebook, entre otros.
- **Integración de Aplicaciones.** Proporciona APIs para la integración de aplicaciones ActiveX, *scripting*, XML, bases de datos, ASP.NET, PHP, servicios web, entre otros.
- **Compatibilidad.** *Evolution* es compatible con una amplia variedad de centrales telefónicas, entre ellas Avaya IP Office 500 V2.

Los requisitos definidos para esta solución son los siguientes [E-11]:

- Avaya IP Office R7.0.23 o superior.
- TAPI3 Service Provider (TAPI).
- Teléfonos Avaya 96XX H.323 *Deskphone*, Avaya 2420.
- *Evolution* 10.1 o superior.
- *Servidor*: Windows Server 2003/2008/2012 32/64 bits.
- *Cliente*: Windows Vista, Windows 7, Windows 8 o XP, IE 9+.
- *Bases de Datos*: MS SQL Server 2005, 2008, 2012 (Incluyendo *Express Edition*).

Adicionalmente, *Evolution* se presenta en dos versiones:

- ***Enterprise*** es la solución profesional, con todos los servicios necesarios de soporte y mantenimiento, para sacar la máxima productividad al *Contact Center*. Esta edición está dirigida a empresas que no se la juegan y buscan una solución robusta, potente, flexible y escalable con la mejor relación precio-beneficio de la industria.
- ***Community*** es la versión gratuita, limitada a 10 usuarios, y con todas las funcionalidades necesarias para poner en producción y operar un *Contact Center* pequeño. *Evolution Community* dispone de un foro de soporte a la instalación, FAQ y toda documentación necesaria para su despliegue. Esta edición está pensada para aquellas empresas que quieren probar la solución o empezar a operar sin asumir ningún compromiso con el proveedor. Inicialmente está limitado a 10 agentes simultáneos, siendo la alternativa utilizada para la comparación en este proyecto.

4.3. ANÁLISIS COMPARATIVO

4.3.1. TÉCNICO-FUNCIONAL

Considerando las dos alternativas de gestión de *call centers,* en esta sección se realizará un análisis a nivel técnico y se determinarán las diferencias entre *Call Manager* y las aplicaciones empresariales profesionales.

La tabla 4.5 provee una lista de características comunes que tienen *Avaya IP Office Contact Center* y *Evolution Call Center* con la solución creada en este proyecto. Nótese que ninguna de ellas proporciona funcionalidades de georreferenciación de llamadas telefónicas (objetivo principal de *Call Manager*) y menos aún soporte específico para el entorno ecuatoriano. De manera similar, *Call Manager* no posee funcionalidades de IVR, campañas de llamadas salientes, o gestión de comunicaciones multicanal, al no ser parte del alcance ni de los objetivos de este proyecto.

Característica\Solución	***Call Manager***	***Avaya IP Office Contact Center***	***Evolution Call Center***
Gestión del estado del teléfono de los agentes	Sí	Sí	Sí
Georreferenciación de llamadas telefónicas	**Sí**	**No**	**No**
Autenticación de usuarios y perfiles	Sí	Sí	Sí
Ingreso de observaciones de las llamadas	Sí	Sí	Parcial
Administración vía Web	Sí	Sí	Sí
Monitoreo en tiempo real de los agentes	Sí	Sí	Sí
Visualización de registros, reportes o estadísticas	Sí	Sí	Sí
Envío de mensajes a los agentes	Sí	Sí	No
Interconexión con bases de datos externas	Sí	Sí	Sí
Soporte de PBXs de otros fabricantes	Parcial	No	Sí
Integración con aplicaciones externas	Parcial	Sí	Sí

Atractivo y de fácil uso	Sí	Sí	Sí
Mapas de calor para llamadas entrantes y salientes	**Sí**	**No**	**No**
Compatibilidad de acuerdo a los planes de numeración del Ecuador	**Sí**	**No**	**No**
Reporte de llamadas por operadora	**Sí**	**No**	**No**
Multiplataforma	Parcial	No	No
Respuesta de voz Interactiva	No	Sí	Sí
Grabación de llamadas	No	Sí	Sí
Campañas de llamadas salientes	No	Sí	Sí
Contacto multicanal con los clientes (e-mail, chat web, etc.)	No	Sí	Sí
Autoservicio	No	Sí	No
Enrutamiento basado en habilidades	No	Sí	Sí

Tabla 4.5 Cuadro Comparativo de Alternativas de Gestión de *Call Centers*

Un estudio de la tabla anterior demuestra que la principal fortaleza de *Call Manager* reside en su adaptación al mercado ecuatoriano que le permite destacarse sobre aplicaciones comerciales. A la vez, la herramienta desarrollada no pretende competir con otras en el campo específico de gestión avanzada de *call centers*, específicamente en las áreas de generación de campañas de llamadas, enrutamiento basados en habilidades, IVRs, sistemas de grabación de llamadas, contacto multicanal, entre otros, ya que jamás se plantearon tales metas en fases iniciales de creación de la aplicación.

La selección de una alternativa sobre otra a nivel técnico, por tanto, deberá estar sujeta a las necesidades y objetivos específicos de cada institución, incluso pudiendo coexistir *Call Manager* junto a otras soluciones de *call center* en caso de ser necesario sin superponer funcionalidades entre ellas.

4.3.2. ECONÓMICO

Para el análisis económico se tomarán en cuenta ciertos factores importantes:

- No será incluido en el análisis el licenciamiento específico requerido en la central telefónica para el funcionamiento normal del *call center*. En otras palabras, el dimensionamiento apropiado del sistema de telefonía es responsabilidad de las instituciones y no está directamente relacionado con el costo del software de gestión del *call center*.
- Tampoco se comprenderán a un nivel económico los requerimientos de hardware y software adicional específicos sobre los cuales operan las soluciones de gestión de software. Esta decisión se debe a que es posible distribuir o agrupar los componentes de las aplicaciones en análisis tanto en máquinas físicas como virtuales, cuya variabilidad dificulta la evaluación de una línea base común. Además, según lo estipulado en la sección 4.1.2, las instituciones interesadas en la adquisición de estas herramientas pueden decidir reutilizar hardware y software existente con poco o ningún uso o incluso adquirirlo a proveedores estratégicos.

Considerando lo anterior, y tomando en cuenta la información obtenida en la sección 4.1.2, la proforma obtenida de Avaya IP Office *Contact Center*, y la información de la versión *Community* de *Evolution*, fueron obtenidos los siguientes costos de despliegue:

- ***Call Manager*.** En la sección 4.1.2 mediante un enfoque práctico de costos fijos se calculó un precio base para las instituciones de $2.596,125 habiéndose considerado 8 posibles clientes con un margen de rentabilidad del 15%. A este valor debe añadirse un costo de instalación que se calculará mediante el producto del número de horas de despliegue por el costo de la mano de obra. Este último utiliza como referencia el salario definido en la sección 4.1 ($1.000) dividido para el número de horas por mes (30), resultando en 33,33 por hora y redondeado a $35 dólares. Se calcula que *Call Manager* requiere de 3 horas para su instalación, por lo que el costo del

sistema para cada institución asciende a **$2.713,723** dólares (incluido IVA). Cabe precisar que, con la excepción del servidor CTI que puede ser instalado en cualquier equipo Windows, todo el software relacionado (sistema operativo, servidor de aplicaciones y base de datos) puede ser obtenido de manera gratuita, reduciendo de forma considerable los costos de despliegue de la solución que incluso puede ser virtualizada o agrupada en un único servidor físico, haciendo de *Call Manager* una solución conveniente considerando el conjunto único de funcionalidades que ofrece para el mercado ecuatoriano.

Además es importante notar que no es necesario utilizar teléfonos especiales o diademas para el funcionamiento normal del *call center*. Dicho de otro modo, la institución únicamente pagaría por el software de gestión y su correspondiente instalación sin incurrir en gastos adicionales.

- **Avaya IP Office *Contact Center*.** Al ser una aplicación propietaria, los clientes de este sistema están cubiertos en temas de garantía y soporte, razones por las que el costo de un sistema de este tipo suele ser mayor que el de alternativas más pequeñas. Tomando en cuenta como referencia la proforma obtenida en el **Anexo G** para el despliegue básico de *IP Office Contact Center*, el costo total por institución equivale a **$8.579,20** dólares (incluye IVA). En esta proforma se incluyen los costos de los equipos terminales (IP Phone 9608G) y diademas para 5 agentes de *call center* requeridos por la solución. Habrá que tomar en cuenta que los sistemas de *contact center*, grabación y estaciones de trabajo, todos requieren de sistemas operativos Windows 7, 2008 o superior.

- ***Evolution Call Center*.** Si bien la versión *Community* se entrega de forma gratuita mediante el llenado de un formulario, no se cuenta con documentación avanzada, garantía, o soporte por parte del proveedor. Esto significa que las instituciones tendrán que invertir recursos propios en el estudio, diseño, despliegue, y pruebas de esta versión. Adicionalmente, tanto

el servidor como los clientes deben contar con el sistema operativo Windows para su operación, así como también son necesarios teléfonos Avaya de la serie 96XX, de forma similar a la opción anterior de Avaya. Esto implica que es necesario incurrir en gastos adicionales específicos para que el sistema pueda funcionar apropiadamente. Considerando únicamente los costos de los teléfonos (junto a las diademas y los adaptadores de energía) para 5 agentes el monto inicial es de **$2.005,00** (incluido IVA).

Finalmente, es necesario recalcar que si bien las tres soluciones comparadas poseen ciertas características comunes, las funcionalidades específicas de *Call Manager* hacen que ésta sea una alternativa válida para casos concretos en los que la georreferenciación de llamadas sea prioritaria y que no necesariamente requieran de funcionalidades completas de gestión de *call centers*. Aún bajo estas condiciones se pudo notar que el sistema puede coexistir con otras soluciones, puesto que no pretende cubrir otras áreas que ya han sido cubiertas por otras alternativas. Además, el precio del producto por institución está a la par (en cuanto a rango de precios se refiere) con alternativas comerciales que le permiten ser asequible para pequeñas y medianas empresas del mercado ecuatoriano.

CAPÍTULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

- La creación de un software de gestión de *call centers* orientado a la georreferenciación de llamadas telefónicas para el Ecuador inicialmente fue concebida como una idea ambiciosa que pretendía cubrir la mayor cantidad de escenarios posibles. Con el avance del desarrollo del proyecto fue posible notar que se necesitaron límites para que la solución compartiera los objetivos y requerimientos planteados en la fase de inicio, lo que a su vez significaría que el producto final funcione bajo ciertas circunstancias específicas. Esto se vuelve especialmente cierto en lo correspondiente a los escenarios de integración con bases de datos externas, en las que la mejor alternativa consistió en la lectura (no escritura) de la información de una única tabla o vista para recuperar la información de los contactos de una llamada. Adicionalmente, el hecho de que haya sido utilizada la central telefónica IP Office de Avaya para la creación del software implicó la utilización de TAPI, razón por la cual el sistema no pudo ser completamente multiplataforma como habría sido lo ideal.

- Sin embargo, los resultados obtenidos en la disciplina de pruebas de la fase de transición permitieron concluir que el sistema de gestión de llamadas de *call centers Call Manager* cumplió positivamente las expectativas y requerimientos planteados en las fases de inicio y elaboración, haciendo de esta solución una alternativa por demás conveniente para los sectores en los que se requiera establecer rápidamente la localización geográfica de las llamadas telefónicas como son: servicios de emergencia, bomberos y policía, transporte (taxis por ejemplo), entrega a domicilio, entre otros.

- Tomando en cuenta que en el mercado ecuatoriano para pequeñas y medianas industrias no existen soluciones de propósito general que permitan georreferenciar llamadas telefónicas, *Call Manager* constituye el primer intento de ofrecer una solución uniforme y general que se adapte no sólo a *call centers* que emplean la plataforma Avaya IP Office sino también a aquellos en los que operan centrales telefónicas compatibles con TAPI con mínimos cambios. Además es importante notar que la naturaleza distribuida de la solución permitiría crear módulos o servidores CTI para cualquier tipo de central telefónica (que no necesariamente adopte TAPI) y mantener la operación de *Call Manager* siempre y cuando se utilice el protocolo de comunicaciones detallado en la fase de diseño del proyecto, cumpliendo así uno de los requisitos secundarios relacionados con la flexibilidad del sistema de adaptación a otras tecnologías de comunicaciones.

- Uno de los mayores inconvenientes en la elaboración de sistemas distribuidos recae en el aumento del número de posibles puntos de falla que pueden afectar la operación del sistema. Si bien se trató de que el software fuera tolerante a fallos, por razones de tiempo y de recursos no fue posible abarcar todos los escenarios de error que pueden ocurrir en la operación no sólo de *Call Manager* sino también de la central telefónica, bases de datos, o teléfonos y computadores de los agentes. Sin embargo, las pruebas realizadas determinaron que bajo condiciones normales de operación el sistema trabaja de forma estable de acuerdo a lo esperado durante prolongados períodos de tiempo.

- Por otro lado, el seguimiento de la metodología de desarrollo de software Proceso Unificado permitió crear una solución de manera profesional de principio a fin, siempre tomando en cuenta las entradas, requisitos y análisis en cada fase del proyecto que maximizarían el éxito del mismo. Entre sus desventajas está el extensivo uso de la documentación y sus

correspondientes referencias que obligaron extender los plazos temporales de desarrollo del sistema. En tal virtud, ha sido posible concluir que la metodología elegida es más apropiada en entornos de trabajo multidisciplinarios donde es posible distribuir tareas y roles a los actores del proyecto. Lo anterior toma mayor relevancia en circunstancias que requieren experiencia en desarrollo en múltiples lenguajes de programación como C++, Java ó HTML, habiendo sido éste el caso (si bien todo fue elaborado por el autor).

- Aunque el despliegue de los aplicativos de escritorio de los agentes es realizado de manera sencilla, los procedimientos previos de instalación de los componentes principales pueden parecer complicados para individuos ajenos a temas relacionados con telefonía, aplicaciones distribuidas, o *call centers*, debido a la arquitectura propia de la solución. No sólo deben ser analizados planes de numeración o permisos de marcación en la central telefónica, sino también tienen que considerarse parámetros de licenciamiento, conectividad de red, configuraciones de bases de datos, o actualizaciones de software en los computadores de las instituciones para que *Call Manager* pueda cumplir con los objetivos propuestos.

- En este sentido, la naturaleza distribuida del sistema permite añadir módulos y funcionalidades que puedan expandir los alcances iniciales de la solución. Para ejemplificar, sería posible extender la funcionalidad de georreferenciación de llamadas a teléfonos inteligentes (*smartphones*) mediante la instalación de un aplicativo dedicado en los teléfonos móviles que alimente un módulo de *Call Manager* que puede ser consultado por el EJB *EventController* en el transcurso de la llamada, o también podría publicarse un nuevo EJB sin estado como servicio web JAX-RPC (la plataforma de Java lo permite) que se interconecte a las funciones de *MainController* para recuperar el estado de las líneas telefónicas desde cualquier otro tipo de aplicación

externa en cualquier momento. Las posibilidades de personalización y crecimiento de la solución son enormes, especialmente si tienen como base el escenario nacional al que está dirigida.

5.2. RECOMENDACIONES

Bajo la hipótesis de que todo software informático es perfectible, existe una gran variedad de ajustes y perfeccionamientos de nivel técnico que podrían ser realizados en *Call Manager* para un mejor desempeño, entre los cuales se mencionan:

- Optimización de la búsqueda de información en la base de datos local para la generación de reportes y estadísticas de operación del *call center*, en especial las funciones relacionadas con la recuperación de datos de las operadoras y zonas de calor por provincias. Así mismo se recomienda incluir sistemas avanzados de reportería (desarrollados o de terceros) que permitan visualizar y exportar los datos de operación del *call center* de una manera más detallada de ser necesario.

- Adicionalmente, y conforme a lo determinado en la disciplina de pruebas de la fase de transición, se recomienda añadir medidas adicionales de tolerancia a fallos, puesto que en este proyecto sólo fueron tratados los casos más importantes.

- En lo que respecta al procesamiento de eventos que se realiza en el EJB *EventController*, se recomienda descomponer dicho elemento de tal forma que su funcionamiento sea más modular y organizado. Esto permitiría ampliar las posibilidades de integración con otros sistemas u ofrecer características avanzadas de una forma más simplificada y menos intrusiva en el código fuente de la aplicación. Así mismo se invita a realizar la refactorización del

código del sistema con el propósito de mejorar su consistencia y claridad y facilitar su futuro mantenimiento.

- De la misma forma, en caso de ser necesario añadir medidas de seguridad se invita a gestionar el inicio de sesión de los usuarios mediante el empleo de procesos estandarizados de autenticación, autorización y registro (AAA), ya que en este proyecto son utilizadas cuentas locales solamente. También se sugiere utilizar el protocolo seguro de transferencia de hipertexto (HTTPS) para la transferencia segura de información entre los usuarios y los navegadores web. Además se recomienda firmar digitalmente los archivos que componen el cliente de escritorio *Call Manager Desktop* mediante el empleo de certificados digitales de confianza de tal forma que sea posible evitar la instalación de software malicioso por error. Otras medidas de seguridad inherentes a las plataformas de operación del sistema (como del servidor de aplicaciones o la base de datos, por ejemplo) son recibidas positivamente y están sujetas a discreción de los administradores del sistema.

Desde un punto de vista general, para *Call Manager* también existen recomendaciones de carácter operacional las cuales son:

- Previo a instalar el sistema debe comprobarse que los requisitos mínimos de despliegue hayan sido cumplidos conforme a lo estipulado en disciplina de despliegue de la fase de transición de este documento. Entre los requisitos de mayor importancia están: la adquisición de la licencia *CTILink Pro* de Avaya para la central telefónica, información de geolocalización en una base de datos o acceso a Internet para utilizar la función de georreferenciación de llamadas, o la actualización del JRE en todos los computadores de los clientes en caso de desplegar el cliente vía web (mediante *Java WebStart*). Se recomienda visitar la sección de anexos B, C y D para comprender a detalle el procedimiento de instalación y configuración de cada uno de los elementos que conforman *Call Manager*.

- Por otro lado, considerando que en este proyecto no fue creado un almacén de datos (o *data warehouse*) para recolectar y resumir los datos de utilización de la herramienta en el tiempo, se sugiere realizar copias periódicas de seguridad de las tablas antes de la eliminación de agentes o puestos de trabajo en el sistema.

- En cuanto se refiere al funcionamiento mismo del sistema, se recomienda reiniciar el servidor CTI si han sido añadidas o eliminadas extensiones telefónicas en la central telefónica. Para ello primero será necesario detener la conexión con el servidor principal (desde el menú *Comunicaciones* del administrador), reiniciar *Call Manager* CTI y a continuación reestablecer la conexión entre los servidores. Además, cualquier cambio realizado en la sección *Parámetros del Sistema* del administrador requiere del reinicio (o un nuevo despliegue) del servidor de aplicaciones para que los nuevos cambios sean cargados por los EJBs *MainController* y *EventController* apropiadamente.

- Se invita a la comunidad a mejorar y adaptar la solución aquí planteada con otras plataformas de telefonía compatibles con TAPI (ajenas a Avaya IP Office) con el ánimo de extender el ámbito de uso de la herramienta y ofrecer alternativas nacionales de gestión de *call centers*.

REFERENCIAS BIBLIOGRÁFICAS

LIBROS, CURSOS, Y MANUALES

[1] **YARBERRY, William**. Computer Telephony Integration. Segunda Edición. Auerbach Publications. USA. 2003.

[2] **WALTERS, Rob.** Computer Telephony Integration. Segunda Edición. ArtechHouse. Londres. 1999.

[3] **SULKIN, Allan.** PBX Sistems for IP Telephony. McGraw-Hill. 2004.

[4] **ARCOTEL.** Plan Técnico Fundamental de Numeración (PTFN). Resolución TEL-068-04-CONATEL-2013.

[5] **AVAYA.** IP Office 9.0.3 Product Description. Código 15-601041 Issue 27.04. 2014.

[6] **AVAYA.** Avaya IP Office ACSS Conversion Trainning. Código 5S99992O.

[7] **AVAYA.** Ventas de IP Office. Código ASC00121WEN.

[8] **PROISE, Jeff.** Programming Windows with MFC. Microsoft Press. 1999.

[9] **ORACLE.** The Java EE 7 Tutorial. Código E39031-01. Oracle Corporation. 2014.

[10] **SCHMULLER, Joseph.** Aprendiendo UML en 24 Horas. Prentice Hall.

[11] **PRESSMAN, Roger S.** Ingeniería del Software. Sexta Edición. Mc-Grawl Hill.

[12] **JACOBSON, Ivar; BOOCH Grady; RUMBAUGH James.** The Unified Software Development Process. Pearson Education. 1999.

[13] **BODIN, Madeline; DAWSON, Keith.** The Call Center Dictionary. CMP Books. 2002.

[14] **SHARP, Duane.** Call Center Operation: Design, Operation, and Maintenance. Digital Press. 2003.

[15] **AVAYA.** TAPI Link Installation. Código 15-601034. 2013.

[16] **AVAYA.** TAPILink Developer's Guide. Código 15-601035. 2013.

[17] **ÇIVICI, Çağatay.** PrimeFaces User Guide 5.0. PrimeTek Informatics.

[18] **GLINZ, Marting.** COCOMO (Constructive Cost Model). Seminar on Software Cost Estimation. PDF. 2003.

[19] **AVAYA.** Avaya IP Office Contact Center Brochure. 2014

[20] **HIDROBO, José Manuel.** Tecnología VoIP y Telefonía IP. Alfaomega, México, 2006.

[21] **HOHPE, Gregor; WOOLF, Booby**. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison Wesley. 2003.

[22] **COULOURIS, George; DOLLIMORE, Jean**; **KINDBERG Tim**. Distributed Systems: Concepts and Design. Tercera Edición. Addison-Wesley. 1994.

DIRECCIONES ELECTRÓNICAS

[E-1]
http://controlenlinea.supertel.gob.ec/wps/portal/informacion/informaciontecnica/telefoniafija/

[E-2] https://msdn.microsoft.com/en-us/library/windows/desktop/ms734273(v=vs.85).aspx

[E-3] https://msdn.microsoft.com/en-us/library/windows/desktop/ms733433(v=VS.85).aspx

[E-4] https://msdn.microsoft.com/en-us/library/windows/desktop/ms733435(v=VS.85).aspx

[E-5] https://msdn.microsoft.com/en-us/library/windows/desktop/ms734223(v=vs.85).aspx

[E-6] http://www.oracle.com/technetwork/java/jtapi-136088.html

[E-7] http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado

[E-8]

http://upload.wikimedia.org/wikipedia/commons/a/a5/RUP_disciplines_greyscale_20060121.svg

[E-9] http://docs.oracle.com/javase/tutorial/deployment/webstart/index.html

[E-10] *http://www.evolutioncallcenter.com/index.php*

[E-11]

http://www.evolutioncallcenter.com/download/v10/doc/Ficha%20de%20producto%20Evolution%20Avaya%20IP%20Office.pdf

[E-12] http://www.openstreetmap.org

ANEXOS

ANEXO A
GLOSARIO DE TÉRMINOS Y DEFINICIONES

AAA (Authorization, Authentication, Accounting).- Corresponde a la familia de protocolos de seguridad en la que se realizan tres funciones: Autorización, Autenticación, y Contabilización, para el aprovisionamiento de servicio a usuarios en sistemas distribuidos.

API (Application Programming Interface).- O Interfaz de Programación de Aplicaciones, define la capa de abstracción de un software que puede ser empleado por otro mediante el empleo funciones y procedimientos.

ATM (Asynchronous Transfer Mode).- O Modo de Transferencia Asíncrona, es una tecnología de comunicaciones de gran capacidad de transmisión para servicios y aplicaciones.

CNT.- Corporación Nacional de Telecomunicaciones.

CÓDEC (Codificador-Decodificador).- Es una tecnología desarrollada en software y/o hardware que transforma una señal analógica o un flujo de datos para su posterior transmisión, almacenamiento o cifrado, siendo posible recuperar esta información para manipularla en un formato más apropiado.

Compilador.- Programa encargado de traducir (transformar) el código de una aplicación escrita en un lenguaje de programación a otro lenguaje que el computador sea capaz de interpretar (generalmente en forma binaria).

COM (Component Object Model).- Es una plataforma de software utilizada para la comunicación entre procesos y la creación de objetos dinámicos.

DLL (Dinamic Link Libraries).- Son bibliotecas que contienen código y datos que pueden ser utilizados por más de un programa al mismo tiempo.

DHCP (Dynamic Host Configuration Protocol).- Es un protocolo de red que permite asignar dinámicamente los parámetros de configuración de red de los clientes.

DTMF (Dual-Tone Multi-Frequency).- O sistema de marcación por tonos, es un sistema de señalización sobre líneas telefónicas analógicas entre teléfonos y otros dispositivos de comunicación y la central telefónica.

Frame Relay.- Es una tecnología de comunicación de voz y datos a alta velocidad mediante la transmisión de información en “tramas” o “marcos”.

***Framework*.-** En desarrollo de software, es una capa de abstracción que proporciona una funcionalidad genérica y en la que otros proyectos de software pueden ser desarrollados y organizados más fácilmente.

GUI (Graphical User Interface).- O interfaz gráfica de usuario, es un programa informático compuesto por imágenes y objetos interactivos que proporciona un entorno visual sencillo y amigable para el usuario.

HTTPS (Hypertext Transfer Protocol Secure).- O protocolo seguro de transferencia de hipertexto, es la versión segura de su similar HTTP.

IETF (Internet Engineering Task Force).- Es una organización de normalización que regula las propuestas y estándares de Internet.

ISO (*International Organization for Standardization*).- Organismo que promueve el desarrollo de normas internacionales de fabricación, comercio y telecomunicaciones.

***IPsec (Internet Protocol Security)*.-** Es un conjunto de protocolos de seguridad cuyo objetivo es la autenticación y el cifrado de paquetes IP en un flujo de datos.

ITU (International Telecommunication Union).- Véase UIT.

LDAP (Lightweight Directory Access Protocol).- Es un protocolo que permite acceder a servicios de directorio ordenados y distribuidos con propósitos de búsqueda de información.

MSPI (Media Service Provider Interface).- O Interfaz del Proveedor de Servicios de Medios, otorga a las aplicaciones un considerable control del medio para un mecanismo de transporte particular.

Middleware.- En entornos distribuidos, es una capa de abstracción de software utilizada para la interacción con otras aplicaciones, redes o hardware.

Multiplexación.- La multiplexación puede definirse como el procedimiento por el cual diferentes canales de información (señales) comparten un mismo medio de transmisión (enlace) con el propósito de aumentar la eficiencia de uso de éste. En telefonía fija el esquema más común es la Multiplexación por División de Tiempo (TDM), en el cual el dominio del tiempo es dividido en ranuras, asignándose una ellas a cada señal para ser transmitida por el enlace.

MPLS (Multi-Protocol Label Switching).- Es un mecanismo de transmisión diseñado para unificar el servicio de transporte para redes de circuitos y de paquetes, para varios tipos de tráfico como voz e IP.

Persistencia de objetos.- Se refiere al almacenamiento y recuperación de los valores de los atributos de uno o más objetos, cuyas fuentes pueden ser tablas, archivos, entre otros.

PCM (Pulse Code Modulation).- o Modulación de Impulsos Codificados**,** es una técnica de digitalización de señales analógicas, donde "*la información contenida en las muestras de la señal analógica son cuantizadas y luego codificadas es decir representadas por palabras binarias (secuencias de bits) que entrega el codificador*"[21].

POJO (Plain Old Java Object).- Es el nombre utilizado por los programadores para referirse a un objeto Java simple e independiente de algún modelo, convención o framework particular.

[21] Comunicación Digital, MSc. Ma. Soledad Jiménez, EPN, Pág. 116.

PSTN (Public Switched Telephone Network).- Red de Telefonía Pública Conmutada. Descrita en la sección 1.1.3 de este Proyecto de Titulación.

PTFN (Plan Técnico Fundamental de Numeración).- Véase apartado 1.1.7.1 de este Proyecto de Titulación.

RIP (*Routing Information Protocol*).- Es un protocolo de enrutamiento utilizado por equipos de capa 3 para el intercambio de información de redes IP.

SMDR (Station Messaging Detail Record).- Es un método de gestión y registro de las actividades de los sistemas de telecomunicaciones. También suele ser conocido como CDR (*Call Detail Record*).

TAPI (Telephony Application Programming Interface).- O Interfaz de Programación de Aplicaciones de Telefonía, es la alternativa de Microsoft para la creación de aplicaciones CTI.

TSPI (Telephony Service Provider Interface).- O Interfaz del Proveedor de Servicios de Telefonía, es una interfaz que a través de comandos estandarizados soporta la gestión y control de los dispositivos de comunicaciones.

UIT (Unión Internacional de Telecomunicaciones).- Sector de normalización de las telecomunicaciones.

ANEXO B

GUÍA DE INSTALACIÓN DE *CALL MANAGER*: SERVIDOR CTI

Instalación del Controlador *TapiLink* de Avaya

1. Extraer los contenidos del archivo *User4_2_43.exe* en un directorio temporal y ejecutar el archivo *Setup.exe*. Seleccionar el idioma de instalación y seguir con los pasos indicados en el *wizard* de instalación.

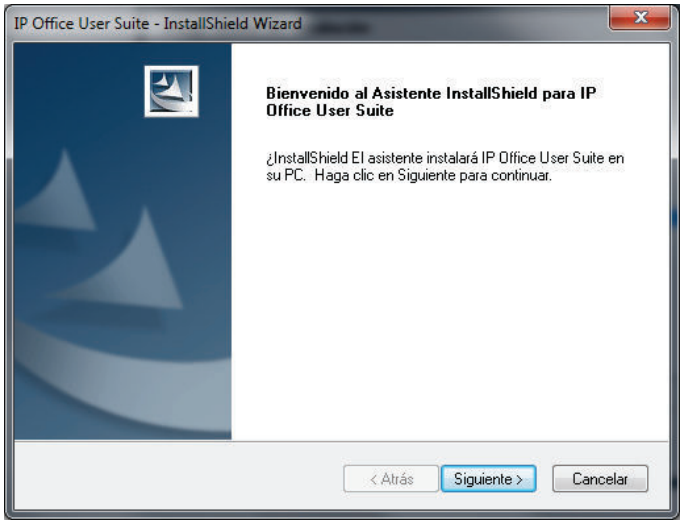


Ilustración B-1 Asistente de Instalación del TAPI de Avaya

2. Introducir la información del cliente y seleccionar el directorio de instalación.

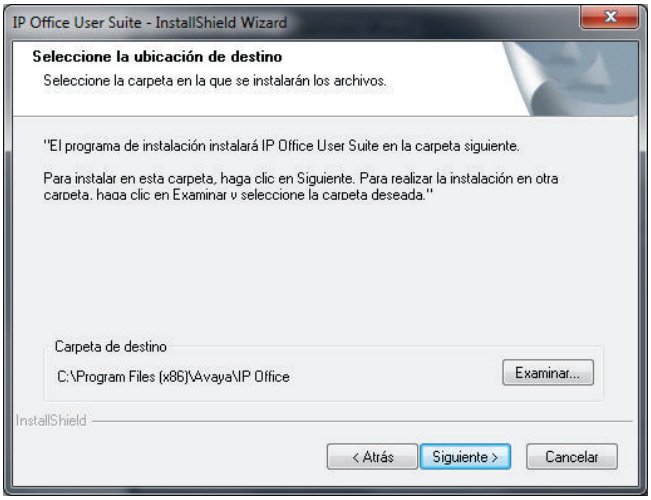


Ilustración B-2 Selección del Directorio de Instalación del Controlador

3. Seleccionar el tipo de instalación *Personalizado*.

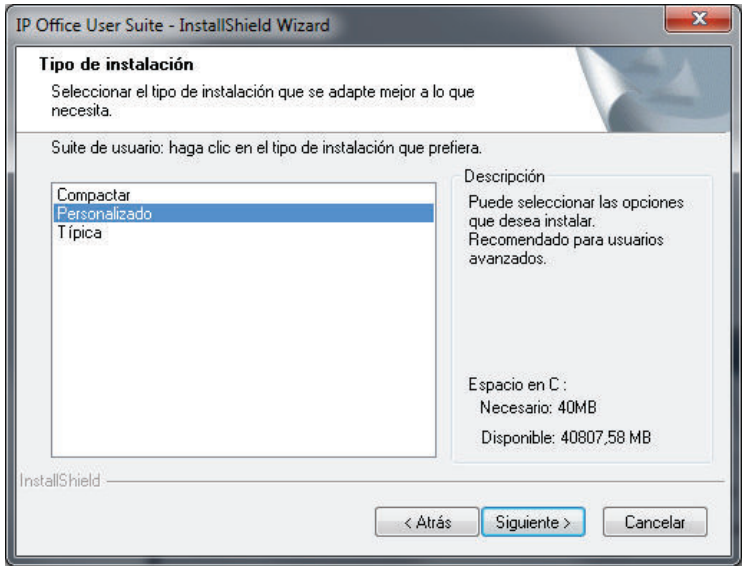


Ilustración B-3 Tipo de Instalación Personalizado del Controlador TAPI de Avaya

4. De entre las opciones ofrecidas seleccionar TAPI.

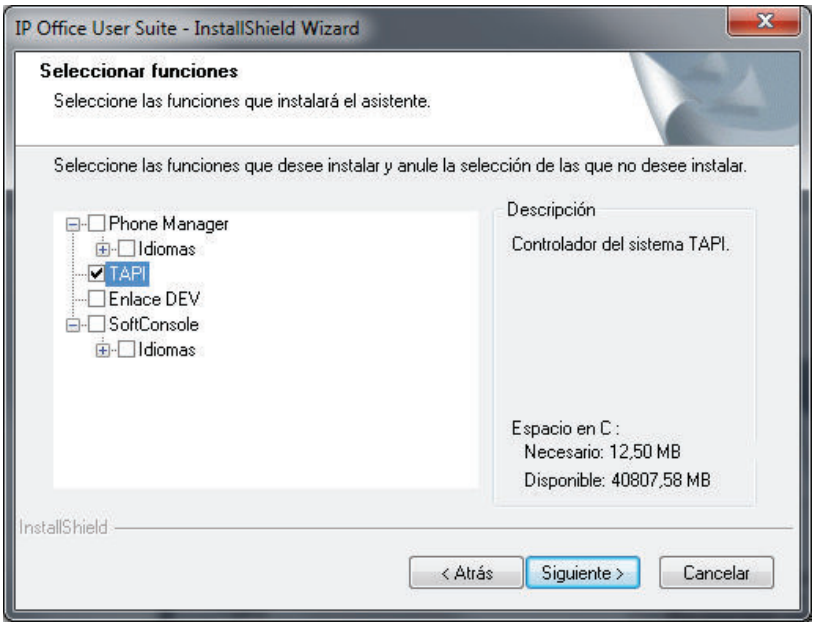


Ilustración B-4 Selecciones de Funciones del TAPI de Avaya

5. Al dar clic en siguiente aparecerá una ventana emergente informando que no ha sido detectado ningún equipo IP Office en la red. Esto es normal y debe cerrarse para continuar con la instalación.

6. Seguidamente tienen que ingresarse las credenciales de acceso que el control utilizará para conectarse a la central telefónica. Para ello también será necesario ingresar la dirección IP del equipo de comunicaciones como se puede apreciar en la figura B.5.

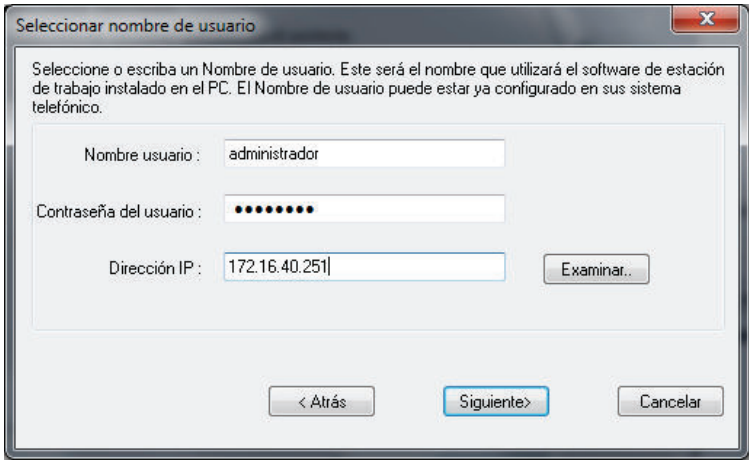


Ilustración B-5 Ingreso de Credenciales de Acceso del TAPI de Avaya

7. Finalizar la instalación y reiniciar el equipo.
8. Acceder a la sección ***Teléfono y Módem*** del Panel de Control de Windows y seleccionar la pestaña *Opciones Avanzadas*.

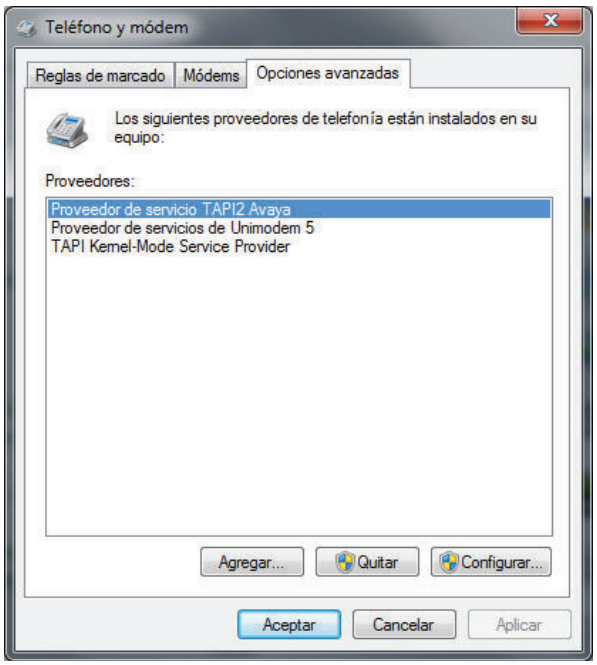


Ilustración B-6 Configuración del Proveedor de Servicio de Telefonía en Windows

9. Seleccionar el proveedor de servicios ***TAPI2 Avaya*** y dar clic en el botón *Configurar*.
10. Verificar que la dirección IP corresponde a la central telefónica e ingresar la contraseña de administración. Guardar los cambios y reiniciar el equipo.

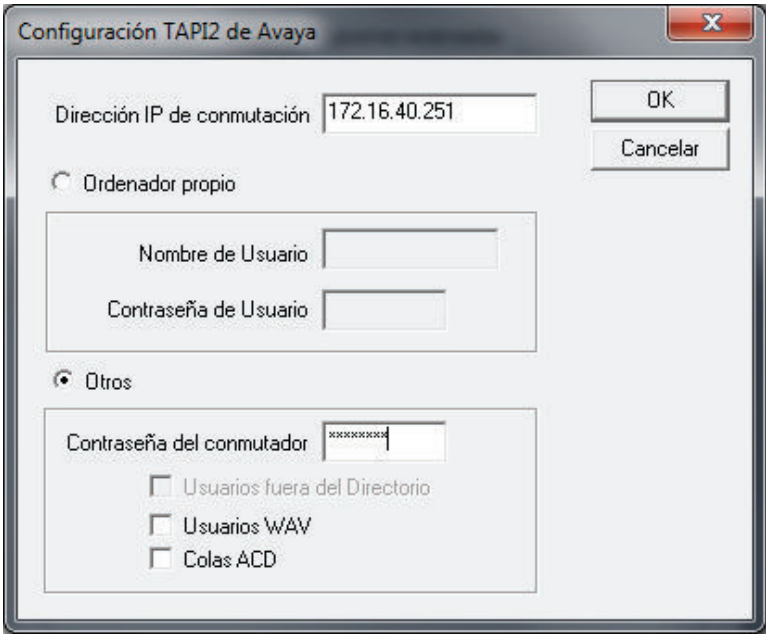


Ilustración B-7 Parámetros de Configuración del Proveedor de Servicios

Instalación de *Call Manager CTI Controller*

1. Instalar los prerrequisitos definidos en la sección de despliegue: Controlador CTI, plataforma .NET *Framework* 3.5 o superior y *Windows Installer* 3.1 o superior.

2. Ejecutar el archivo ***Setup.exe*** o ***CallManagerSetup.msi*** para iniciar la instalación del componente CTI mediante el seguimiento de los pasos del *wizard* de instalación.

Ilustración B-8 Pantalla Inicial del Asistente de Instalación de *Call Manager CTI*

3. Seleccionar el directorio de instalación.

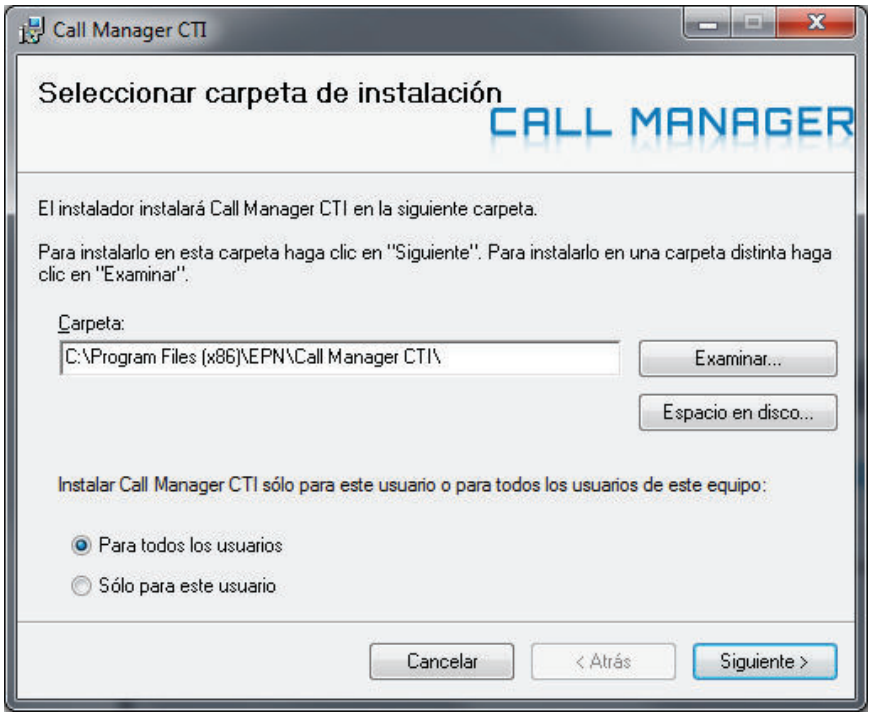


Ilustración B-9 Selección del Directorio de Instalación de *Call Manager CTI*

4. Ingresar el puerto de escucha del servidor (por defecto 47915).

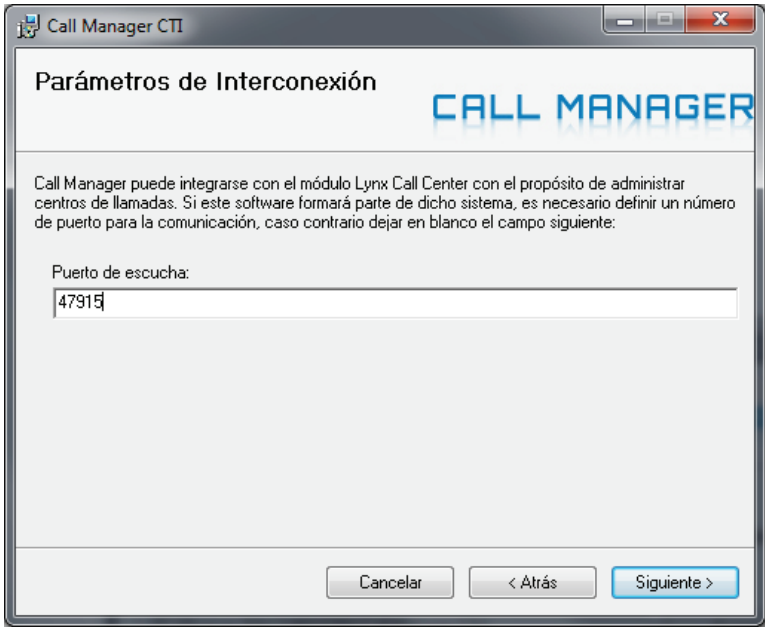


Ilustración B-10 Ingreso del Puerto de Interconexión con el Servidor Principal

5. Elegir si se desea que la aplicación funcione al arrancar Windows y si se desea crear un ícono de acceso en el escritorio.

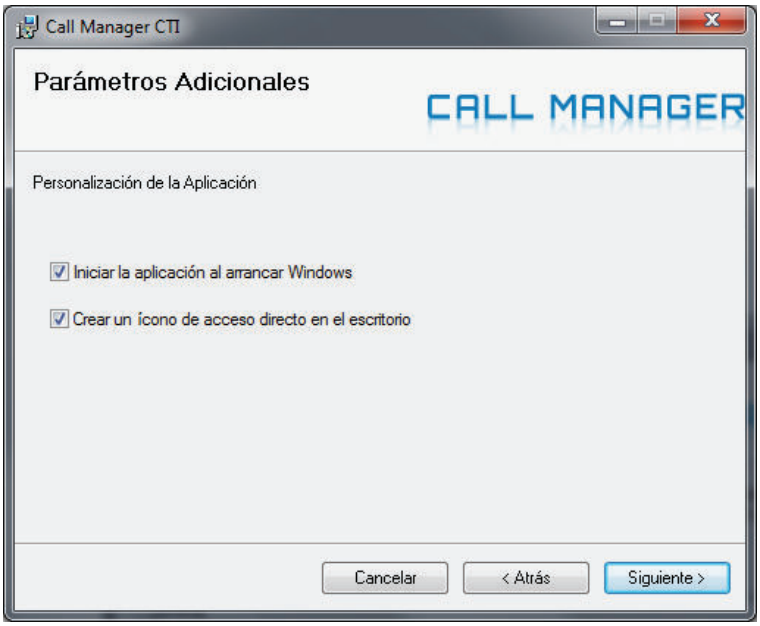


Ilustración B-11 Selección de Parámetros de Instalación de *Call Manager CTI*

6. Confirmar y finalizar la instalación.

ANEXO C

GUÍA DE DESPLIEGUE DE *CALL MANAGER*: SERVIDOR PRINCIPAL

Esta guía asume que los siguientes requisitos han sido cumplidos (según lo descrito en la disciplina de despliegue de la fase de transición TD-CORE):

- Servidor CentOS 6 de 64 bits.
- Plataforma Java JDK 7 Update 4 de 64 bits.
- Cortafuegos deshabilitado.

Además se recalca que todos los procedimientos de instalación se han realizado mediante el uso de interfaces gráficas para facilitar la comprensión de este apartado.

Instalación del Gestor de Base de Datos *PostgreSQL* y Creación de la BDD CallManager DB

1. Abrir el asistente de instalación. Seleccionar los directorios de instalación y datos.

Ilustración C-1 Selección del Directorio de Instalación de *PostgreSQL*

2. Ingresar la contraseña de super-usuario, ingresar el número de puerto y seleccionar la configuración regional de idioma.

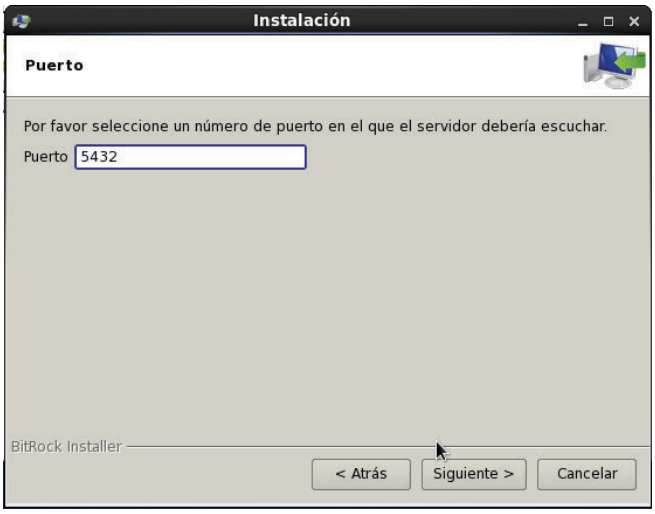


Ilustración C-2 Selección del Número de Puerto de *PostgreSQL*

3. Finalizar y completar la instalación. Instalar las herramientas de administración en caso de ser necesario.
4. Crear el usuario ***managerUser*** en el gestor de base de datos.

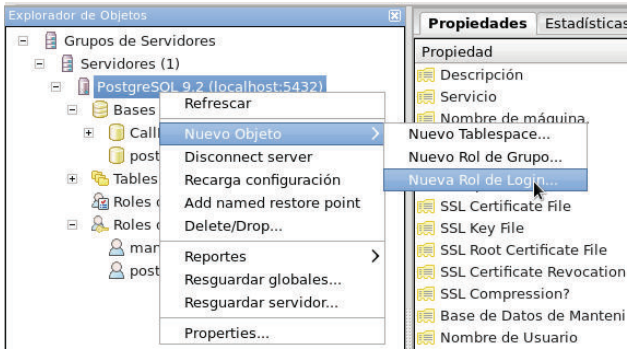


Ilustración C-3 Creación de un Nuevo Rol de Inicio de Sesión en *PostgreSQL*

5. Crear la base de datos ***CallManagerDB*** y asignar al usuario *managerUser* privilegios de administración de ésta.

```
-- DROP DATABASE "CallManagerDB";
CREATE DATABASE "CallManagerDB"
  WITH OWNER = "managerUser"
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       LC_COLLATE = 'es_ES.UTF-8'
       LC_CTYPE = 'es_ES.UTF-8'
       CONNECTION LIMIT = -1;
```

Ilustración C-4 Script de Creación de la BDD *CallManagerDB*

6. Ejecutar el *script* ***Tablas.sql*** en *CallManagerDB* para la creación de las tablas del sistema.

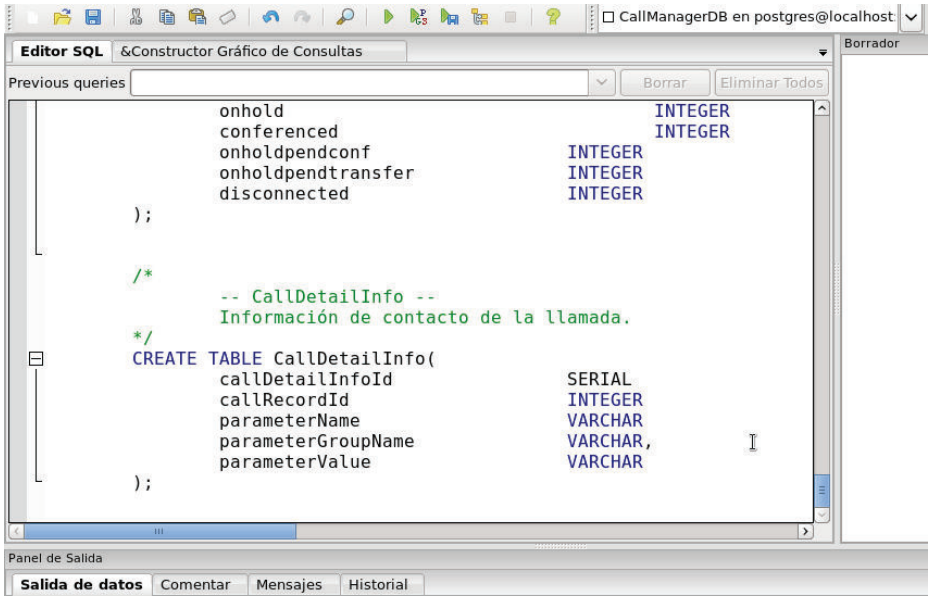


Ilustración C-5 Ejecución del *Script* Tablas.sql

7. Ejecutar el script ***InsercionDatos.sql*** en *CallManagerDB* para la creación de los registros por defecto del sistema.

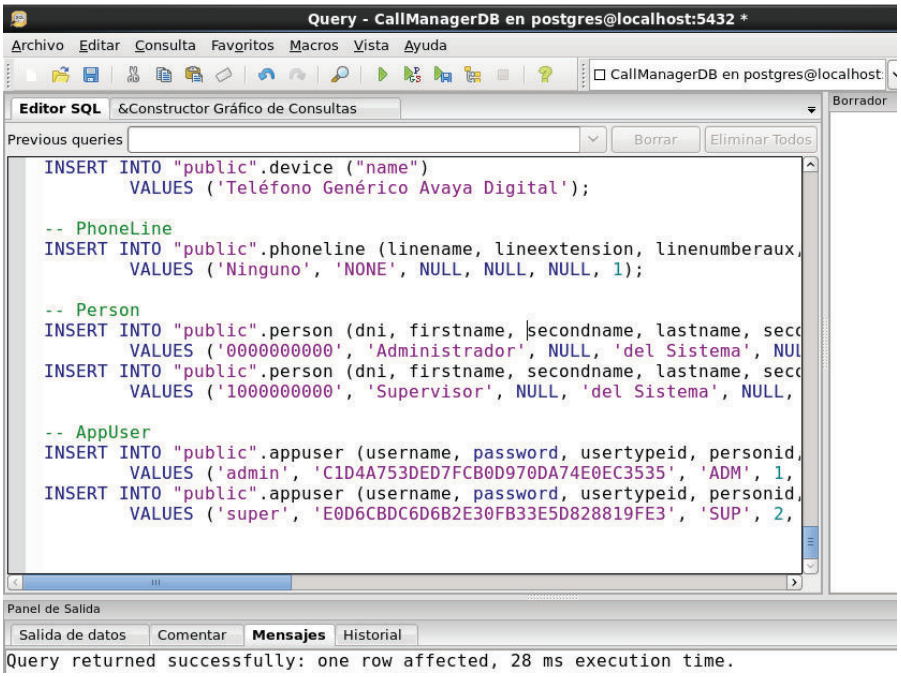


Ilustración C-6 Ejecución del *Script* InsercionDatos.sql

Instalación del Servidor de Aplicaciones *GlassFish Server Open Source Edition*

1. Abrir el asistente de instalación y elegir la opción *Instalación Personalizada*.

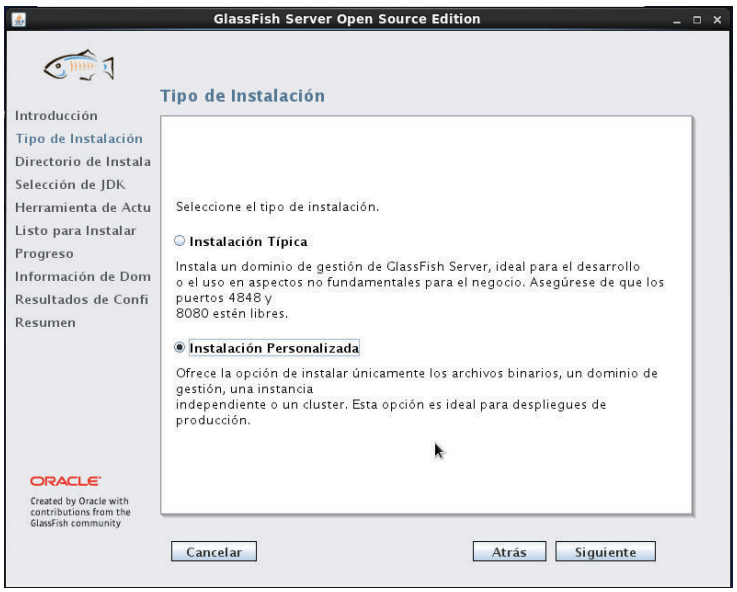


Ilustración C-7 Instalación Personalizada de *GlassFish Server Open Source Edition*

2. Ingresar el directorio de instalación.

Ilustración C-8 Especificación del Directorio de Instalación de *GlassFish*

3. Elegir la activación de la herramienta de actualización (de ser necesaria) y proceder a instalar el software.

4. Configurar los parámetros de información del dominio.

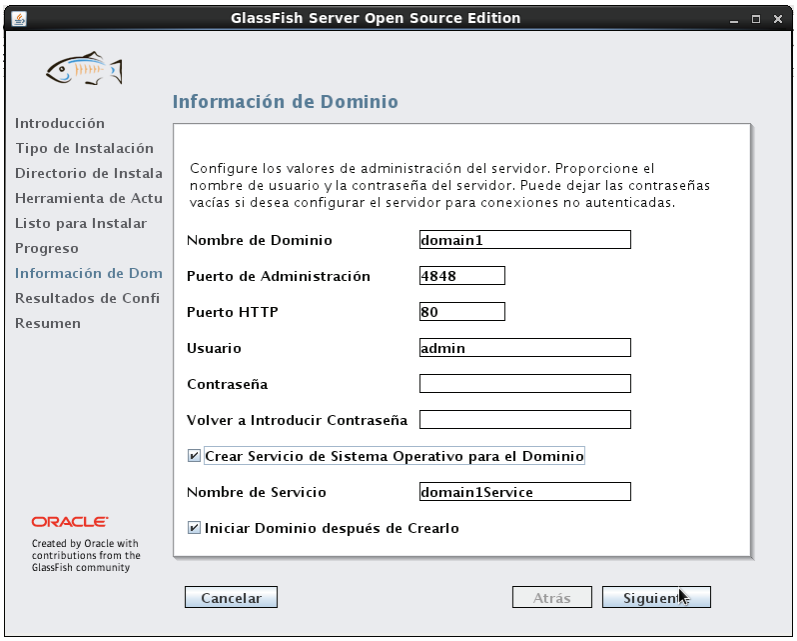


Ilustración C-9 Ingreso de Información de Dominio de *GlassFish*

5. Verificar los cambios y salir del asistente.

Interconexión del Servidor de Aplicaciones con Bases de Datos

1. Instalar los JDBC según las bases de datos a interconectar en el servidor de aplicaciones. Para ello será necesario copiar los controladores de extensión ***.jar*** correspondientes a los JDBCs en la carpeta ***lib*** de la instancia del servidor de aplicaciones *Glassfish*. Seguidamente es necesario reiniciar el servidor para que las nuevas librerías sean cargadas y utilizadas por el mismo.

Ilustración C-10 Drivers JDBC de las BDD soportadas instaladas en *GlassFish*

2. El siguiente paso consiste en la configuración del servidor de aplicaciones para que éste pueda acceder a las bases de datos. Para ello será necesario acceder a la página de administración que conforme a una instalación estándar es accedida desde el enlace:

http://IP_SERVIDOR_APP:4848/common/index.jsf

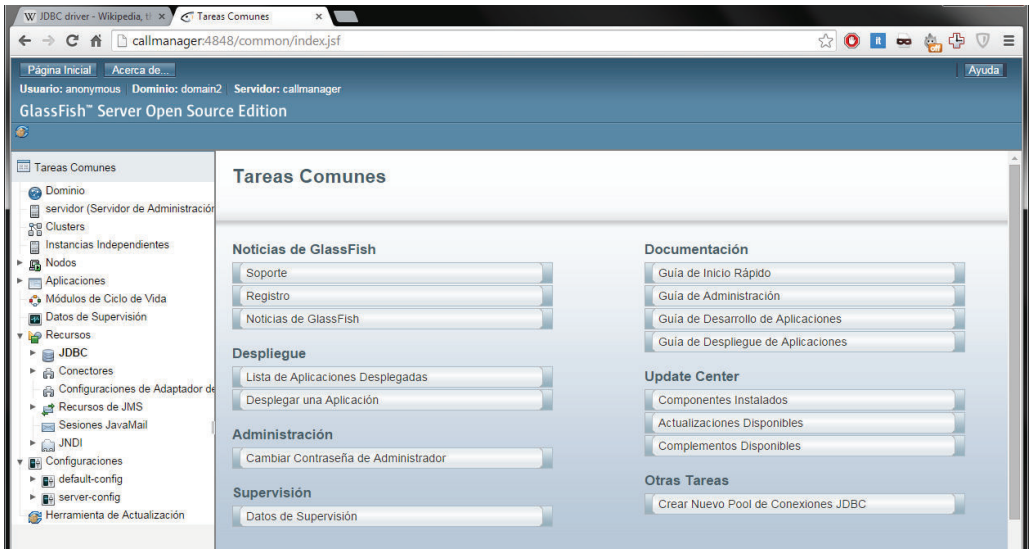


Ilustración C-11 Página de Administración del Servidor de Aplicaciones

3. A continuación acceder a la sección ***Recursos → JDBC → Pools de Conexiones JDBC*** para crear la conexión con la(s) base(s) de datos externa(s).

Ilustración C-12 Creación de un Nuevo *Pool* de Conexiones en *GlassFish*

4. Al dar clic en el botón ***Nuevo*** de la tabla se abre un *wizard* en el que deben ingresarse los parámetros de interconexión según el tipo de base de datos a utilizar. Para la base de datos local el nombre del *pool* será ***jdbc/CMW1Pool*** cuyo tipo de recurso será ***javax.sql.XADataSource***. Para las bases de datos externas el nombre del recurso será definido por el administrador y el tipo de recurso deberá ser ***javax.sql.DataSource***.

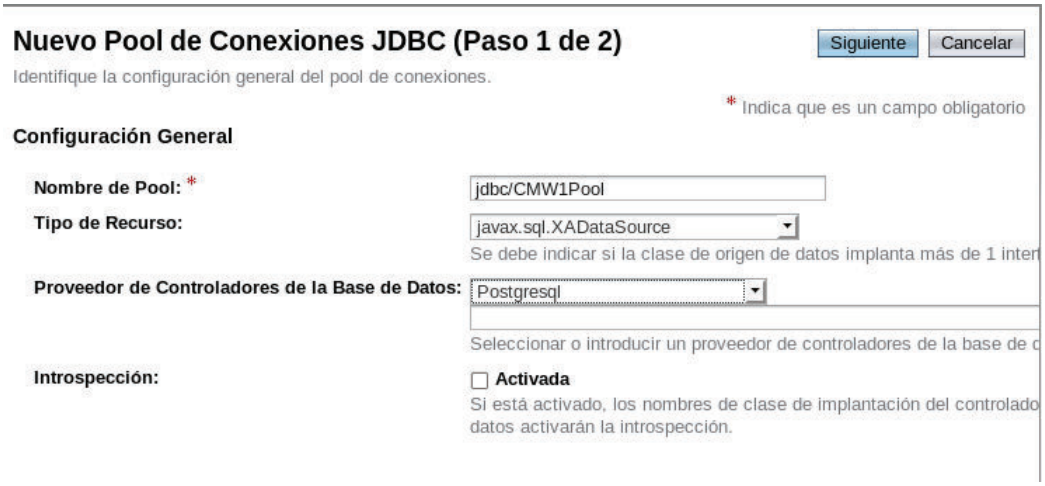


Ilustración C-13 Parámetros de Creación de un Nuevo *Pool* de Conexiones

5. Al dar clic en *Siguiente* deben ingresarse las propiedades de acceso a la base de datos como el nombre de usuario, base de datos, servidor, número de puerto y contraseña.

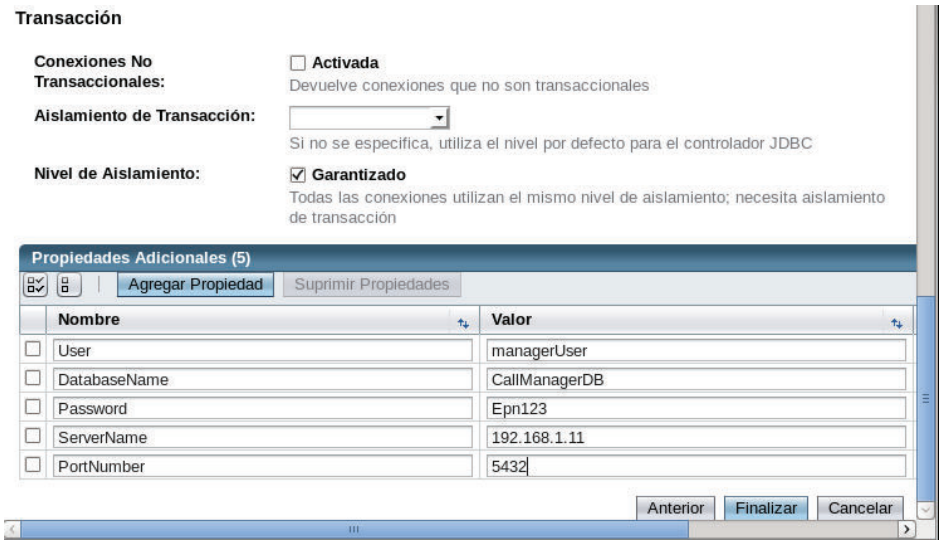


Ilustración C-14 Propiedades Adicionales de Conexión a la BDD

6. Enseguida debe seleccionarse el recurso recientemente creado para comprobar el acceso a la BDD mediante el botón ***Ping.*** Si el resultado es positivo es posible continuar con el paso 7.

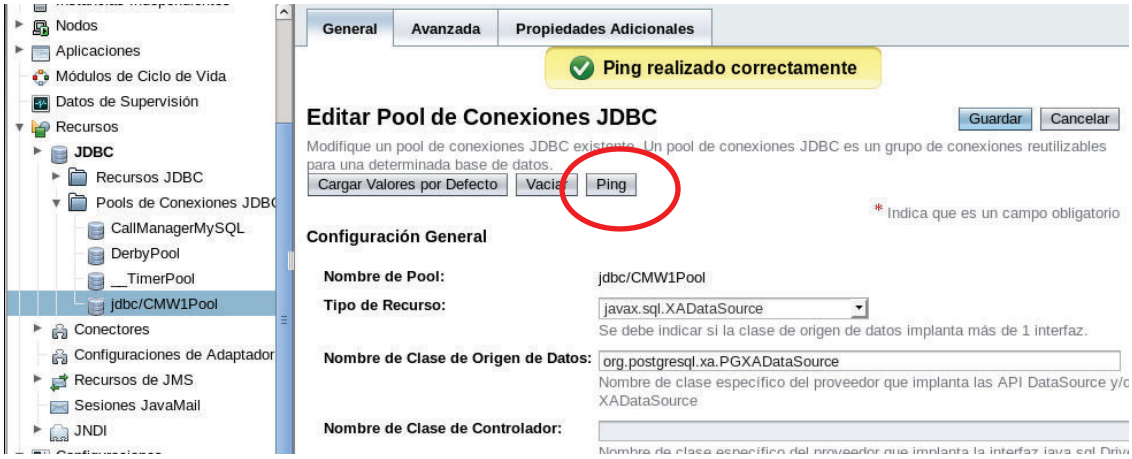


Ilustración C-15 Comprobación de Acceso a la BDD Desde el Servidor de Aplicaciones

7. Posteriormente en la sección ***Recursos JDBC*** deben crearse dos nuevos elementos: el primero deberá tener por nombre ***jdbc/CMW1JDBC*** y estará relacionado con el *pool* de la BDD local (es decir, jdbc/CMW1Pool), mientras que el segundo se denominará ***jdbc/ClientDatabaseJDBC*** y será utilizado para la interconexión con la base de datos externa (con su *pool* correspondiente).

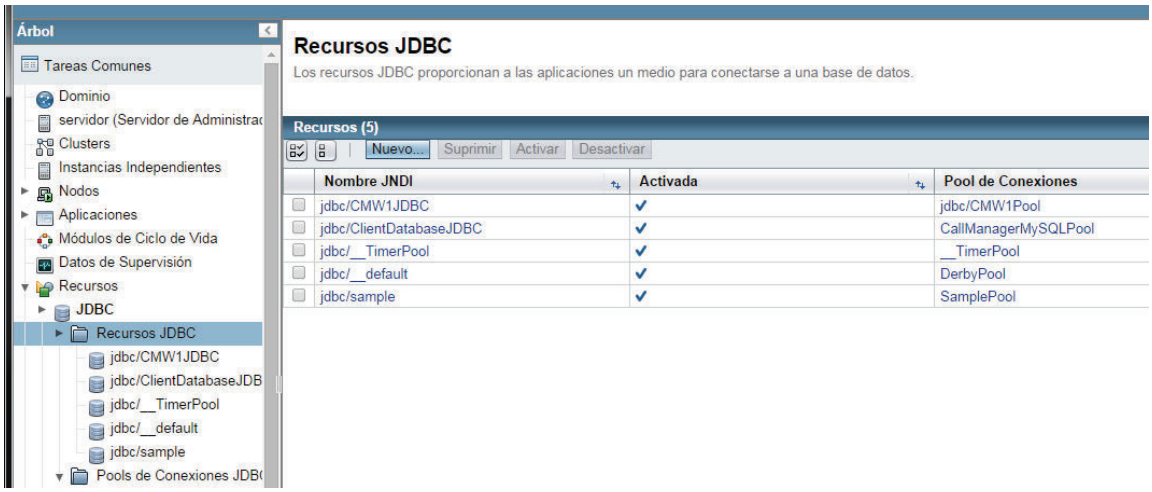


Ilustración C-16 Recursos JDBC del Servidor de Aplicaciones

Ilustración C-17 Asociación del *Pool* de Conexiones con el Recurso JDBC para la BDD Local

Despliegue de la Aplicación Empresarial *Call Manager*

1. Ingresar a la sección ***Aplicaciones*** de la consola de administración del servidor de aplicaciones mediante el enlace:

   http://IP_SERVIDOR_APP:4848

Ilustración C-18 Sección Aplicaciones de la Consola de Administración de *GlassFish*

2. Al dar clic en el botón *Desplegar* se podrá cargar en el servidor un archivo empaquetado de extensión ***.ear***.

Ilustración C-19 Pantalla de Despliegue de Aplicaciones en *GlassFish*

3. Seleccionar el archivo ***CMW1.ear*** de la fase de transición (TD-CORE) y seleccionar el tipo de aplicación como *Aplicación Empresarial*.

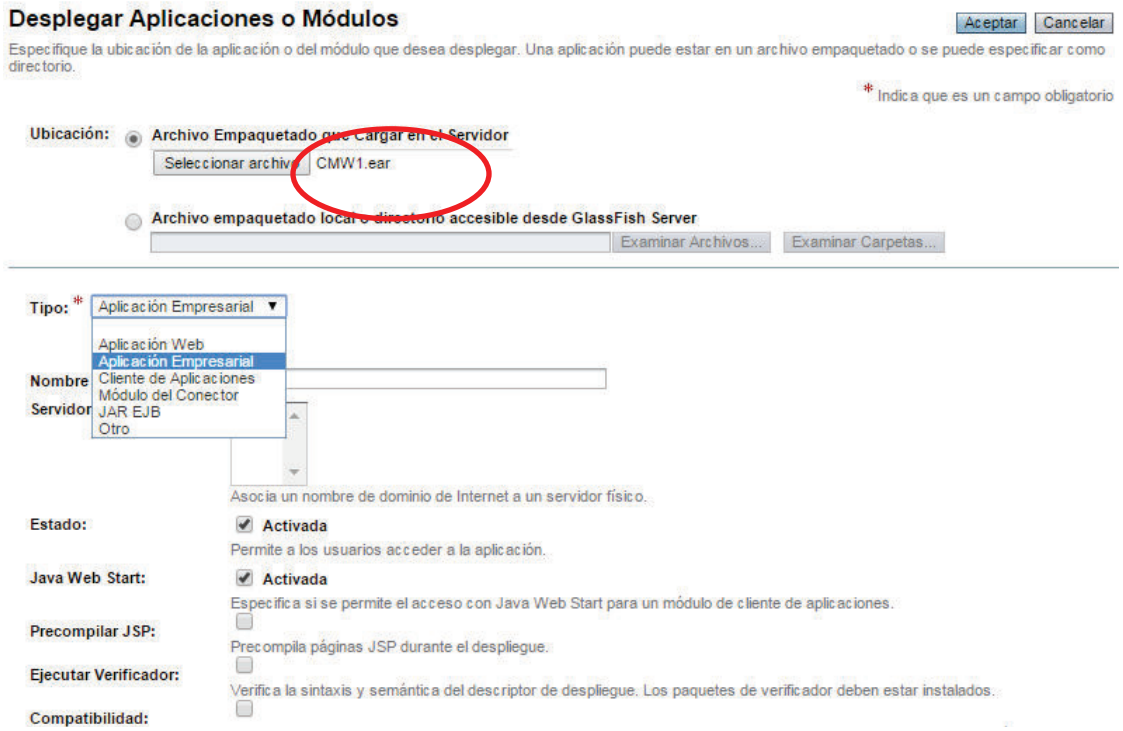


Ilustración C-20 Carga de la Aplicación *Call Manager* en *GlassFish*

4. Al dar clic en *Aceptar* luego de un corto tiempo de espera se visualizará la aplicación adecuadamente instalada y lista para su uso.

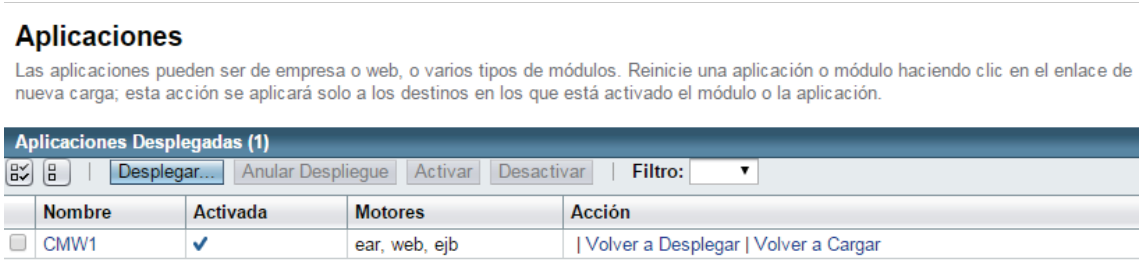


Ilustración C-21 *Call Manager* Correctamente Desplegado en *GlassFish*

5. Comprobar el adecuado funcionamiento del sistema mediante el enlace:

http://IP_SERVIDOR_GLASSFISH/CMW1-war

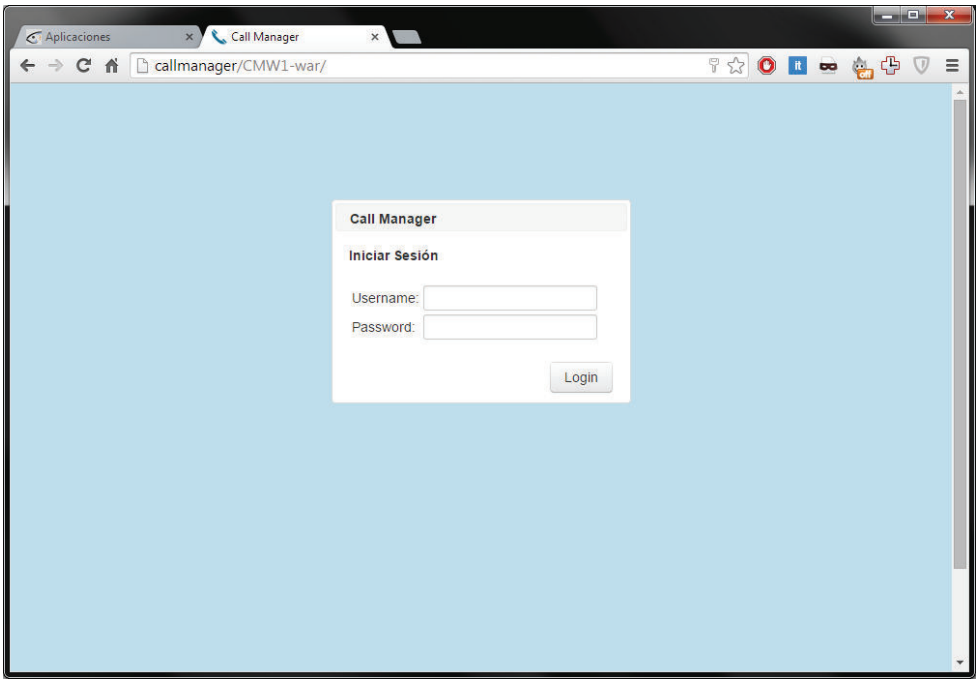


Ilustración C-22 Pantalla de Inicio de Sesión de *Call Manager*

ANEXO D

GUÍA DE INSTALACIÓN DE *CALL MANAGER*:

AGENTE DE ESCRITORIO

Conforme a lo definido en la fase de transición (TD-DSK), el siguiente prerrequisito ha sido cumplido:

- Java Runtime Environment (JRE) 7 Update 3 o superior instalado.

Instalación en Ambientes Windows Mediante el Empleo del Instalador

1. Ejecutar el archivo ***setup.exe*** para iniciar la instalación del programa.

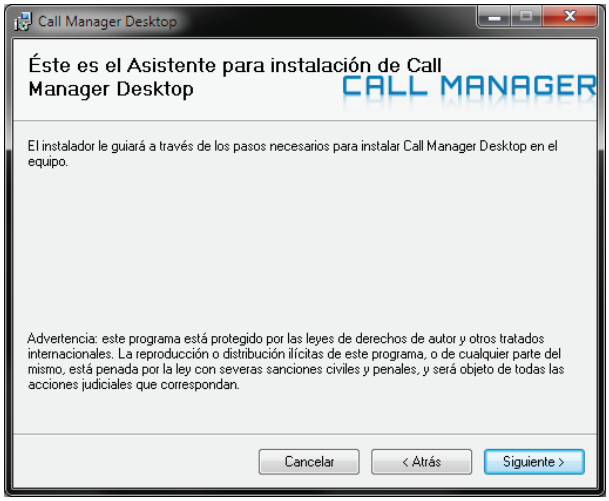


Ilustración D-1 Asistente de Instalación de *Call Manager Desktop*

2. Ingresar el directorio de instalación del programa y completar la instalación.

Ilustración D-2 Ingreso del Directorio de Instalación de *Call Manager Desktop*

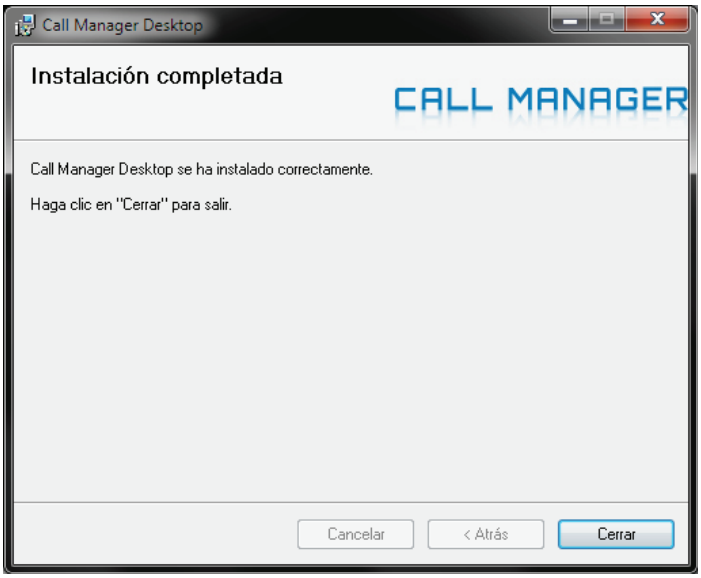


Ilustración D-3 Instalación de *Call Manager Desktop* Completada

3. Ir a la sección ***Configuración Inicial de Call Manager Desktop*** descrita más adelante en este anexo.

Instalación Mediante el Empleo de Java WebStart

1. De acuerdo a lo dispuesto en la disciplina de despliegue de la fase de transición debe comprobarse que el sistema operativo del cliente es capaz de resolver el nombre ***callmanager*** que apunta al servidor principal (*GlassFish*).

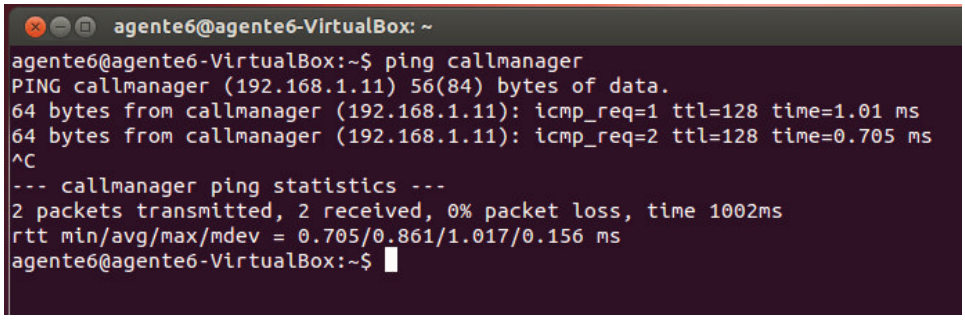


Ilustración D-4 Prueba de Conectividad con el *Servidor Call Manager*

2. Ingresar al *Panel de Control de Java* desde el sistema operativo. En la pestaña *Seguridad*, hacer clic en el botón *Editar lista de sitios…*

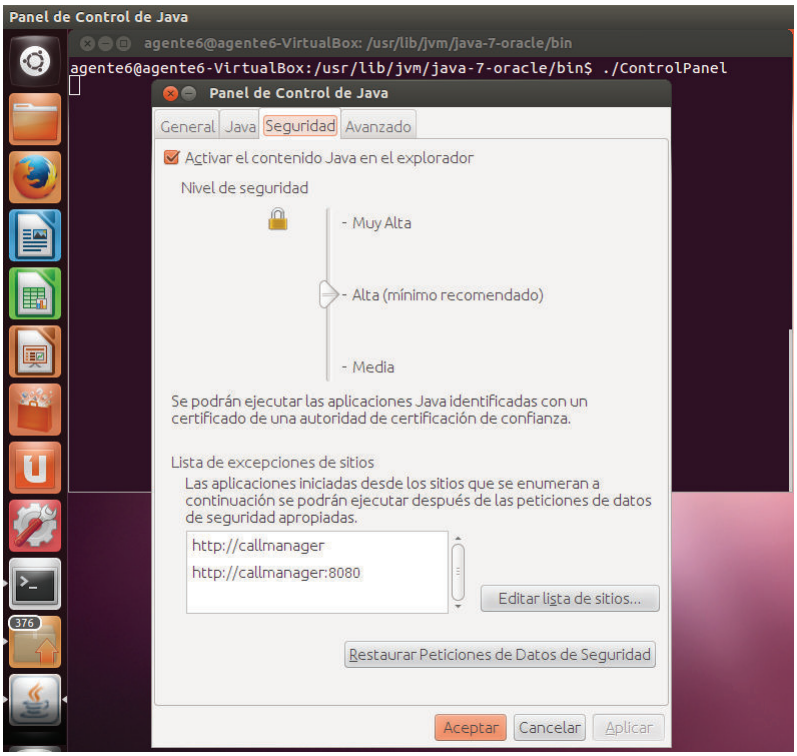


Ilustración D-5 Pestaña Seguridad del Panel de Control de Java en Ubuntu

3. Ingresar la ubicación del servidor principal en el formato: ***http://callmanager***

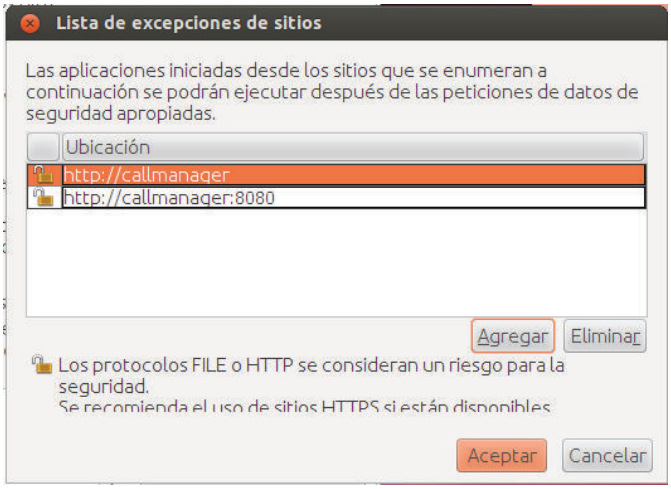


Ilustración D-6 Adición del Servidor Principal en la Lista de Excepciones de Sitios

4. Aceptar y guardar los cambios.
5. Ingresar con credenciales de agente en URL: **http://callmanager/CMW1-war**
6. Acceder a la sección *Aplicación Cliente* y dar clic en la imagen del logotipo del programa (botón *Descargar*) para iniciar la descarga del archivo *.jnlp*. Es

posible que el navegador web bloquee la ventana emergente de la descarga por lo que es necesario añadir la excepción necesaria para el efecto.

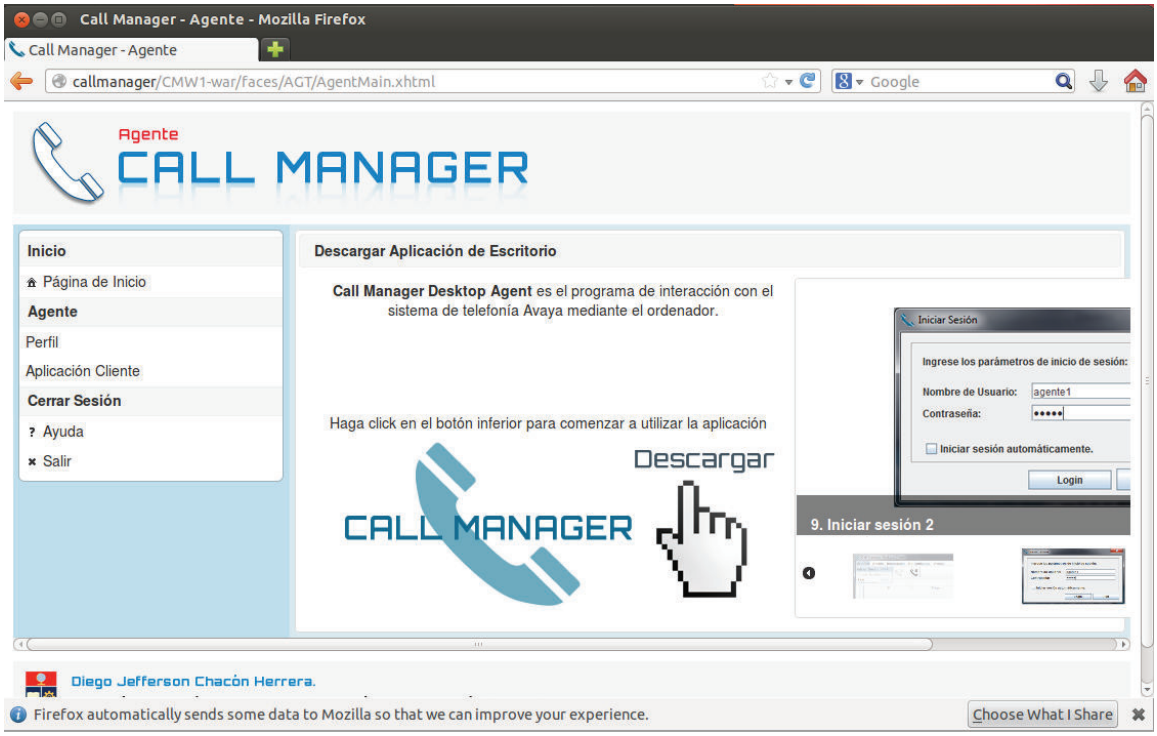


Ilustración D-7 Página *Aplicación Cliente* del Agente de *Call Center*

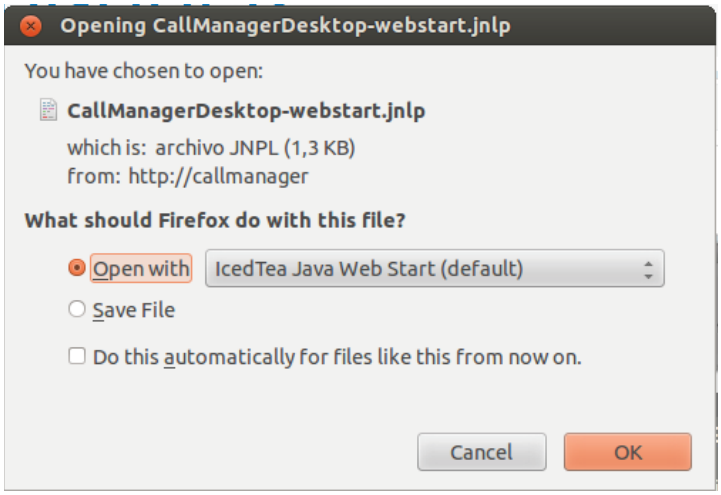


Ilustración D-8 Descarga del archivo JNPL desde el Servidor Principal

7. Ejecutar el archivo descargado, aceptar la casilla de aceptación de riesgos (ya que el software es autofirmado) y dar clic en el botón *Ejecutar*.

Ilustración D-9 Instalación de *Call Manager Desktop* vía *Java WebStart*

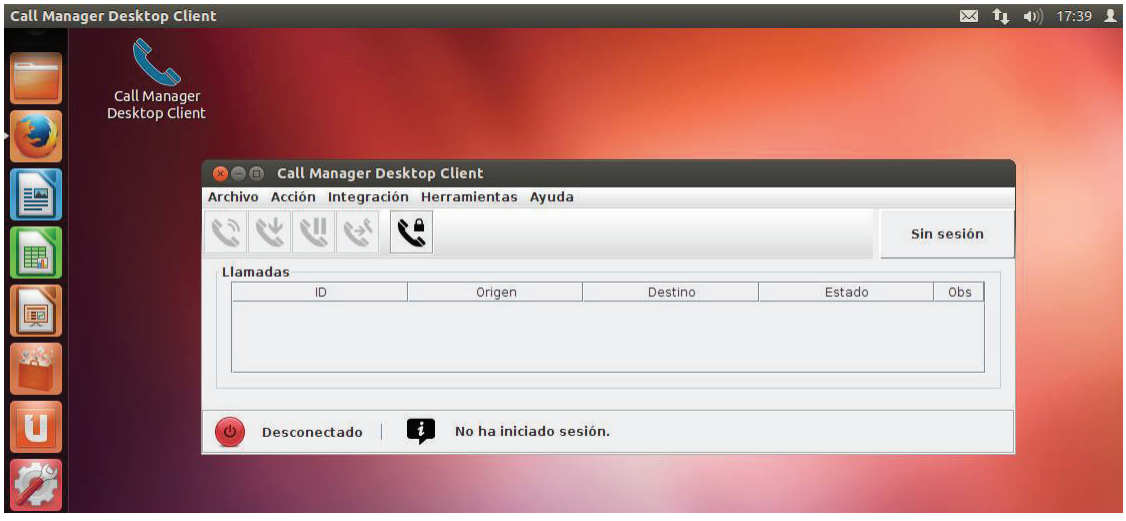


Ilustración D-10 *Call Manager Desktop* Instalado

8. El programa se abrirá y estará listo para ser inicializado en la sección **Configuración Inicial de *Call Manager Desktop*** descrita a continuación.

Configuración Inicial de *Call Manager Desktop*

1. Abrir el aplicativo de escritorio y acceder la opción ***Configuración*** del menú ***Herramientas***.

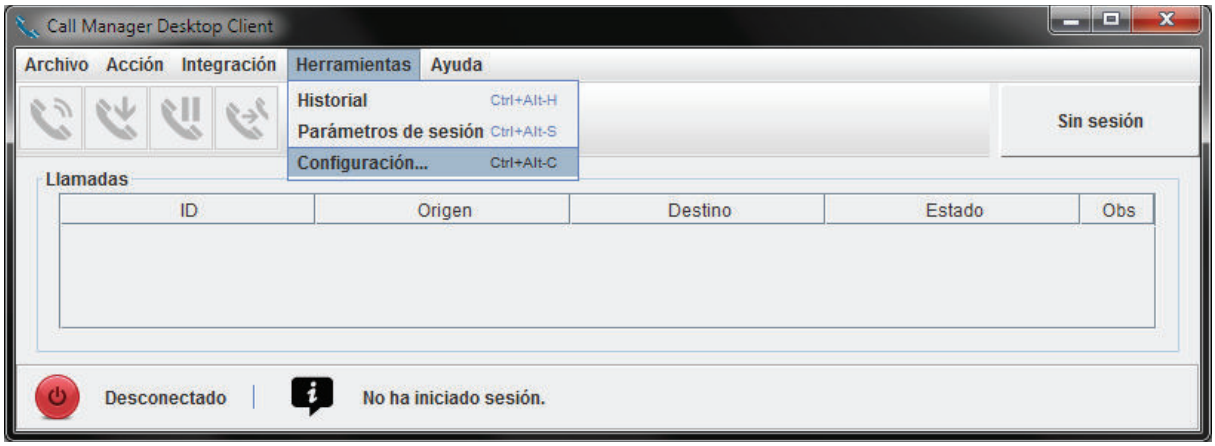


Ilustración D-11 Configuración de Parámetros de *Call Manager Desktop*

2. Ingresar la dirección IP y el número de puerto de escucha del servidor principal de la aplicación. Guardar los cambios y reiniciar el programa.

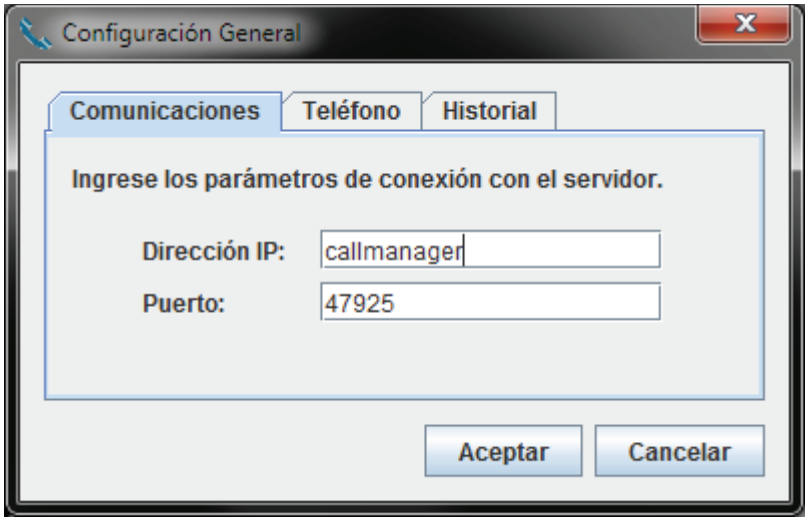


Ilustración D-12 Parámetros de Comunicación de *Call Manager Desktop*

3. Abrir nuevamente el aplicativo e iniciar sesión en el sistema haciendo clic en ***Archivo → Iniciar Sesión*** de la barra de menús.

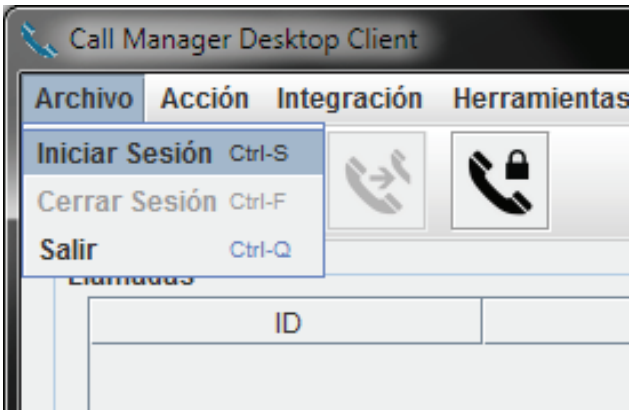


Ilustración D-13 Inicio de Sesión en *Call Manager Desktop*

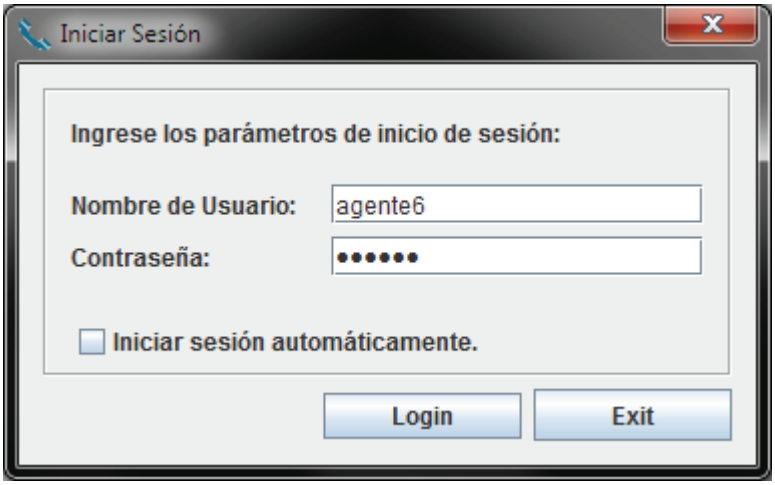


Ilustración D-14 Ingreso de Credenciales de Inicio de Sesión en *Call Manager Desktop*

ANEXO E
GUÍA DE CREACIÓN Y CARGA DE PLANES DE NUMERACIÓN

1. Descarga de planes de numeración oficiales

El primer paso consiste en acceder a la página de la Secretaría Nacional de Telecomunicaciones:

http://www.regulaciontelecomunicaciones.gob.ec/plan-tecnico-fundamental-de-numeracion-series-numericas/

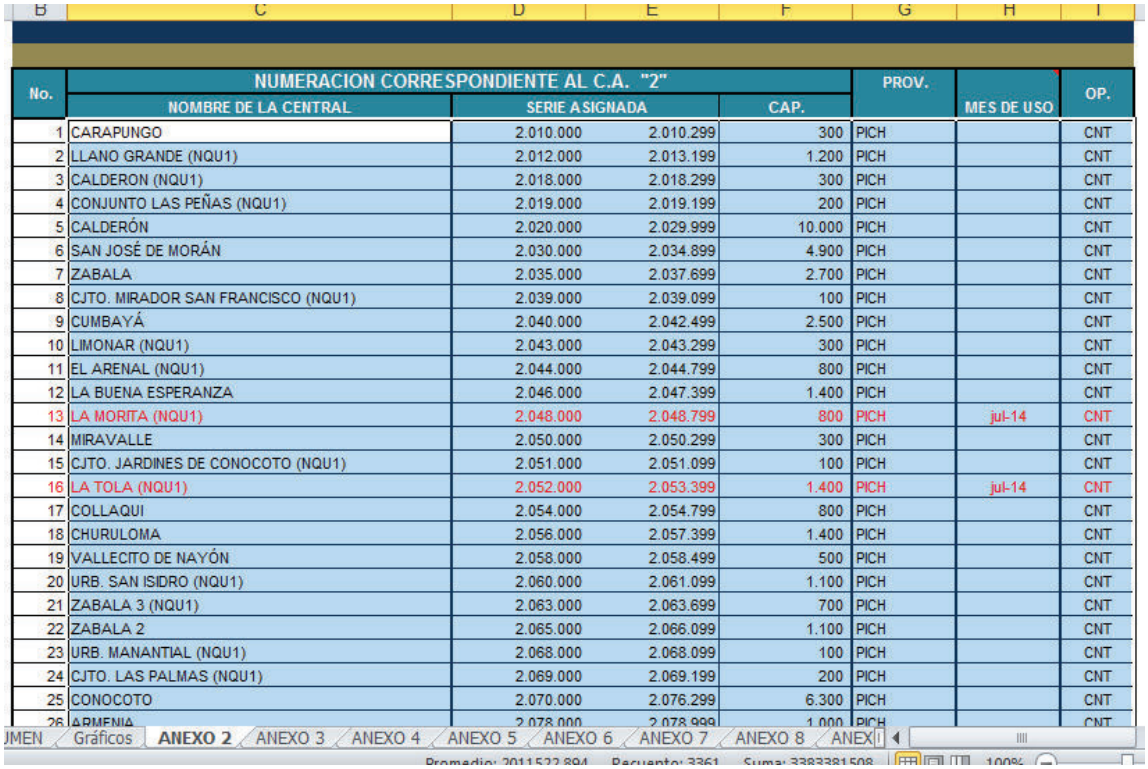
Acceder a la sección **Inicio -> Series Numéricas** y descargar los archivos de Informe de series de telefonía fija y SMA.

Ilustración E-1 Descarga de las Series Numéricas

2. Creación de archivos de tipo CSV

El siguiente paso consiste en la creación de los archivos de tipo valores separados por comas (.csv) según el RFC 4180.

En lo que corresponde a los planes de numeración de los números telefónicos fijos (numeración geográfica) se crearán archivos independientes según el código de área, por lo que cada pestaña de Anexo del plan de numeración de la ARCOTEL pasará a ser un archivo a cargar en el sistema. Como se muestra en la ilustración siguiente se seleccionará el contenido de las tablas: Nombre de la central, serie asignada, capacidad, provincia, mes de uso y operadora.



No.	NUMERACION CORRESPONDIENTE AL C.A. "2"			PROV.		OP.
	NOMBRE DE LA CENTRAL	SERIE ASIGNADA	CAP.		MES DE USO	
1	CARAPUNGO	2.010.000 2.010.299	300	PICH		CNT
2	LLANO GRANDE (NQU1)	2.012.000 2.013.199	1.200	PICH		CNT
3	CALDERON (NQU1)	2.018.000 2.018.299	300	PICH		CNT
4	CONJUNTO LAS PEÑAS (NQU1)	2.019.000 2.019.199	200	PICH		CNT
5	CALDERÓN	2.020.000 2.029.999	10.000	PICH		CNT
6	SAN JOSÉ DE MORÁN	2.030.000 2.034.899	4.900	PICH		CNT
7	ZABALA	2.035.000 2.037.699	2.700	PICH		CNT
8	CJTO. MIRADOR SAN FRANCISCO (NQU1)	2.039.000 2.039.099	100	PICH		CNT
9	CUMBAYÁ	2.040.000 2.042.499	2.500	PICH		CNT
10	LIMONAR (NQU1)	2.043.000 2.043.299	300	PICH		CNT
11	EL ARENAL (NQU1)	2.044.000 2.044.799	800	PICH		CNT
12	LA BUENA ESPERANZA	2.046.000 2.047.399	1.400	PICH		CNT
13	LA MORITA (NQU1)	2.048.000 2.048.799	800	PICH	jul-14	CNT
14	MIRAVALLE	2.050.000 2.050.299	300	PICH		CNT
15	CJTO. JARDINES DE CONOCOTO (NQU1)	2.051.000 2.051.099	100	PICH		CNT
16	LA TOLA (NQU1)	2.052.000 2.053.399	1.400	PICH	jul-14	CNT
17	COLLAQUI	2.054.000 2.054.799	800	PICH		CNT
18	CHURULOMA	2.056.000 2.057.399	1.400	PICH		CNT
19	VALLECITO DE NAYÓN	2.058.000 2.058.499	500	PICH		CNT
20	URB. SAN ISIDRO (NQU1)	2.060.000 2.061.099	1.100	PICH		CNT
21	ZABALA 3 (NQU1)	2.063.000 2.063.699	700	PICH		CNT
22	ZABALA 2	2.065.000 2.066.099	1.100	PICH		CNT
23	URB. MANANTIAL (NQU1)	2.068.000 2.068.099	100	PICH		CNT
24	CJTO. LAS PALMAS (NQU1)	2.069.000 2.069.199	200	PICH		CNT
25	CONOCOTO	2.070.000 2.076.299	6.300	PICH		CNT
26	ARMENIA	2.078.000 2.078.999	1.000	PICH		CNT

Ilustración E-2 Selección de Campos del Plan de Numeración, Sección Anexos

A continuación será creado un nuevo archivo de Excel en el que pegará la información anterior para luego borrar las columnas Capacidad y Mes de uso, de acuerdo a lo definido en la sección 2.4.3.5 de este Proyecto de Titulación.

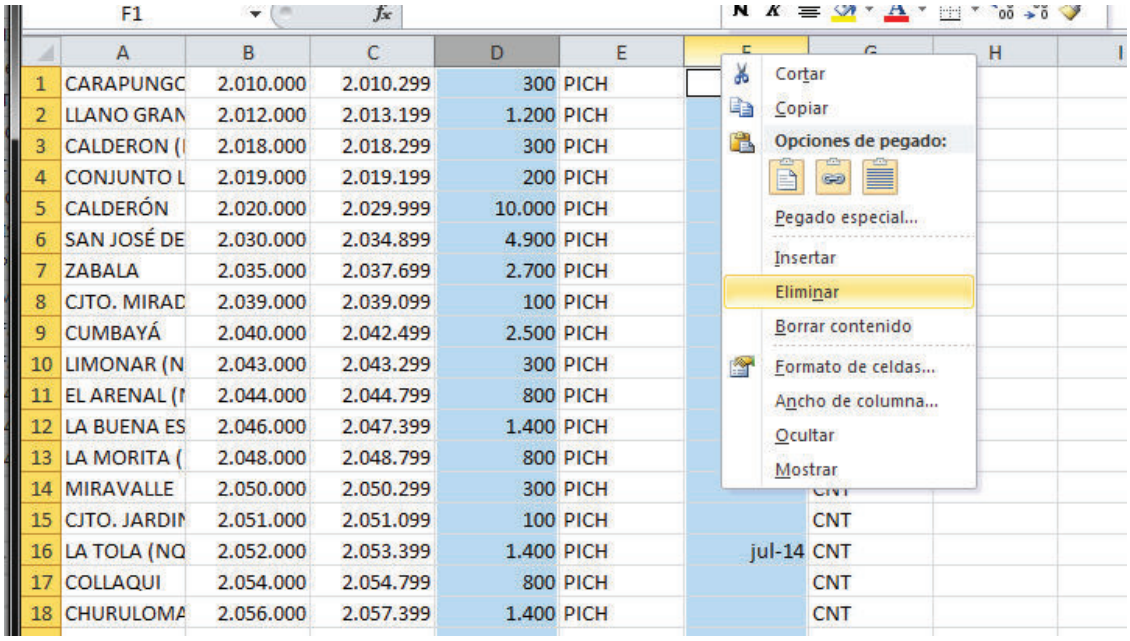


Ilustración E-3 Borrado de Columnas Innecesarias

Seguidamente las columnas correspondientes a la serie asignada tienen que ser formateadas de tal forma que sean consideradas de tipo texto, como se muestra en las ilustraciones siguientes:

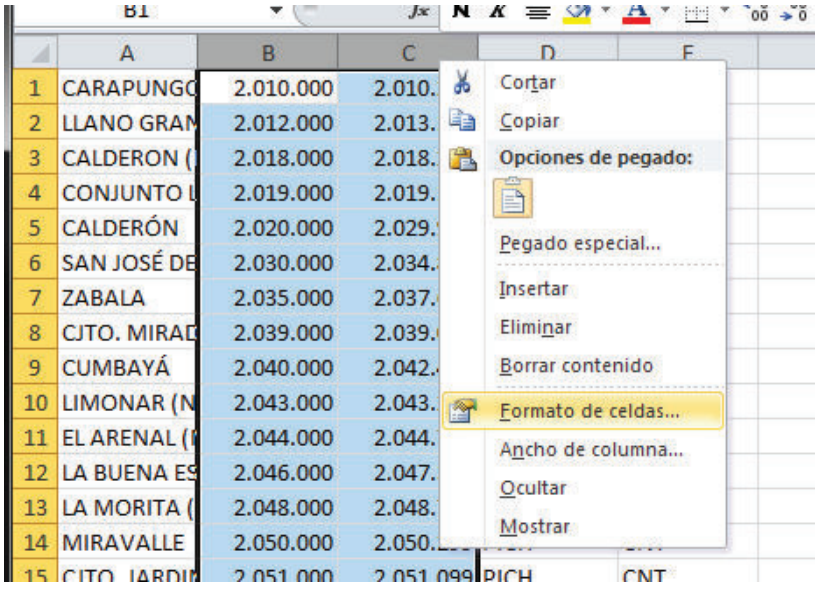


Ilustración E-4 Formato de Celdas para los Campos de la Serie Asignada

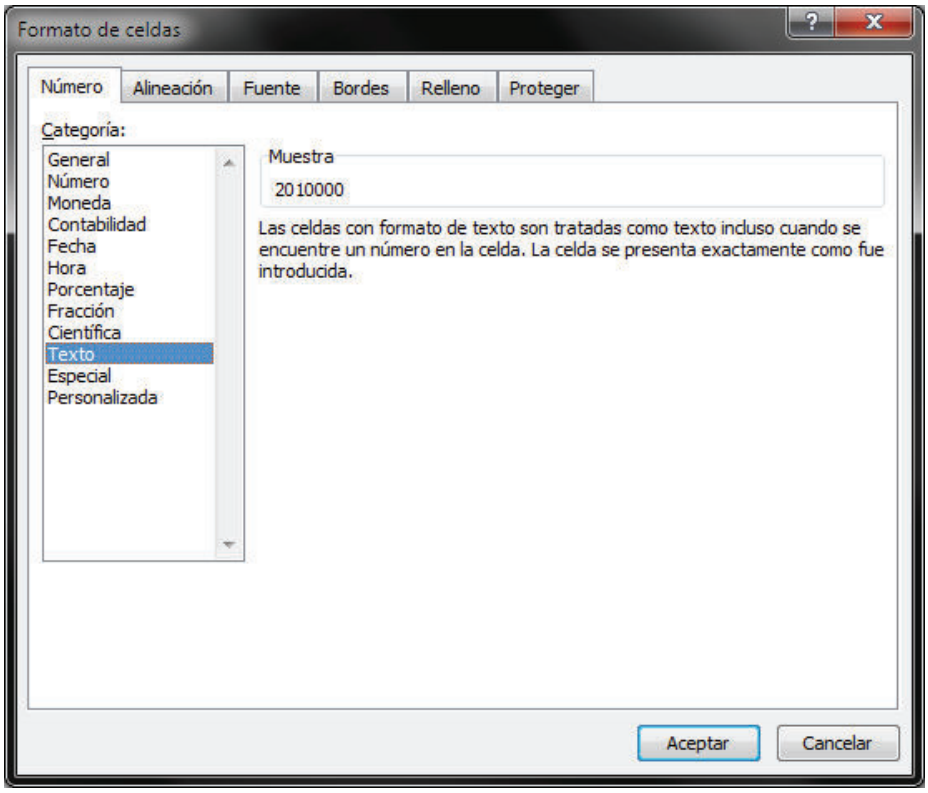


Ilustración E-5 Formato de la Serie Asignada de Tipo Texto

El siguiente paso consiste en establecer filtros en los campos serie asignada, provincia y operadora para borrar los datos innecesarios o incorrectos.

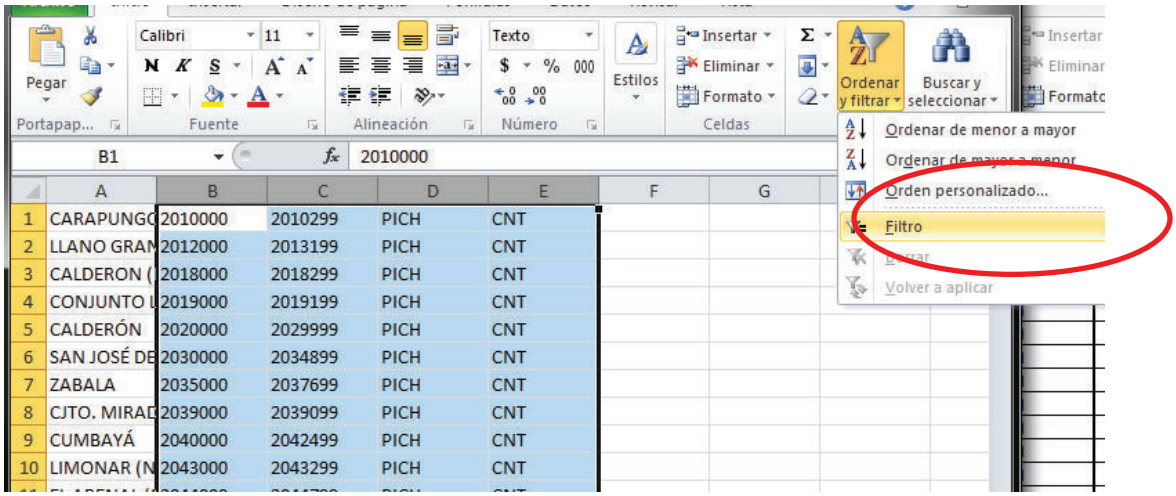


Ilustración E-6 Creación de Filtros en Campos serie, Operadora y Provincia

A continuación se borrarán todos los datos innecesarios y se corregirá la información inadecuada según las siguientes directrices:

- Las filas de la serie asignada que estén vacías serán borradas.

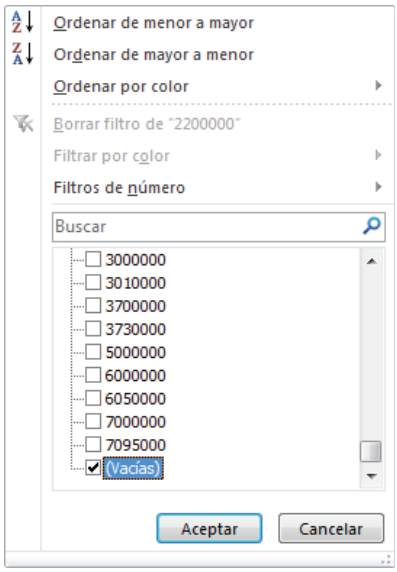


Ilustración E-7 Selección de Campos Vacíos de la Columna Serie Previa su Eliminación

- Los registros de la columna correspondiente a la provincia estarán sujetos al campo **Identificador** de la tabla E.1. Cualquier otro registro no señalado deberá ser cambiado para que coincida con el conjunto de valores de dicha tabla.

Identificador	Provincia
*	Campo desconocido/vacío/nulo
AZUAY	Azuay
BOLI	Bolívar
CAÑAR	Cañar
CARH	Carchi
CHIM	Chimborazo
COTX	Cotopaxi
EL ORO	El Oro
ESME	Esmeraldas

GALAP.	Galápagos
GUAYAS	Guayas
IMBA	Imbabura
LOJA	Loja
LOS RIOS	Los Ríos
MANABI	Manabí
MORONA	Morona Santiago
NAPO	Napo
ORELL	Orellana
PAST	Pastaza
PICH	Pichincha

Tabla E.1 Identificadores de Provincias

- Los campos correspondientes a la columna operadora seguirán la especificación de la tabla E.2.

Identificador	Operadora
*	Campo desconocido/vacío/nulo
C	Conecel S.A.
CNT	CNT
CP	Coripar S.A.
E	Etapa
ET	EcuadorTelecom. S.A.
ETT	EtapaTelecom S.A.
G	Globalnet S.A.
GC	Global Crossing S.A.
L	Linkotel S.A.
O	Otecel S.A.
S	Setel S.A.
T	Telecsa S.A.

Tabla E.2 Identificadores de Operadoras

Finalmente los archivos serán guardados con formato de tipo CSV (delimitado por comas) como se muestra en la ilustración posterior. Cada archivo a ser creado deberá contener como nombre ***zc_CODIGOAREA*** donde CODIGOAREA corresponde al trunk code de cada conjunto numérico.

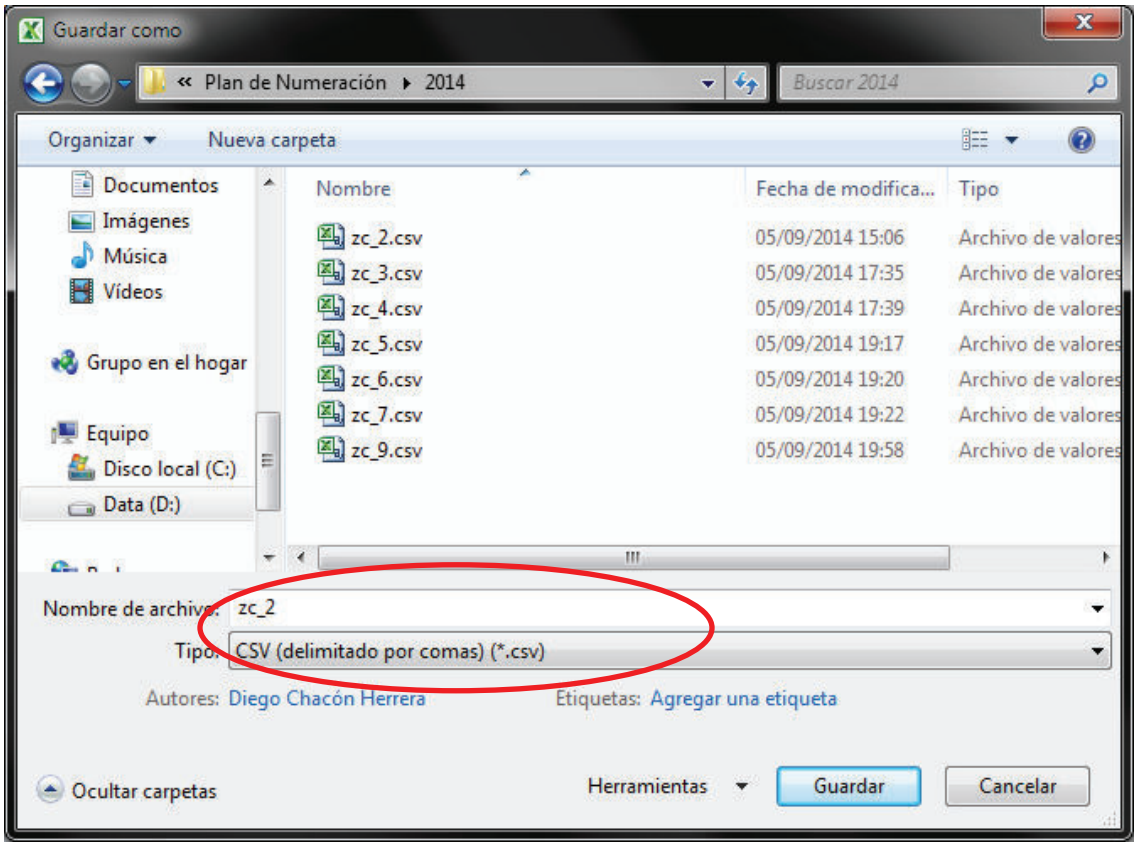


Ilustración E-8 Almacenamiento de Archivos CSV

La comprobación del correcto formato de los datos se ejemplifica en la imagen siguiente:

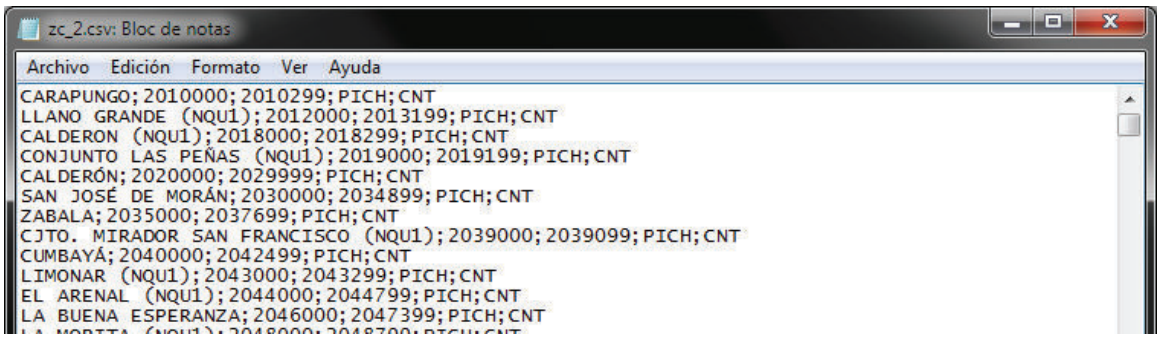


Ilustración E-9 Verificación de un Archivo de Tipo CSV

Aunque el sistema de numeración para la telefonía móvil sigue un tratamiento similar se requiere de pasos adicionales. Los campos a copiar desde el archivo original, como se muestra en la imagen siguiente, son: nombre del operador, serie asignada, capacidad, mes de uso y operadora (nótese que el campo provincia no existe).

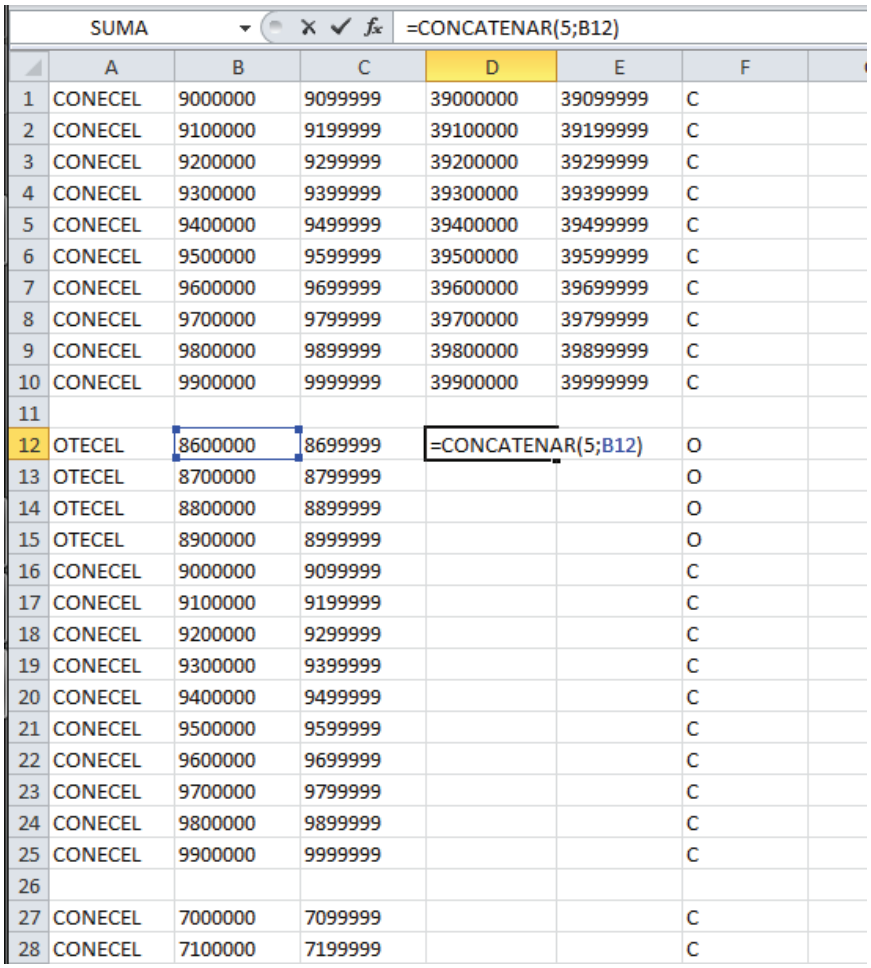
No.	NUMERACION CORRESPONDIENTE AL CODIGO DE RED. "97"				MES DE USO	OP.
	NOMBRE DEL OPERADOR	SERIE ASIGNADA		CAP.		
1	OTECEL	9.000.000	9.099.999	100.000		O
2	OTECEL	9.100.000	9.199.999	100.000		O
3	OTECEL	9.200.000	9.299.999	100.000		O
4	OTECEL	9.300.000	9.399.999	100.000		O
5	CONECEL	9.400.000	9.499.999	100.000		C
6	CONECEL	9.500.000	9.599.999	100.000		C
7	CONECEL	9.600.000	9.699.999	100.000		C
8	CONECEL	9.700.000	9.799.999	100.000		C
9	CONECEL	9.800.000	9.899.999	100.000		C
10	CONECEL	9.900.000	9.999.999	100.000		C

Ilustración E-10 Campos a Copiar del Plan de Numeración – Telefonía Móvil

Adicionalmente, debido a que el recurso numérico celular se encuentra separado en anexos (92, 93, 94, 95, 96, 97, 98 y 99), se copiarán los conjuntos numéricos uno después de otro como se puede ver en la ilustración 11. Además será necesario adjuntar el prefijo numérico a los campos rango asignado de cada conjunto según las pestañas (anexos) del plan de numeración. Esto quiere decir que si la pestaña Anexo 9**3** contiene la serie 9000000 - 9099999 ésta deberá ser cambiada a **3**9000000 – **3**9099999.

Lo anterior ha sido determinado con el propósito de crear un único archivo correspondiente a la numeración móvil que será cargado en el sistema. Por lo tanto, las columnas del archivo serán las siguientes:

- Columna 1: Nombre de la central
- Columna 2: Serie asignada (inicio)
- Columna 3: Serie asignada (fin)
- Columna 4: Símbolo asterisco (*)
- Columna 5: Identificador de la operadora

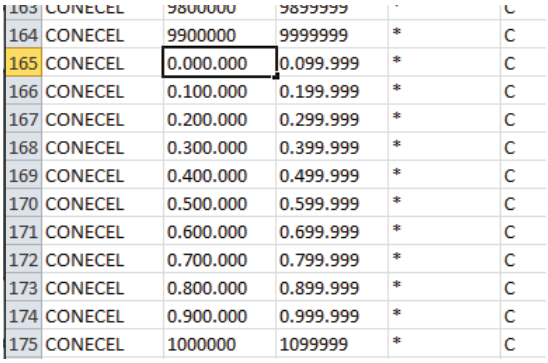


SUMA | =CONCATENAR(5;B12)

	A	B	C	D	E	F
1	CONECEL	9000000	9099999	39000000	39099999	C
2	CONECEL	9100000	9199999	39100000	39199999	C
3	CONECEL	9200000	9299999	39200000	39299999	C
4	CONECEL	9300000	9399999	39300000	39399999	C
5	CONECEL	9400000	9499999	39400000	39499999	C
6	CONECEL	9500000	9599999	39500000	39599999	C
7	CONECEL	9600000	9699999	39600000	39699999	C
8	CONECEL	9700000	9799999	39700000	39799999	C
9	CONECEL	9800000	9899999	39800000	39899999	C
10	CONECEL	9900000	9999999	39900000	39999999	C
11						
12	OTECEL	8600000	8699999	=CONCATENAR(5;B12)		O
13	OTECEL	8700000	8799999			O
14	OTECEL	8800000	8899999			O
15	OTECEL	8900000	8999999			O
16	CONECEL	9000000	9099999			C
17	CONECEL	9100000	9199999			C
18	CONECEL	9200000	9299999			C
19	CONECEL	9300000	9399999			C
20	CONECEL	9400000	9499999			C
21	CONECEL	9500000	9599999			C
22	CONECEL	9600000	9699999			C
23	CONECEL	9700000	9799999			C
24	CONECEL	9800000	9899999			C
25	CONECEL	9900000	9999999			C
26						
27	CONECEL	7000000	7099999			C
28	CONECEL	7100000	7199999			C

Ilustración E-11 Concatenación de Recurso Numérico – Telefonía Móvil

Finalmente debe tomarse en cuenta que los puntos (y/o separadores de miles/millones) deben ser eliminados del grupo serie asignada para que puedan ser apropiadamente procesados en el sistema.



164	CONECEL	9900000	9999999	*	C
165	CONECEL	0.000.000	0.099.999	*	C
166	CONECEL	0.100.000	0.199.999	*	C
167	CONECEL	0.200.000	0.299.999	*	C
168	CONECEL	0.300.000	0.399.999	*	C
169	CONECEL	0.400.000	0.499.999	*	C
170	CONECEL	0.500.000	0.599.999	*	C
171	CONECEL	0.600.000	0.699.999	*	C
172	CONECEL	0.700.000	0.799.999	*	C
173	CONECEL	0.800.000	0.899.999	*	C
174	CONECEL	0.900.000	0.999.999	*	C
175	CONECEL	1000000	1099999	*	C

Ilustración E-12 Rangos Numéricos no Deben Contener Separadores de Miles

3. Carga de archivos en el sistema

El procedimiento de almacenamiento de los planes de numeración es bastante sencillo, el cual comienza por el inicio de sesión del administrador del sistema y seleccionar el submenú ***Planes de Numeración*** como se aprecia en la ilustración siguiente.

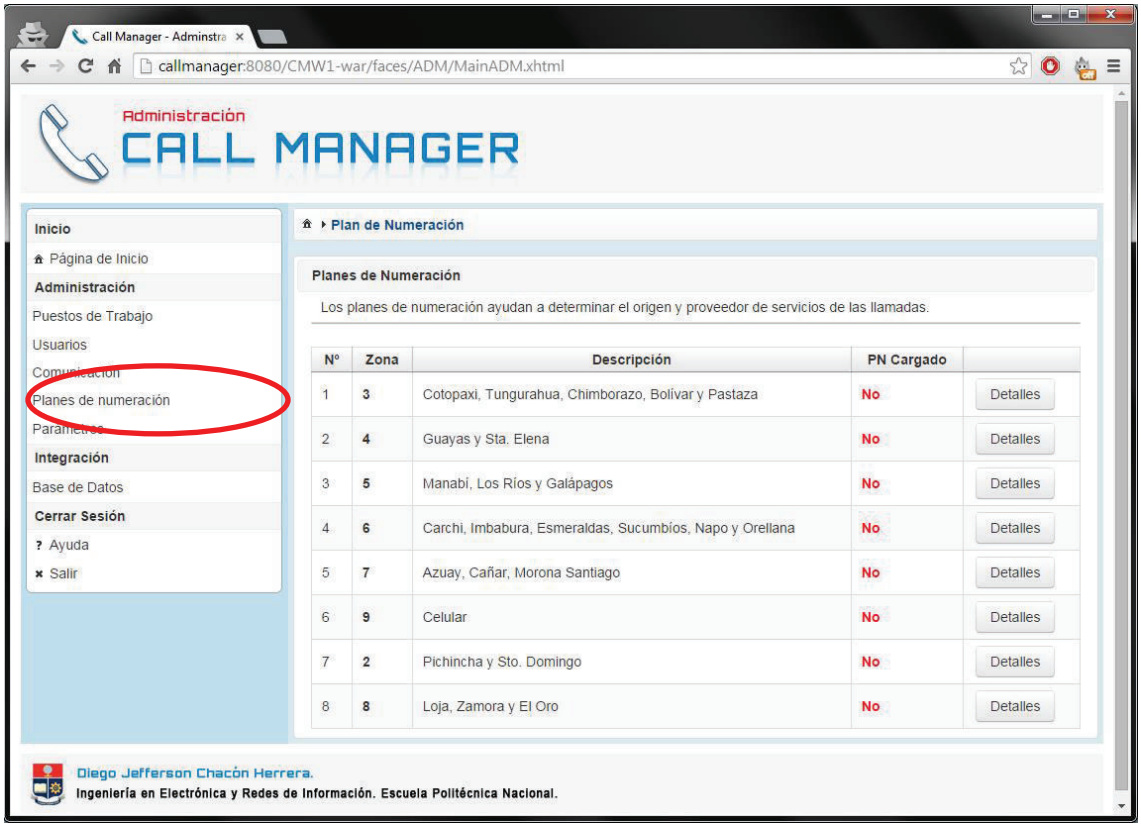


Ilustración E-13 Página de Administración de Planes de Numeración de *Call Manager*

A continuación se seleccionará el botón *Detalles* para acceder a cada zona numérica del plan de numeración y actualizar su información.

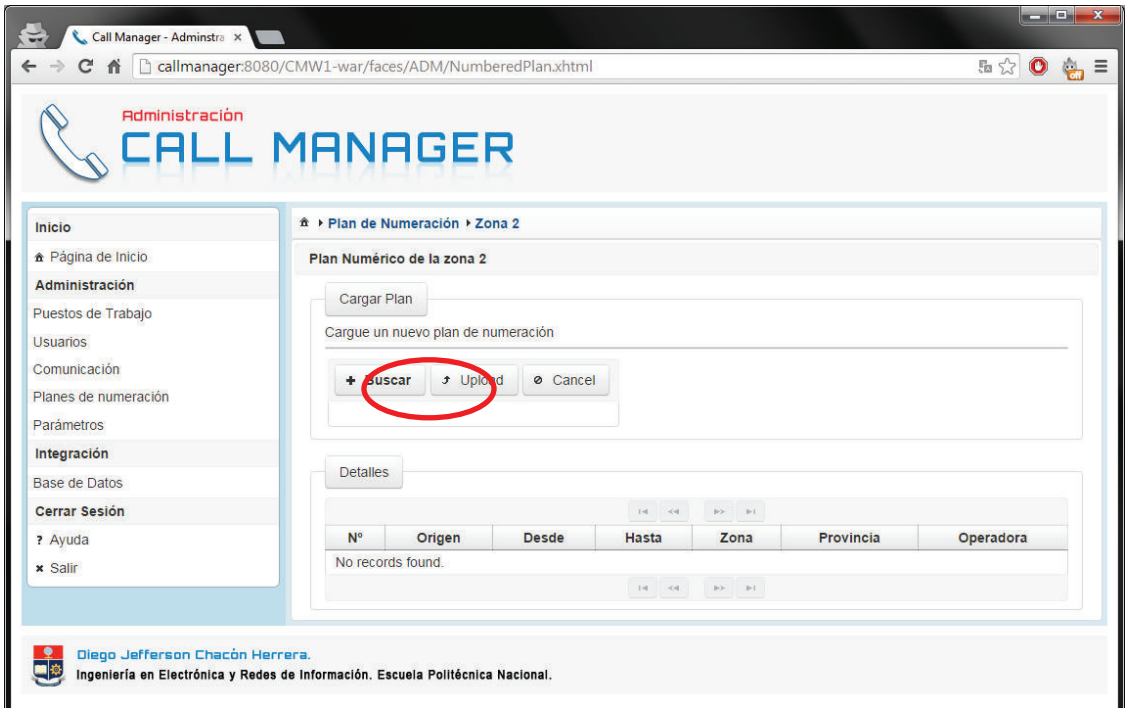


Ilustración E-14 Detalles del Plan Numérico – Zona 2

En caso que no existieran datos, se habilitará la opción de carga de la información a través de un archivo CSV. Para ello se deberá hacer clic en el botón **Buscar** seguido de la selección del correspondiente archivo de la zona en cuestión. Cuando éste haya sido elegido, la interfaz se actualizará de modo que la información procederá a ser guardada en la base de datos del sistema.

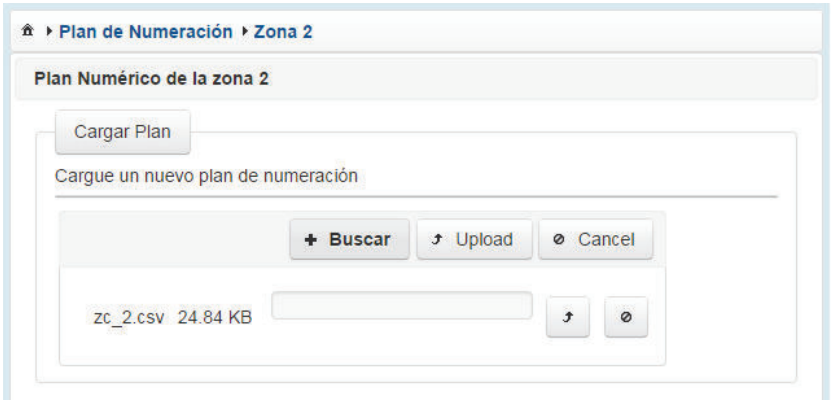
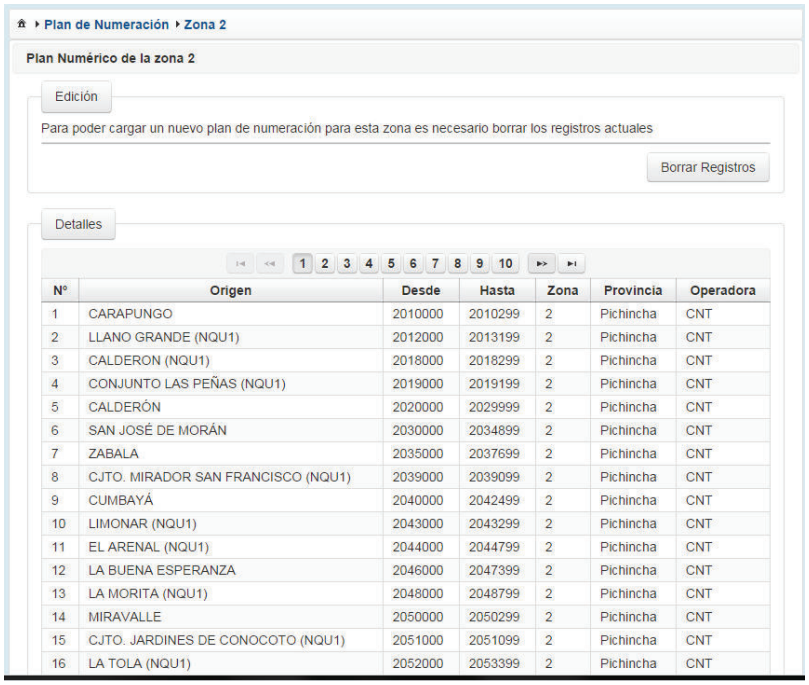


Ilustración E-15 Plan de Numeración a Ser Cargado en el Sistema

A continuación se dará clic en el botón ***Upload*** luego de lo cual la interfaz se actualiza para mostrar el recurso numérico correctamente ingresado en la

base de datos. El proceso deberá repetirse para cada código de área así como también la telefonía móvil.



N°	Origen	Desde	Hasta	Zona	Provincia	Operadora
1	CARAPUNGO	2010000	2010299	2	Pichincha	CNT
2	LLANO GRANDE (NQU1)	2012000	2013199	2	Pichincha	CNT
3	CALDERON (NQU1)	2018000	2018299	2	Pichincha	CNT
4	CONJUNTO LAS PEÑAS (NQU1)	2019000	2019199	2	Pichincha	CNT
5	CALDERÓN	2020000	2029999	2	Pichincha	CNT
6	SAN JOSÉ DE MORÁN	2030000	2034899	2	Pichincha	CNT
7	ZABALA	2035000	2037699	2	Pichincha	CNT
8	CJTO. MIRADOR SAN FRANCISCO (NQU1)	2039000	2039099	2	Pichincha	CNT
9	CUMBAYÁ	2040000	2042499	2	Pichincha	CNT
10	LIMONAR (NQU1)	2043000	2043299	2	Pichincha	CNT
11	EL ARENAL (NQU1)	2044000	2044799	2	Pichincha	CNT
12	LA BUENA ESPERANZA	2046000	2047399	2	Pichincha	CNT
13	LA MORITA (NQU1)	2048000	2048799	2	Pichincha	CNT
14	MIRAVALLE	2050000	2050299	2	Pichincha	CNT
15	CJTO. JARDINES DE CONOCOTO (NQU1)	2051000	2051099	2	Pichincha	CNT
16	LA TOLA (NQU1)	2052000	2053399	2	Pichincha	CNT

Ilustración E-16 Información Correctamente Almacenada en el Sistema

Finalmente la pantalla principal se actualizará para corroborar los cambios realizados.

N°	Zona	Descripción	PN Cargado	
1	8	Loja, Zamora y El Oro	No	Detalles
2	2	Pichincha y Sto. Domingo	Sí	Detalles
3	3	Cotopaxi, Tungurahua, Chimborazo, Bolívar y Pastaza	Sí	Detalles
4	4	Guayas y Sta. Elena	Sí	Detalles
5	5	Manabí, Los Ríos y Galápagos	Sí	Detalles
6	6	Carchi, Imbabura, Esmeraldas, Sucumbíos, Napo y Orellana	Sí	Detalles
7	7	Azuay, Cañar, Morona Santiago	Sí	Detalles
8	9	Celular	Sí	Detalles

Ilustración E-17 Planes de Numeración Actualizados

ANEXO F
GUÍA DE INTERCONEXIÓN CON BASES DE DATOS EXTERNAS

1. Los administradores de las bases de datos de las instituciones interesadas deben crear una tabla o vista que contenga, entre otros, obligatoriamente los siguientes campos:
    a. **Número telefónico**. El número telefónico debe anteponer el código de área para números telefónicos fijos y el código de red para números móviles. No se soportarán caracteres especiales o de país de ningún tipo.
    b. **Nombre o identificador del cliente**. Puede ser cualquier atributo que determine de forma inequívoca a cada cliente.
    c. **Latitud**. Utilizado en la georreferenciación de llamadas (puede ser nulo).
    d. **Longitud**. Utilizado en la georreferenciación de llamadas (puede ser nulo).

    Adicionalmente los administradores deben proporcionar un usuario que pueda acceder a la tabla o vista creada en modo de sólo lectura. Esta información será utilizada para acceder desde *Call Manager* en el transcurso de las llamadas telefónicas.

2. Ejecutar los pasos 1-7 de la sección "*Interconexión del Servidor de Aplicaciones con Bases de Datos*" del Anexo C (Despliegue del Servidor Principal de *Call Manager*) de este documento utilizando las credenciales de acceso a la BDD externa del paso 1. Debe enfatizarse que el recurso JDBC creado en *GlassFish* debe tener por nombre ***jdbc/ClientDatabaseJDBC*** al que debe asociarse el *pool* de conexiones de la BDD externa como se puede apreciar en la ilustración de la página siguiente. Una vez seleccionado el *pool* adecuado deben guardarse los cambios y se recomienda reiniciar el servidor de aplicaciones.

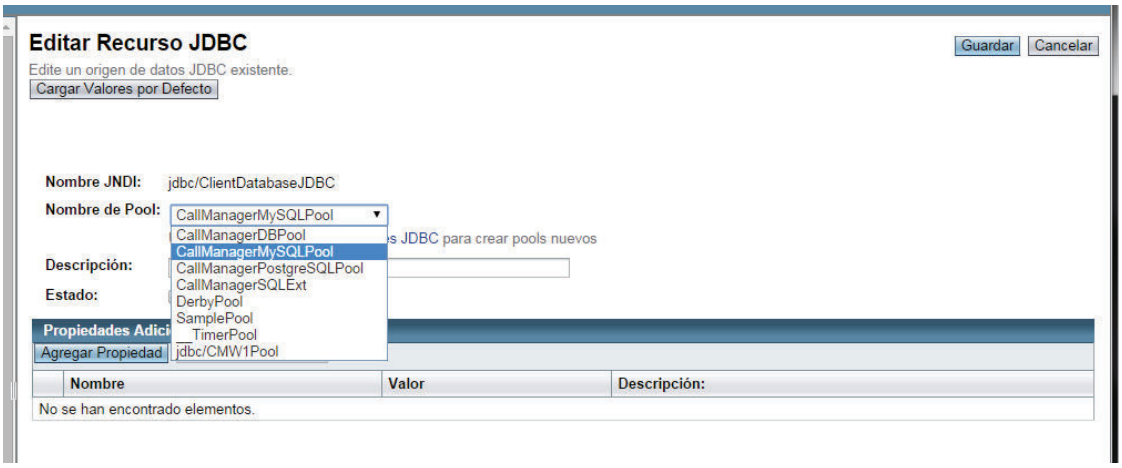


Ilustración F-1 Asociación del *Pool* de Conexiones con el Recurso JDBC

3. El siguiente paso está relacionado con la configuración del aplicativo *CallManager* para que éste pueda utilizar los recursos recientemente configurados en el servidor de aplicaciones. Para ello el administrador del aplicativo deberá acceder a la sección ***Integración → Bases de Datos*** del menú como muestra la ilustración posterior.

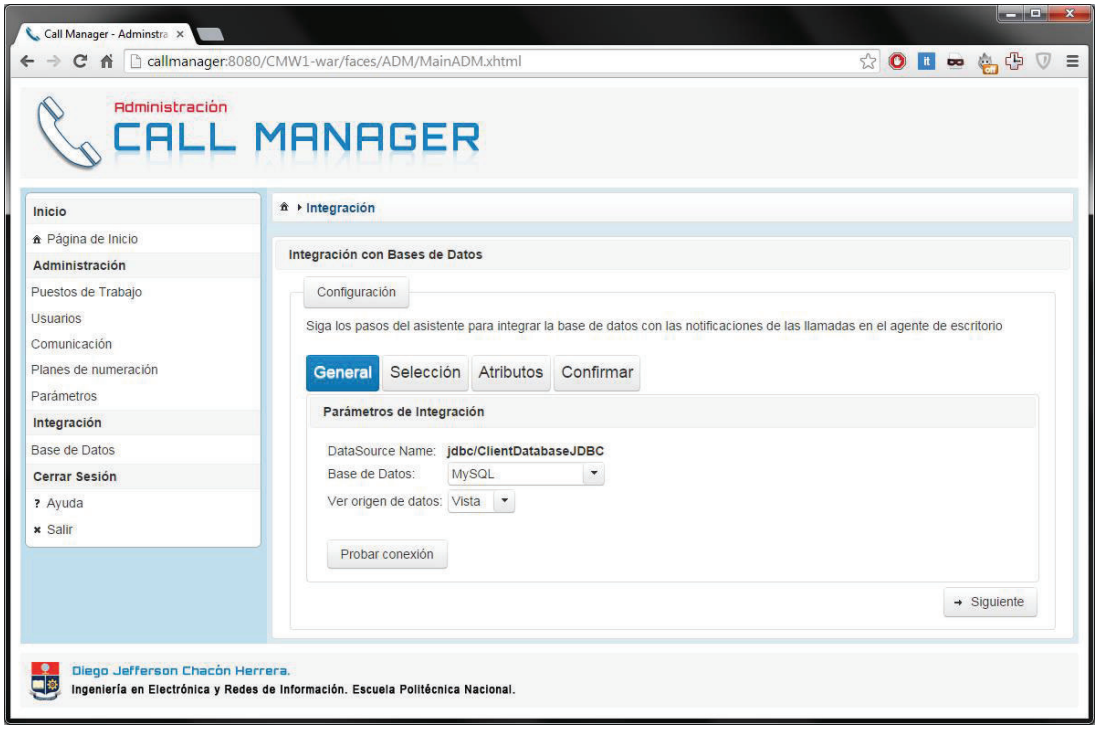


Ilustración F-2 Página de Integración con BDD Externas en *Call Manager*

4. Seleccionar el tipo de BDD a interconectar (MySQL, SQL Server ó PostgreSQL) así como el tipo de origen de datos (tabla o vista). Luego dar clic en ***Probar Conexión*** para comprobar el adecuado funcionamiento.

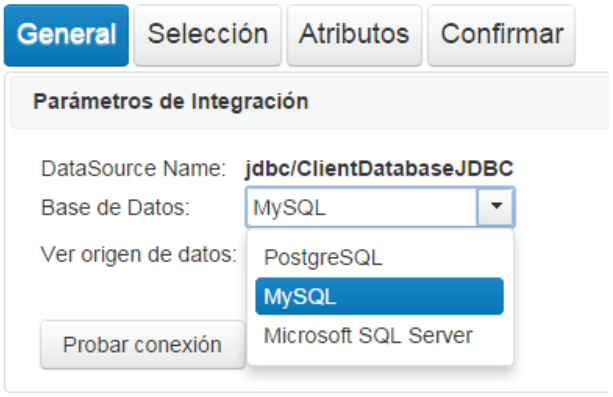


Ilustración F-3 Selección del Tipo de BDD en *Call Manager*

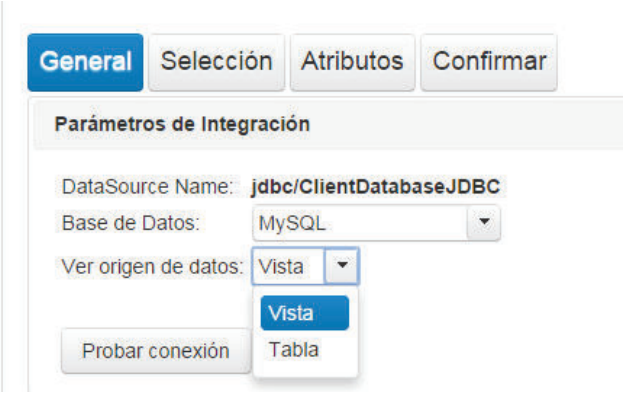


Ilustración F-4 Selección del Tipo de Origen de Datos en *Call Manager*

Ilustración F-5 Comprobación de los Recursos de Conexión con BDD Externas

5. Después de dar clic en el botón ***Siguiente*** del *wizard* será necesario elegir del catálogo de tablas o vistas creadas en la base de datos aquélla que contenga la información de los clientes a utilizar en *Call Manager*.

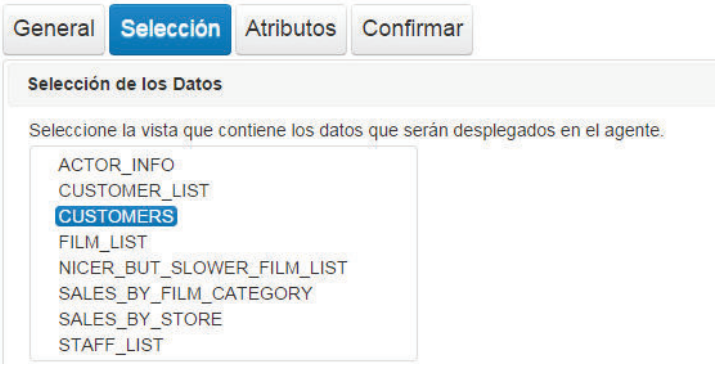


Ilustración F-6 Selección de la Vista o Tabla a Utilizar en *Call Manager*

6. Al dar clic en el botón ***Siguiente*** se presentará la sección ***Atributos*** en la cual se deberá mapear adecuadamente las columnas de la tabla o vista elegida con los campos que serán utilizados en *Call Manager* para determinar el número telefónico, la latitud y la longitud de los contactos. Cualquier campo no obligatorio (ver paso 1) deberá ser identificado como tipo *Información*. Se recuerda que los campos: *identificador, teléfono, latitud y longitud,* son obligatorios para poder continuar con el procedimiento de configuración.

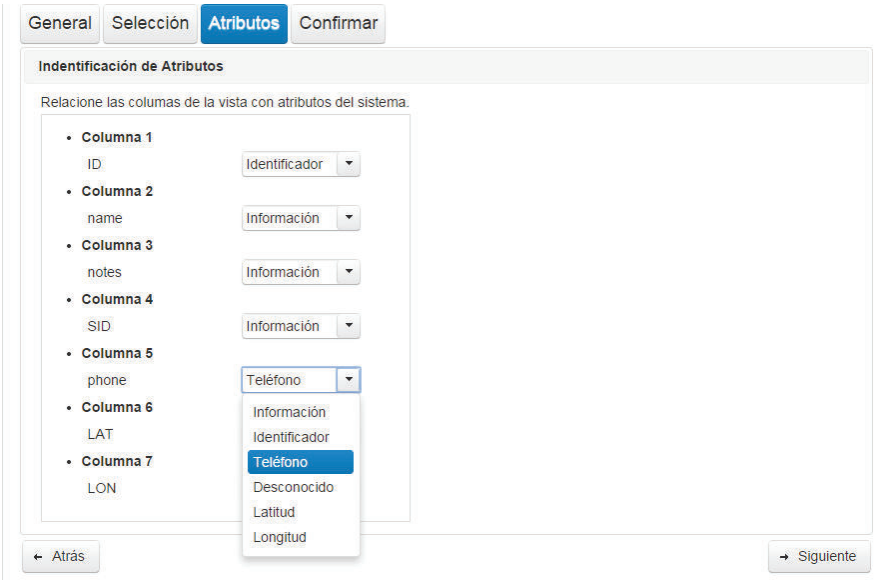


Ilustración F-7 Identificación de Atributos

7. El flujo final muestra en pantalla un resumen de todos los parámetros seleccionados previo su almacenamiento en la base de datos interna de la solución. De esta manera el servidor principal será capaz de recuperar información adicional de los clientes desde bases de datos externas en el transcurso de las llamadas.

Ilustración F-8 Confirmación de Parámetros de Interconexión

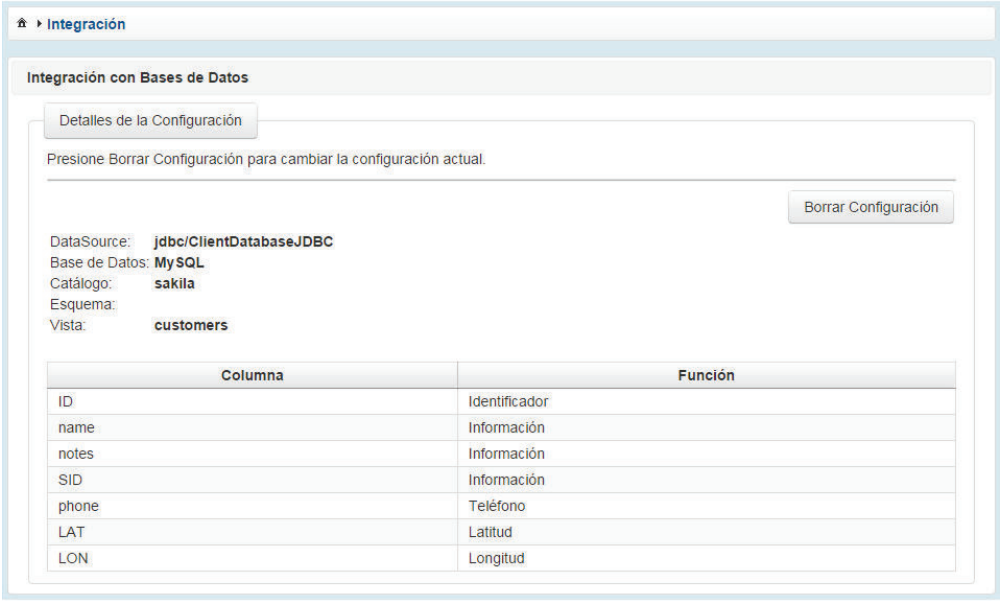


Ilustración F-9 Cambios Almacenados en el Sistema

ANEXO G

PROFORMA DE AVAYA IP OFFICE *CONTACT CENTER*

OFERTA Nº	**2015-MAR-06-71**		**LYNXSOURCE CIA. LTDA.**		
DIRIGIDA A	**DIEGO CHACÓN**		**RUC:**	**1792154073001**	
ELABORADO POR	**JOSE CONDE - LYNX**				
FECHA	**06/03/2015**				

CANTIDAD	# DE PARTE	DESCRIPCION		PRECIO UNITARIO	PRECIO TOTAL	TIEMPO DE ENTREGA
IP OFFICE CONTACT CENTER						
LICENCIAS						
1	308394	IPO R9+ IPOCC BASE IP500 V2 LIC	Licencia Base IP Office Contact Center Multicanal, incluye sistema de grabación de llamadas con capacidad para 5 agentes.	2.050,00	2.050,00	**45 días**
1	308397	IPO R9+ IPOCC SPV LIC	Licencia de supervisor de contact center. Incluye capacidad para ser agente multicanal (voz, email, chat).	650,00	650,00	
5	339499	IPO R9+ IPOCC VCE AGT LIC	Licencia de agente de voz de IP Office Contact Center	340,00	1.700,00	
5	308395	IPO R9+ IPOCC MULTI CH AGT LIC	Licencia multicanal (voz, email, chat) para agente de IP Office Contact Center	251,00	1.255,00	
EQUIPOS TERMINALES						
5	700505424	IP PHONE 9608G GRY	Teléfono IP AVAYA modelo 9608G - 2 puertos 10/100/1000; 24 botones administrables, PoE.	250,00	1.250,00	**45 días**
5	64338-31	HW251N SUPRAPLUS HEADSET WIDEBAND MONAURAL W/NOISE CANCELLING	Diadema Plantronics modelo HW251N **monoaural** con cancelación de ruido.	120,00	600,00	
5	72442-41	HIS,ADAPTER CABLE	Adaptador para la conexión a teléfonos AVAYA	31,00	155,00	
			TOTAL (No incluye IVA)		**7.660,00**	
			TOTAL (Incluye 12% IVA)		**8.579,20**	

REQUERIMIENTOS:	**IP OFFICE CONTACT CENTER** Sistema Operativo: Microsoft ® Windows Server 2008 R2 Standard 64–bit Edition SP1, con licencia válida. - Microsoft ® Windows Server 2012 R2 Standard 64–bit Edition, con licencia válida. Procesador: Intel Xeon E3 Quadcore de 3.1 GHz Memoria: RAM 8 GB Disco Duro: 250 GB Media: DVD-ROM drive Red: 1 NIC de 1GB También es Soportado en un entorno de servicio virtualizado como VMware con las características anteriores.
	CONTACT RECORDER - Sistema de grabación Sistema Operativo Microsoft®Windows Server 2008 R2 Standard 64–bit Edition SP1, con licencia válida. Microsoft®Windows Server 2012 R2 Standard 64–bit Edition, con licencia válida. Procesador: Intel Dual Core de 2 GHz Memoria: RAM 4 GB Disco Duro: 250 GB Media: DVD-ROM drive Red: 1 NIC de 1GB También es Soportado en un entorno de servicio virtualizado como VMware con las características anteriores.
	ESTACIONES DE TRABAJO - AGENTES / SUPERVISOR Sistema Operativo Microsoft® Windows 7, o Windows 8.1 Procesador: Intel Pentium 4 processor 2.2 GHz o superior. Memoria: RAM 4 GB Disco Duro: 20 GB Buscador: Microsoft ® Internet Explorer 8.x o superior higher; Mozilla Firefox 3.6 o superior. Red: 1 tarjeta de Red

FORMA DE PAGO	**60% anticipado; 40% contraentrega**
VALIDEZ DE LA OFERTA	**15 DÍAS**

Atentamente,

LYNX SOLUCIONES TECNOLÓGICAS

José Conde
LYNXSOURCE CIA. LTDA.