UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

PROYECTO FIN DE CARRERA

Desarrollo software para control de tráfico telefónico por medio del protocolo SMDR

TITULACIÓN: Ingeniería Técnica de Telecomunicación. Especialidad de Telemática.

TUTORES: Carlos M. Ramírez Casañas y Adán San Gil Martín.

AUTOR: Juan Francisco Rodríguez Báez.

FECHA: Junio 2015.

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

PROYECTO FIN DE CARRERA

Desarrollo software para control de tráfico telefónico por medio del protocolo SMDR

PRESIDENTE: **SECRETARIA/O:** **VOCAL:**

TUTORES: **AUTOR:**

NOTA: _____

TITULACIÓN: Ingeniería Técnica de Telecomunicación. Especialidad de Telemática.

TUTORES: Carlos M. Ramírez Casañas y Adán San Gil Martín.

AUTOR: Juan Francisco Rodríguez Báez.

FECHA: Junio 2015.

A mi tío Agustín, por introducirme en el mundo de la electrónica.

A mi madre.

A Inés.

Ellos me han traído hasta aquí.

Agradecimientos

Este Proyecto Fin de Carrera no se habría podido realizar sin la colaboración y apoyo de los tutores, Carlos Ramírez Casañas y Adán San Gil Martín cuya labor ha sido muy valiosa para la consecución de los objetivos propuestos. A ellos y a todas las personas que me han ayudado de una u otra forma:

Muchas gracias.

ÍNDICE.

Capítulo 1

Introducción

Las comunicaciones de voz hasta hace poco tiempo sólo eran posibles a través de la Red Pública Telefónica, basada en nodos de conmutación de circuitos soportada en centrales telefónicas, es decir, se necesitaba establecer un circuito exclusivo o dedicado entre los nodos antes de que los usuarios se pudieran comunicar. Con el avance tecnológico, gran parte de los circuitos analógicos dejaron paso a los elementos digitales [Landivar 2009]. Sin embargo la aparición de redes de conmutación de paquetes, como es el caso de Internet, hicieron que se comenzara a investigar sobre modelos orientados a enviar la voz a través de una red de este tipo, denominando a este concepto Voz sobre IP (VoIP).

La VoIP se ha adoptado de forma amplia en el mundo empresarial, ya que integra en la misma tanto voz como datos, con el consecuente ahorro de costes. Según Silicon News [Silicon 2013], el nivel de adopción de la Telefonía IP a nivel mundial va en aumento, significando en 2012 que el 30% de las llamadas han sido realizadas a través de Skype. Pero las empresas que disponen de estas infraestructuras requieren de una información detallada y fidedigna del tráfico en su red, una auditoría constante que dé información de cara a la toma de decisiones para optimizar la gestión y aminorar costes, un asunto más relevante que nunca a día de hoy. Más adelante se detalla la tecnología relativa a VoIP.

1.1. Objetivos.

El objetivo principal de este Proyecto Fin de Carrera (en adelante PFC) es el de desarrollar una herramienta de análisis del tráfico telefónico basada en el estudio de archivos SMDR generados por centralitas IP Avaya. Este *software* permitirá auditar el tráfico en centralitas digitales del fabricante Avaya, existentes en el laboratorio de Transmisión por Línea de la Escuela de Ingeniería de Telecomunicación y Electrónica (EITE) de la Universidad de Las Palmas de Gran Canaria (ULPGC). Dada la incorporación de este tipo de centralitas de forma intensiva en el mundo empresarial, es de gran importancia el estudio, análisis y desarrollo de este protocolo para su uso en el entorno que

nos rodea, así como para tomar decisiones respecto a una respuesta adecuada en determinados escenarios.

Como objetivos particulares se establecen los siguientes:

- Analizar los fundamentos de la tecnología VoIP.
- Definir los elementos básicos en el desarrollo de aplicaciones prácticas relativas a la tecnologóa VoIP.
- Conocer el estado del arte, incluyendo el estudio de las centralitas Avaya y equipos relacionados.
- Entender el protocolo SMDR, las particularidades de su formato y los resultados que se derivan de su estudio.
- Desarrollar *software* que evalúe tráfico entrante en centralitas que utilicen el estándar SMDR y que permita identificar problemáticas existentes o potenciales.
- Identificar parámetros críticos en el análisis de tráfico entrante en centralitas que utilizan tecnología IP que permitan interpretar datos de forma rápida e intuitiva.
- Calcular potenciales amenazas en entornos laborales donde el tráfico telefónico entrante es esencial.
- Comparar los productos comerciales existentes con los que propone el presente proyecto.
- Proponer desarrollos y mejoras sobre el presente proyecto.

De forma específica, este proyecto pone énfasis en la necesidad de evaluar el tráfico entrante, en lugar del saliente o interno, que es lo habitual[1]. Esto permite un análisis de las carencias de la centralita instalada a efectos de dar mejor servicio al tráfico entrante, especialmente en escenarios especialmente sensibles, como pueden ser los servicios de urgencias sanitarias, policía, bomberos o cualesquier otro centro de trabajo que busque una mejor eficiencia en el tratamiento al cliente/usuario.

[1] La mayoría de las aplicaciones informáticas evalúa costes relativos a las llamadas realizadas por el personal de una empresa, así como el tiempo dedicado al tráfico saliente e interno.

1.2. Fases de desarrollo.

Esta herramienta *software* de análisis de la información de tráfico telefónico entrante permite conocer no sólo detalles de cada llamada de cara a una mejora en la calidad del servicio, sino que ayuda en la adopción de decisiones para la optimización de recursos y, por ende, una minoración de los tiempos de respuesta en centros de trabajo donde este factor sea decisivo.
Para el desarrollo del presente proyecto se han seguido las siguientes fases:

- Documentación previa.
- Análisis de requisitos.
- Diseño global.
- Implementación.
- Pruebas.

1.3. Estructura de la memoria.

Esta memoria se estructura de la siguiente forma:
El capítulo 1, la introducción, incluye la definición de los elementos que se propone abordar en este PFC.
El capítulo 2, fundamentos, define las bases tecnológicas en las que se soporta el presente proyecto. Una somera introducción a la tecnología de voz sobre IP y una referencia sobre la teoría de colas introducen, por un lado, en la tecnología que da pie al presente desarrollo y, por otro, los elementos matemáticos que deben dar pie a la toma de decisiones en base a los datos aportados por la aplicación informática.
El capítulo 3 se centra en el estado del arte, esto es, el conjunto de equipos (centralita y teléfonos) que deben formar parte de un entorno laboral del que se va a evaluar su tráfico entrante. Los equipos reseñados en esta memoria forman parte del equipamiento del Laboratorio de Transmisión por Línea de la EITE-ULPGC.
El capítulo 4 expone un pormenorizado estudio del SMDR y los mecanismos de descubrimiento asociados.

El capítulo 5 define en detalle la aplicación informática de estudio de tráfico entrante en base a datos SMDR suministrados. Adicionalmente se hace referencia a una aplicación para la creación de ficheros SMDR pseudoaleatorios de gran ayuda en este proyecto[2].

El capítulo 6 propone una serie de conclusiones redactadas en término de propuestas y sugerencias para potenciales futuros proyectos, ya que en el desarrollo del presente se ha observado el potencial existente en la información suministrada por los ficheros SMDR en lo referente a la evaluación de tráfico entrante en entornos críticos.

El capítulo 7 detalla la bibliografía, el pliego de condiciones y el presupuesto.

Por último, en el apartado de *Anexos,* se incluye información complementaria ilustrativa del presente PFC, como son: herramientas matemáticas relacionadas con el análisis de colas; el formato de archivo de registro de llamadas del otro gran competidor de Avaya en el mundo de la telefonía IP, Cisco, así como listados del código de las aplicaciones desarrolladas.

1.4. Medios y recursos empleados.

Equipos.

A continuación se detalla el equipamiento informático utilizado para dar soporte a las herramientas de *software* utilizadas. De forma detallada, el equipamiento utilizado ha sido:

- Ordenador de sobremesa con microprocesador Intel® S3 marca Acer modelo Aspire X3950 con 4GB de RAM y 300GB de disco duro.
- Ordenador portátil Acer Aspire S5 con Microsoft Windows® 7 Home Premium, procesador Intel Core i5-3317U Dual-core 1,70 GHz, 4 GB, DDR3 SDRAM y 128 GB de disco duro de estado sólido.

[2] La normativa actual, desarrollada al amparo de la Ley Orgánica de Protección de Datos, ha sido determinante en la disposición de datos reales de tráfico en centralitas instaladas en entornos colaborativos. Por esta razón, ha sido necesario disponer de una herramienta de simulación de ficheros SMDR de gran tamaño.

- Dos monitores Philips de 23 pulgadas.
- Impresora Brother DCP-7065DN.
- Impresora Brother HL-L8250CDN.

Software.

Entre los recursos materiales utilizados para la realización de este proyecto, se incluyen la herramienta *software* de desarrollo Lazarus, los paquetes de ofimática utilizados para la redacción de la memoria, y el sistema operativo bajo el que se ejecutó el trabajo:

- Lazarus IDE v 1.4.0.
- Microsoft Office® 2003.
- Microsoft Windows® 7.

Capítulo 2

Fundamentos

En este capítulo se realiza un somero análisis de los elementos en los que se basa la tecnología de voz sobre IP, así como los fundamentos matemáticos utilizados y las aplicaciones prácticas de la tecnología en la que se apoya para su desarrollo.

2.1. Voz sobre IP

La Voz sobre IP (que en adelante denominaremos VoIP, acrónimo en inglés de *Voice over* IP), [Davidson, 2001] es un grupo de recursos que hacen posible que la señal de voz humana viaje a través de Internet empleando el Protocolo de Internet (en adelante IP, Internet Protocol). Esto permite la transmisión de voz en forma digital, siguiendo la agrupación de información en paquetes de datos.

Una de las principales ventajas de las redes que soportan VoIP es que permiten la integración del tráfico de voz y del tráfico de datos sobre una misma infraestructura de red, dando lugar como primer paso en la evolución hacia las verdaderas redes multiservicios (voz, datos y vídeo) [Huidobro 2006].

Debe diferenciarse entre VoIP y Telefonía sobre IP. La primera es el conjunto de normas, dispositivos y protocolos, esto es, la tecnología que permite comunicar voz sobre el protocolo IP. Sin embargo, la telefonía sobre IP es el servicio telefónico disponible al público realizado con tecnología de VoIP. En este apartado del proyecto queda definido el protocolo (VoIP) para, a continuación, desarrollar una aplicación basada en el uso (telefonía).

La fuente (persona que llama) establece y origina el proceso (la llamada de voz). Esta información se codifica, se empaqueta siguiendo el estándar IP y, de la misma forma, se decodifica y reproduce en destino.

La justificación del uso de VoIP viene dada por el abaratamiento de las comunicaciones (especialmente las de mayor coste, las internacionales) y por la mejora de la comunicación tanto externas como internas en grandes corporaciones y servicios públicos. Pero también permite un auditado de datos excelente, que en este PFC tiene una aplicación inmediata, a la par que práctica.

En lo que concierne a las desventajas de esta tecnología, cabe mencionar en primer lugar el hecho de que no queda garantizada la recepción de los paquetes de datos y, por otro lado, que no ofrezca el mismo grado de fiabilidad que la PSTN[3]. Esto se debe a diversas razones, como puede ser la complejidad por el uso de múltiples protocolos, la falta de estandarización de los proveedores de equipos y de servicios, además de utilizar diferentes sistemas operativos y sistemas de gestión de red que implican la no existencia de interoperabilidad extremo a extremo [Verma 2011].

2.1.1. Centralitas VoIP.

Las centralitas VoIP son el núcleo del sistema o red VoIP. Éstas generan, entre otros usos, los ficheros con información del tráfico, objeto de estudio en este proyecto. Los datos generados permiten hacer un seguimiento de las operaciones realizadas a efectos de contabilidad, pero también la identificación de problemas críticos derivados del tráfico entrante en momentos determinados, lo que permite una toma de decisiones en cuanto a ampliación de los equipos e incluso de los recursos humanos destinados en un servicio de emergencias, bomberos, policía u otros escenarios laborales donde la atención telefónica debe ser gestionada de forma precisa.

2.1.2. Protocolos de VoIP.

VoIP utiliza diversos protocolos para el diálogo entre dispositivos. Cabe destacar:

- H.323 (Protocolo definido por la ITU-T[4]),
- SIP (definido por la IETF[5]),
- UNIStim (Protocolo propiedad de Nortel Avaya),
- IAX (Protocolo original para la comunicación entre PBXs),
- Asterisk y

[3] *Public Switched Telephone Network* o Red Telefónica Conmutada.

[4] Unión Internacional de Telecomunicaciones.

[5] *Internet Engineering Task Force*.

- Skype (Protocolo propietario *peer-to-peer* utilizado en la aplicación Skype).

2.1.2.1. Estándar VoIP (H.323).

Definido en 1996 por la Unión Internacional de Telecomunicaciones (UIT), H.323 proporciona una serie de normas y estándares para la industria en cuanto a la VoIP. Estas definiciones permiten controlar el tráfico de la red, aminorando las posibilidades de pérdidas en el rendimiento. Además, VoIP es:

- Independiente del tipo de red física que lo soporta.
- Permite la integración en grandes redes IP.
- Es independiente del *hardware* utilizado.
- Permite ser implementado tanto en *software* como en *hardware.*
- Permite la integración de Vídeo (p. ej. cámaras) y TPV (terminales de punto de venta).
- Proporciona un enlace a la red de telefonía tradicional (integrando equipos que no son VoIP con los que sí lo son).

2.1.2.2. Estándar VoIP (SIP[6]).

El Protocolo de Inicio de Sesión es un protocolo reciente que es en la actualidad el mayormente utilizado. SIP es un protocolo desarrollado por el IETF para convertirse en el estándar para la iniciación, modificación y finalización de sesiones de usuario donde intervienen elementos tales como el vídeo, la voz o la mensajería instantánea. Con una sintaxis similar a HTTP, SIP nace con el objeto de que la telefonía se vuelva un servicio más en Internet. SIP es uno de los protocolos de señalización para VoIP, al igual que H323 y el IAX2 y se vale de funciones aportadas por otros protocolos para su funcionamiento. Así, SIP se centra en el establecimiento, modificación y terminación de las sesiones, demostrando ser un protocolo de señalización. Adicionalmente, SIP se complementa con el protocolo RTP[7], encargado del

[6] *Session Initiation Protocol* (Protocolo de Inicio de Sesión).

[7] *Real-time Transport Protocol.*

contenido de voz y vídeo que intercambian los intervinientes en una sesión establecida por SIP.

Las funciones básicas del protocolo incluyen:

- Determinar la ubicación de los usuarios, aportando movilidad.
- Establecer, modificar y terminar sesiones a varias bandas entre usuarios.

El protocolo SIP adopta el modelo cliente-servidor y es transaccional, esto es, el cliente realiza peticiones (*requests*) que el servidor atiende, generando una o más respuestas. Por ejemplo, para iniciar una sesión, el cliente realiza una petición en donde indica con qué usuario o recurso quiere establecer la sesión. El servidor responde, bien rechazando bien aceptando esa petición en una serie de respuestas. Las respuestas llevan un código de estado que brindan información acerca de si las peticiones fueron resueltas con éxito o si se produjo un error. La petición inicial y todas sus respuestas constituyen la mencionada transacción.

Los servidores utilizan TCP[8] o UDP[9] para recibir las peticiones de los clientes SIP. Aunque existen muchos otros protocolos de señalización para VoIP, SIP se caracteriza porque su origen reside en la comunidad de Internet y no en la industria de las telecomunicaciones. De ahí sus grandes diferencias con el resto de alternativas mencionadas. Así, SIP ha sido estandarizado y dirigido principalmente por el IETF, frente al protocolo VoIP H.323, que ha sido tradicionalmente más asociado a la UIT[10].

2.1.2.3. Asterisk.

Asterisk es una aplicación de código abierto para la creación de aplicaciones de comunicaciones desarrollada por *Digium*. Este *software* convierte un ordenador en un servidor de telecomunicaciones bastante flexible y potente, creando subsistemas PBX IP, puertas de enlace VoIP y servidores de

[8] *Transmission Control Protocol.*

[9] *User Datagram Protocol.*

[10] Unión Internacional de Telecomunicaciones.

conferencia. Hoy *Asterisk* es utilizado por pequeñas y grandes empresas, centros de llamadas (*call centers*) e incluso entornos institucionales públicos. *Asterisk* incluye todos los componentes básicos necesarios para crear una PBX, un sistema IVR[11], un puente de conferencias y cualesquier otros servicios necesarios en un entorno de telecomunicaciones.

2.1.3. Arquitectura de red.

El estándar define tres elementos fundamentales en su estructura: Los terminales (teléfonos), que se pueden implementar tanto en *software* como en *hardware*; los *gatekeepers*, centro de toda la organización VoIP, equivalentes a las tradicionales centralitas y los *gateways*, que enlazan la red VoIP con la red telefónica tradicional.

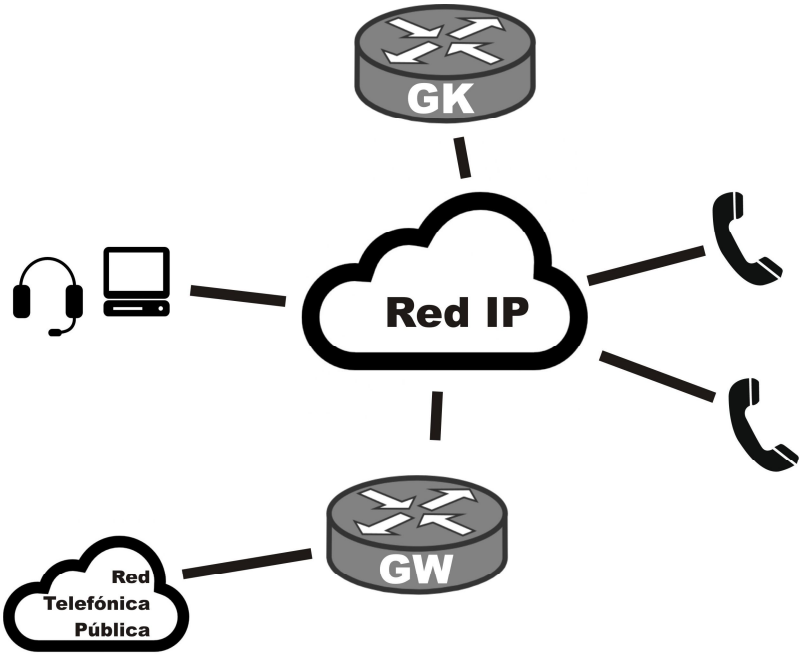


Figura 2.1. Arquitectura VoIP.

Con estos tres elementos, la estructura de la red VoIP podría ser la conexión de dos delegaciones de una misma empresa. La ventaja es inmediata: todas las comunicaciones entre las delegaciones son completamente gratuitas. Este

[11] *Interactive Voice Response* o respuesta de voz interactiva.

mismo esquema se podría aplicar para proveedores, con el consiguiente ahorro que esto conlleva.

2.1.4. Códecs.

La voz se codifica para poder ser transmitida por la red IP. Para ello se hace uso de códecs que garanticen la codificación y compresión del audio o del video para su posterior decodificación y descompresión antes de poder generar un sonido o imagen legibles. Según el códec utilizado en la transmisión, se utilizará un mayor o menor ancho de banda, que suele ser directamente proporcional a la calidad de los datos transmitidos. Entre los códecs más utilizados en VoIP están G.711, G.723.1 y el G.729 (especificados por la ITU-T). Estos códecs tienen los siguientes anchos de banda de codificación:

- G.711: bit-rate de 56 o 64 Kbps.
- G.722: bit-rate de 48, 56 o 64 Kbps.
- G.723: bit-rate de 5,3 o 6,4 Kbps.
- G.728: bit-rate de 16 Kbps.
- G.729: bit-rate de 8 o 13 Kbps.

Este dato no se corresponde con el ancho de banda real utilizado, ya que hay que sumar el tráfico. Por ejemplo el códec G.729 utiliza 31.5 Kbps de ancho de banda en su transmisión.

2.1.5. Parámetros críticos.

2.1.5.1. Retardo o latencia. Una vez establecidos los retardos de propagación y el retardo de procesado, la conversación se considera aceptable por debajo de los 150 ms. de retardo.

2.1.5.2. Pérdida de tramas. Durante su recorrido por la red IP, las tramas pueden perderse como resultado de una congestión de red o corrupción de datos. Además, para tráfico en tiempo real como la voz, la retransmisión de tramas perdidas en la capa de transporte es molesta por ocasionar retardos adicionales. Por consiguiente, los terminales de voz tienen que retransmitir con muestras de voz perdidas, también llamadas *Frame Erasures*. El efecto de las

tramas perdidas en la calidad de voz depende de cómo los terminales gestionen estas. En el caso más simple, si se pierde una muestra de voz, el terminal dejará un intervalo de tiempo en el flujo de voz. Si muchas tramas se pierden, la pérdida de información será de sílabas o incluso palabras perdidas. Una posible estrategia de recuperación es reproducir las muestras de voz previas. Esto funciona bien si sólo unas cuantas muestras son perdidas. Para combatir mejor las ráfagas de errores usualmente se emplean sistemas de interpolación. Basándose en muestras de voz previas, el decodificador predecirá las tramas perdidas. Esta técnica es conocida como *Packet Loss Concealment* (PLC). La ITU-T G.113 apéndice I provee algunas líneas de guía de planificación provisional en el efecto de pérdida de tramas sobre la calidad de voz. El impacto se mide en términos de *Ie*, el factor de deterioro. Este es un número en el cual 0 significa sin deterioro. A mayor valor de *Ie,* mayor es el grado de deterioro.

2.1.5.3. Calidad del servicio. Para mejorar el nivel de servicio, se ha tendido a disminuir los anchos de banda utilizados, para lo que se ha trabajado bajo diferentes iniciativas, como la supresión de silencios, que otorga más eficiencia a la hora de realizar una transmisión de voz, ya que se aprovecha mejor el ancho de banda al transmitir menos información. Otra opción ha sido la compresión de cabeceras aplicando los estándares RTP/RTCP.

Para la medición de la calidad de servicio (QoS), existen cuatro parámetros clave: ancho de banda (BW), retardo temporal (*delay*), variación de retardo (*jitter*) y pérdida de paquetes (*packet loss*).

Para solucionar este tipo de problemas, en una red se pueden incorporar tres tipos básicos de métodos o criterios QoS:

- *Best effort* (mejor esfuerzo). Este método envía paquetes a medida que los va recibiendo, sin aplicar ninguna tarea específica real. Es decir, no tiene ninguna prioridad para ningún servicio, ya que solo trata de enviar los paquetes de la mejor manera.
- Servicios Integrados. Este sistema predefine un camino para los datos que necesitan prioridad. Esta arquitectura no es escalable, debido a la cantidad de recursos que necesita para reservar los anchos de banda de

cada aplicación. RSVP[12] fue desarrollado como el mecanismo para programar y reservar el ancho de banda requerido para cada una de las aplicaciones que son transportados por la red.

- Servicios Diferenciados. Este sistema permite que cada dispositivo de la red pueda manejar paquetes individualmente, además cada *router* y *switch* puede configurar sus propias políticas de QoS, para tomar sus propias decisiones acerca de la entrega de los paquetes. Los servicios diferenciados utilizan 6 bits en la cabecera IP (DSCP[13]). Los servicios para cada DSCP son los siguientes:
  - Best Effort, que no ofrece garantías, como hemos visto.
  - *Assured Forwarding* (AF), que asegura un trato preferente. Si los valores de DSCP son más altos, tendrá mayor prioridad el tráfico y disminuye la posibilidad de ser eliminado por congestión.
  - *Expedited Forwarding* (EF), que se utiliza para dar el mejor servicio, brindando más garantías (utilizada para tráfico de voz o vídeo).

2.1.6. IPv6.

La implantación de IPv6 proporciona mayor espacio de direccionamiento y la posibilidad de *tunneling,* lo que también permite un uso eficiente del ancho de banda, así como la priorización de los paquetes que requieran menor latencia, donde destacan las siguientes técnicas:

2.1.6.1. *Priority Queueing* (PQ). Mecanismo de asignación de prioridad que se caracteriza por definir 4 colas con diferentes niveles de prioridad: alta, media, normal y baja. Además, determina cuáles son los paquetes que van a estar en cada una de dichas colas. Sin embargo, si estas colas no son configuradas, serán asignadas por defecto a la prioridad normal. Por otra parte, mientras existan paquetes en la cola de prioridad alta, no se atenderá ningún paquete con prioridad media hasta que la cola de prioridad alta se encuentre vacía, así para los demás tipos de cola.

[12] *Resource Reservation Protocol* o protocolo de reserve de recursos.

[13] Differentiated Services Code Point o punto de código para servicios diferenciados.

2.1.6.2. *Weighted fair queuing* (WFQ). Este método divide el tráfico en flujos, proporcionando una cantidad de ancho de banda justo a los flujos activos en la red. Los flujos con poco volumen de tráfico serán enviados más rápido. Es decir, WFQ prioriza aquellas aplicaciones de menor volumen, más sensibles al *delay* (retardo) como VoIP. Por otra parte, penaliza aquellas que no asocia como aplicaciones en tiempo real como FTP, donde el retardo no es un factor crítico.

2.1.6.3. *Custom Queueing* (CQ). Este mecanismo asigna un porcentaje de ancho de banda disponible para cada tipo de tráfico (voz, vídeo y/o datos), además especifica el número de paquetes por cola. Las colas son atendidas según *Round Robin* (RR). El método RR asigna el ancho de banda de forma dinámica a cada uno de los diferentes tipos de tráfico existentes en la red. Con este método no es posible priorizar tráfico ya que todas las colas son tratadas de la misma manera.

2.2. Teoría de colas.

La teoría de colas estudia, basándose en un soporte matemático importante, las colas o líneas de espera dentro de un sistema. Ésta teoría estudia factores tales como el tiempo de espera medio en las colas o la capacidad de trabajo del sistema sin que llegue a colapsarse, aspecto de gran valor en el análisis de la aplicación objeto de este proyecto. Dentro de las matemáticas, la teoría de colas se engloba en la investigación operativa y es un complemento muy importante a la teoría de sistemas y la teoría de control. Se trata de una teoría que encuentra aplicación en una amplia variedad de situaciones.

En el caso concreto de la ingeniería [Chee 2008], la teoría de colas permite modelar sistemas en los que varios agentes (materializados en nuestro caso en usuarios que llaman) demandan cierto servicio o prestación (llamada / respuesta) confluyen en un mismo servidor (centralita) y, por lo tanto, pueden registrarse esperas desde que un agente llega al sistema y el servidor atiende sus demandas. En este sentido, la teoría es muy útil para modelar procesos tales como la llegada de datos (en nuestro caso, en formato VoIP) a una cola en sistemas de telecomunicación. En este contexto, las situaciones de

espera (fijadas en el parámetro timbre del fichero de datos SMDR estudiado) dentro de una red son más frecuentes. Así, por ejemplo, los procesos enviados a un servidor para su ejecución forman colas de espera mientras no son atendidos; la información solicitada, a través de Internet, a un servidor *Web* puede recibirse con demora debido a la congestión en la red. Cualquier analogía entre equipos de Internet y los equipos objetos de estudio de este proyecto es válida.

2.2.1. Elementos de la teoría de colas.

2.2.1.1. Proceso básico de colas. Los clientes que requieren un servicio se generan en una fase de entrada. Estos clientes entran al sistema y se unen a una cola. En determinado momento se selecciona un miembro de la cola, para proporcionarle el servicio, mediante alguna regla conocida como disciplina de servicio. Luego, se lleva a cabo el servicio requerido por el cliente en un mecanismo de servicio, después de lo cual el cliente sale del sistema de colas.
2.2.1.2. Fuente de entrada o población potencial. Una característica de la fuente de entrada es su tamaño. El tamaño es el número total de clientes que pueden requerir servicio en determinado momento. Puede suponerse que el tamaño es infinito o finito. En nuestro caso, supondremos que es finito, limitado por la capacidad de la centralita.
2.2.1.3. Cliente. Es todo individuo de la población potencial que solicita servicio como por ejemplo una lista de trabajo esperando para imprimirse. En nuestro caso serían las llamadas.
2.2.1.4. Capacidad de la cola. Es el máximo número de clientes que pueden estar haciendo cola (antes de comenzar a ser atendidos). De nuevo, puede suponerse finita o infinita. En nuestro caso, la capacidad se supone finita.
2.2.1.5. Disciplina de la cola. La disciplina de la cola se refiere al orden en el que se seleccionan sus miembros para recibir el servicio. Por ejemplo, puede ser:

- FIFO (*first in first out*) primero en entrar, primero en salir, según la cual se atiende primero al cliente que antes haya llegado. Será nuestro caso.

- LIFO (*last in first out*) también conocida como pila que consiste en atender primero al cliente que ha llegado el último.
- RSS (*Random Selection of Service*) que selecciona los clientes de manera aleatoria, de acuerdo a algún procedimiento de prioridad o a algún otro orden. Este escenario es el de los *call centers*.
- *Processor Sharing*. Sirve a los clientes igualmente. La capacidad de la red se comparte entre los clientes y todos experimentan con eficacia el mismo retraso.

2.2.1.6. Mecanismo de servicio. Define cómo son atendidos los clientes. El mecanismo de servicio consiste en una o más instalaciones de servicio, cada una de ellas con uno o más canales paralelos de servicio, llamados servidores. En nuestro caso partimos del escenario más limitado, con un canal/línea único/a. Sin embargo puede ampliarse en función del número de operadores en la pestaña designada.

2.2.1.7. Redes de colas. Sistema donde existen varias colas y los trabajos fluyen de una a otra. Por ejemplo: las redes de comunicaciones o los sistemas operativos multitarea.

2.2.2. Notación Kendall.

David G. Kendall introdujo la notación de colas A/B/C en 1953 para describir las colas y sus características. Ha sido desde entonces extendida a 1/2/3/ (4/5/6) donde los números se reemplazan con:

Un código que describe el proceso de llegada. Los códigos usados son:

- M para Markoviano (la tasa de llegadas sigue una distribución de *Poisson*), significando una distribución exponencial para los tiempos entre llegadas.
- D para unos tiempos entre llegadas "determinísticas".
- G para una "distribución general" de los tiempos entre llegadas, o del régimen de llegadas.

Un código similar que representa el proceso de servicio (tiempo de servicio). Se usan los mismos símbolos.

2.2.2.1. Número de canales de servicio (o servidores). La capacidad del sistema, o el número máximo de clientes permitidos en el sistema. Cuando el número llega al valor máximo, las llegadas siguientes son rechazadas (que se materializan en nuestro caso en llamadas perdidas). Un caso particular de esta situación es el modelo M/M/n/n o Erlang-B, en el cual no hay cola de espera, sino n recursos (servidores) y hasta n usuarios como máximo; si llega el usuario n+1, es rechazado. Este último modelo es el que se aplica en telefonía convencional. Otro caso particular es el modelo Erlang-C o M/M/n, donde la capacidad del sistema es ilimitada, aunque haya sólo n recursos; en caso de llegar el recurso número n+1, pasará a una cola de espera, pero no es rechazado.

El orden de prioridad en la que los trabajos en la cola son servidos:

- *First Come First Served* (FCFS) o *First In First Out* (FIFO).
- *Last Come First Served* (LCFS) o *Last In First Out* (LIFO).
- *Service in Random Order* (SIRO).
- *Processor Sharing*.

El tamaño del origen de las llamadas. El tamaño de la población desde donde los clientes vienen, lo que limita la tasa de llegadas.

2.2.3. Aplicación a la telefonía.

Las redes telefónicas se han diseñado para acomodar el tráfico ofrecido a una mínima pérdida. El funcionamiento de los sistemas depende de si la llamada es rechazada, de si la llamada se ha perdido, etc. Normalmente, los sistemas de desbordamiento hacen uso de rutas alternativas e incluso estos sistemas tienen una capacidad de carga finita o máxima de tráfico. Sin embargo, el uso de las colas permite que los sistemas esperen por las peticiones del cliente hasta que los recursos libres estén disponibles. Esto significa que si los niveles de la intensidad del tráfico exceden de la capacidad disponible, las llamadas de los clientes se pierden. La disciplina de colas determina la manera de cómo manejar las llamadas de los clientes. Define la manera en que se gestionan, el orden en que se sirven, y la manera en la que los recursos se dividen entre los clientes.

En nuestro caso, suponemos que no se han desarrollado estudios de análisis de colas al tráfico SMDR analizado, sino que es precisamente a partir de los datos ofrecidos por la aplicación donde deben iniciarse los estudios y análisis de necesidades en los equipos existentes.

2.3 Aplicaciones en call-center.

En este proyecto se trabaja sobre una serie de supuestos que enmarcan el desarrollo de la herramienta *software*, así como del modelo de instalación *hardware* de trabajo. La herramienta de *software* aquí propuesta es un recurso de ayuda en la toma de decisiones sobre instalaciones telefónicas orientadas al tráfico entrante, por lo que se puede hacer una equiparación con los Centros de Llamadas, más conocidos como *call centers*.

Los *call centers* se fundamentan en buena parte en la teoría de colas, una perspectiva natural y útil para el análisis y diseño de estos sistemas telefónicos. Sin embargo, los *call centers* actuales son sistemas cada vez más complejos, con novedosas características como IVR[14], enrutamiento basado en habilidades, chat, correo electrónico, redes sociales y otros factores que hacen especialmente complejo el análisis, dejando incluso la teoría de colas como una herramienta insuficiente. Aparece entonces la simulación como única vía para el manejo de estos casos.

Los grandes avances en las telecomunicaciones han hecho que el número, tamaño y alcance de los *call centers* haya crecido en gran forma durante la última década. Tan solo en EEUU, la industria de los *call centers* emplea a más personas que la agricultura. En Europa, la situación es análoga. Una disciplina denominada Ingeniería de Servicio busca dar soporte al diseño y gestión de las operaciones involucradas en el campo de acción a los *call centers*.

2.3.1. Modelo general.

Un *call center* consiste en un sistema tecnológico con gran nivel de interacción humana. La figura 2.3.1 muestra un modelo general de lo que puede ser un *call*

[14] Interactive Voice Response.

center con un solo tipo de llamada entrante. En el mismo se puede ver un sistema con n agentes atendiendo k llamadas y n + k líneas telefónicas (con K >= 0).

En el caso en el que un cliente llama (llamada entrante) tenemos tres escenarios posibles:

1. Ser atendido inmediatamente si hay algún agente libre.
2. Esperar en cola, si todos los agentes están ocupados y hay líneas libres.
3. Ser rechazado, por no haber líneas libres (llamada perdida).

Para el caso del cliente que queda en cola se dan dos posibilidades:

1. Esperar hasta ser atendido, cuando algún agente esté libre.
2. Dejar el sistema sin ser atendido.

Adicionalmente existen los reintentos y las rellamadas, que corresponden a clientes que ya fueron atendidos pero que, por alguna razón, vuelven a ponerse en contacto con el *call center*.

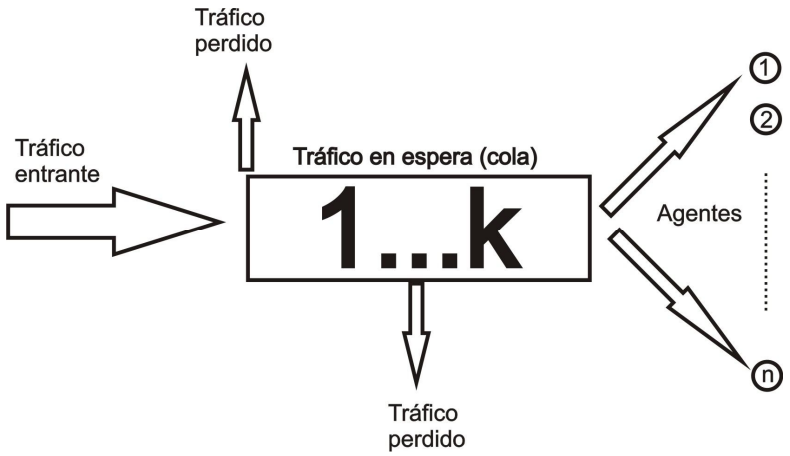


Fig. 2.3.1. Modelo general de *call center* con un solo tipo de llamada entrante

2.3.2. Indicadores de calidad.

A continuación se relaciona una lista de los principales indicadores utilizados para analizar la calidad de un *call center*. Se considera el parámetro Timbre como el tiempo de espera de los clientes para ser atendidos.

2.3.2.1. Grado de servicio[15].

Es el principal indicador de calidad desde el punto de vista de la atención al cliente. Se corresponde con el porcentaje de llamadas atendidas antes de un cierto tiempo, parámetro de valor fijo definido en función de la calidad de servicio que se quiere dar[16].

$$SL = \frac{n_{AWT}}{n_a}$$

Donde n_{AWT} es el número de llamadas atendidas antes de pasado un tiempo AWT y n_a es el número de llamadas atendidas total.

2.3.2.2. Espera media (Timbre medio)[17].

Es el tiempo medio que espera un cliente en ser atendido. En el analizador SMDR se distingue también entre el timbre medio de las llamadas perdidas y el de las atendidas.

2.3.2.3. Porcentaje de abandonos (o de llamadas perdidas).

Es la fracción de llamadas (del total de entrantes), en que los clientes abandonan por las razones que son de suponer (agotamiento de la paciencia, principalmente o limitaciones del sistema).

$$P(A_b) = \frac{n_p}{n_e}$$

Donde $P(A_b)$ es la probabilidad de abandono/pérdida, n_p es el número de llamadas perdidas y n_e el número total de llamadas entrantes.

[15] Service Level (SL), también conocido como Grade of Service (GoS).

[16] Tiempo medio de espera aceptable (AWT).

[17] Average Speed of Answer (ASA).

Una variante de este indicador es contabilizar solamente las llamadas perdidas que esperan un tiempo mínimo a establecer. De esta forma se elimina a los clientes ansiosos.

$$P(\frac{A_b}{Timbre} > t_{\min}) = \frac{n_p}{n_e}$$

Donde n_p es el número de llamadas perdidas que esperan más de t_{min} y n_e es el número total de llamadas entrantes.

2.3.2.4. Porcentaje de bloqueos.

Es el ratio de llamadas entrantes que son bloqueadas por no tener líneas disponibles.

$$P(Blk) = \frac{n_b}{n_e}$$

Donde P(Blk) es la probabilidad de bloqueos, n_b es el número de bloqueos y n_e es el número total de llamadas entrantes.

2.3.2.5. Porcentaje de ocupación de los agentes.

Es el ratio de tiempo del total que opera el *call center*, en el que los agentes están atendiendo llamadas.

$$P(A_o) = \frac{\sum m}{T_t}$$

Donde $P(A_o)$ es el porcentaje de ocupación de las y los agentes, m es el número de agentes atendiendo llamadas y T_t el tiempo total de trabajo del *call center.*

2.3.3. Diseño.

En base a los indicadores de calidad definidos, en el proceso de diseño del *call center* se establecen objetivos y parámetros críticos de calidad. Así, a partir de un objetivo de diseño determinado, en el cual se define el grado de servicio, la probabilidad de bloqueo y otros, lo que se desea es hallar los recursos necesarios para cumplir con los objetivos establecidos. Esto genera nuevas variables a definir:

2.3.3.1. Predicción de llamadas entrantes: Se debe modelar el proceso de llegada para poder estimar la carga a gestionar. En este proyecto no se considera esta predicción debido a que se analizan ficheros históricos SMDR, que es su objetivo principal.

2.3.3.2. Estudio del comportamiento humano: Es necesario definir los tiempos de servicio y la paciencia de los clientes, eventos asociados al comportamiento de las personas. En este proyecto no se considera este estudio debido a que quedaría fuera del ámbito de aplicación del mismo

2.3.3.3. Modelos estacionarios: Se deben considerar intervalos donde se cumpla esta propiedad, por lo que es necesario estudiar el funcionamiento del sistema, fraccionando el día en períodos donde se cumpla la hipótesis.

2.3.3.4. Esquema de horarios: Después de ubicar los agentes para cada intervalo horario, se debe crear un esquema de horarios compatible para tener la cantidad necesaria de agentes en cada tramo, con jornadas de trabajo adecuadas. Adicionalmente pueden establecerse ésquemas que consideren las habilidades de los agentes.

2.3.4. Modelos basados en teoría de colas.

Como se ha indicado anteriormente, el estudio de los *call centers* no complejos está basado principalmente en la teoría de colas, esencial para el modelado del sistema. Los parámetros del sistema se consideran constantes durante cierto período, para trabajar con un modelo estacionario. A partir del modelo, se obtiene la distribución estacionaria del sistema. Con ella es posible hallar de forma analítica los indicadores de calidad del *call center*.

2.3.4.1. Erlang-C.

Este modelo detallado en el epígrafe 2.2 es el más habitual para el dimensionado de *call centers*, es el denominado M/M/N, siendo N la cantidad de agentes e infinito el número de líneas. Este modelo no toma en cuenta los abandonos, bloqueos o reintentos. En la figura 2.3.2 se puede ver el modelo Erlang-C.

Fig. 2.3.2: Modelo Erlang-C.

2.3.4.2. Erlang-A: Considerando abandonos.

Erlang-A se denomina M/M/N+M, y añade la variable de paciencia de los clientes mediante un valor aleatorio exponencial. Así, si las llamadas perdidas se producen a una tasa θ, el tiempo medio de espera en ser atendido es 1/θ.
Es evidente que en cualquier supuesto se debe considerar el tráfico perdido sea cual sea el modelo utilizado.

2.3.4.3. Erlang-C vs Erlang-A.

Si realizamos una comparación entre Erlang-c y Erlang-A, descubrimos que Erlang-C es inestable para un número de operadores determinado, pero por otro lado, si utilizamos Erlang-A podemos obtener un sistema estable para un mayor número de operadores. Esto implica que para la misma tasa de llamadas entrantes, el tamaño medio de la cola y la espera promedio es bastante menor para el caso de Erlang-A, con un nivel de ocupación de los agentes un tanto menor y un porcentaje de pérdidas reducido.

Este aparentemente pequeño porcentaje de abandonos da como resultado una calidad de servicio muy superior para los clientes que permanecen en el sistema. La razón es que los abandonos reducen la carga justo cuando se necesita, es decir cuando la congestión del sistema es alta. De la misma forma llegamos a la conclusión de que en el caso de Erlang-C, a menor tasa de llegada de llamadas se reduce en igual medida el porcentaje de llamadas perdidas. En cualquier caso obtenemos una calidad del servicio menor que para el caso de Erlang-A.
Evidentemente se puede mejorar la calidad del sistema con un aumento de la cantidad de agentes, pero este parámetro busca un mínimo. Debe reseñarse que la aplicación de análisis SMDR desarrollada no se ha utilizado el modelo Erlang-A debido a que el formato SMDR no aporta información sobre abandonos.

2.3.5. Regímenes de operación.

Una de las metas en el diseño y gestión de un *call center*, es lograr un equilibrio entre la calidad de servicio que se ofrece y la eficiencia con que se utilizan los recursos. Considerando una elección adecuada del número de agentes, esto se debe traducir en un dimensionamiento adecuado para no afectar a la calidad del servicio.
Según la decisión por la que se opte, un *call center* puede tener un diseño orientado a optimizar el sistema desde el punto de vista del cliente o desde el punto de vista de la eficiencia en la ocupación de los agentes.
Así, deben considerarse distintos regímenes que simplifican el cálculo de los indicadores de calidad. Estos regímenes se presentan a continuación en el ámbito del modelo Erlang-A. En todos los casos N es el número de agentes y $\rho = \lambda/\mu$.

2.3.5.1. ED - *Efficiency Driven*.

Se busca trabajar con un 100% de ocupación de los agentes. En este caso todos los clientes esperan. Este régimen es útil para aplicaciones sin fines de lucro (donaciones, acciones humanitarias) donde se quiere sacar el máximo

provecho a los recursos existentes, siempre escasos. En este caso la regla de diseño es:

$$N = \rho \times (1 - \gamma) + o \times \sqrt{\rho}$$

Donde $\gamma > 0$, el grado de abandonos en este caso tiende a γ y el tiempo medio de espera es aproximadamente λ/θ.

2.3.5.2. QD - *Quality Driven*.

QD está orientado a la calidad de servicio. Se utiliza como criterio para aplicaciones donde la calidad es mucho más importante que la eficiencia, como puede ser el caso de teléfonos de emergencia, caso bastante orientado al objetivo de este proyecto. La regla de diseño será:

$$N = \rho \times (1 + \gamma) + o \times \sqrt{\rho}$$

Donde $\gamma > 0$. De esta forma el tiempo de espera medio, el porcentaje de abandonos/pérdidas y la probabilidad de esperar tienden a cero de forma exponencial con N.

2.3.5.3. QED - *Quality & Efficiency Driven*.

Para este caso se busca un compromiso entre la calidad de servicio y la eficiencia, esto es, dar buena calidad de servicio pero con un alto grado de ocupación de los agentes. Es el régimen de operación habitual en centros de ventas y servicios de atención al cliente. A saber:

$$N = \rho + \beta \times \sqrt{\rho} + o\sqrt{\rho}$$

Siendo $-\infty < \beta < \infty$. Debe entenderse β como el parámetro que define la calidad de servicio. Para porcentajes de abandono moderados se cumple que β está entre -1 y 2.

2.3.6. Enrutamiento basado en habilidades[18].

En los modelos presentados anteriormente Erlang-C y Erlang-A, solo se considera un tipo de llamada entrante. A continuación se presenta un caso más complejo, que recibe varios tipos de llamadas. En este caso se establecen varios grupos de agentes, cada uno de los cuales con distintas *habilidades*, definidas en función de las llamadas que puede atender cada uno.
La tecnología de enrutamiento basado en habilidades permite diferenciar distintos tipos de llamadas (y por tanto de perfiles de clientes) y varios grupos de agentes. Si bien la selección de los tipos de llamadas/clientes es una tarea más propia del área de marketing (lo cual deja este supuesto fuera del ámbito de aplicación del presente proyecto), la definición de los grupos de agentes es una tarea del área de recursos humanos. La correcta organización de los grupos en función de las necesidades constituye un importante desafío de diseño, ya que exige tomar la decisión en tiempo real de elegir la llamada a atender así como seleccionar al agente a asignar y qué llamada atiende de la cola un agente que se libera.

2.3.6.1. Matriz agentes-habilidades.

Una de las decisiones clave en este contexto es definir qué tipo de llamadas sabe/debe atender cada grupo de agentes. Una opción para esto es utilizar la denominada matriz agentes-habilidades, en la que se definen tanto las habilidades de cada agente como la prioridad para atender cada llamada. En esta matriz M, de tamaño N × C (N es el número de agentes y C los tipos de llamadas), las filas se corresponden con los agentes mientras que las columnas se corresponden con las prioridades. Si la entrada $A_{i,j} = h$ significa que el agente *i* tiene la habilidad *h* con nivel de prioridad K; Si $A_{i,j} = 0$, entonces el agente *i* no tiene la habilidad *h* con nivel de prioridad *j*; Si no existe *j* tal que $A_{i,j} = h$, entonces no puede atender ese tipo de llamadas.

[18] *Skill-based routing* (SBR).

Así, la fila *i* define las habilidades y sus prioridades para el *i−ésimo* agente. Se considera en este caso que un agente tiene como máximo una habilidad para una prioridad definida (aunque no tiene por qué ser así) y que para cada agente, una habilidad determinada puede estar como máximo en un nivel de prioridad determinado. La primera columna de *M* tiene las prioridades más altas para cada agente. Por lo tanto *Ai,1* es la habilidad primaria para el agente *i*. Se parte de la base de que cada agente tiene una habilidad primaria.

Después de definir los grupos de agentes y sus habilidades, es necesario decidir una política de enrutamiento, esto es, se deben especificar las decisiones a adoptar en dos casos:

1. Cuando llega una llamada.
2. Cuando un agente queda libre.

1. Cuando se tiene una nueva llamada entrante del tipo *h*, la misma se encauza al grupo de agentes que tienen esta habilidad con nivel de prioridad 1. Para asignarla a un agente en particular existen varias opciones, entre las que destaca la denominada LIAR[19], por ser la más justa en el reparto de carga de trabajo para los agentes. Esta política asigna la llamada al agente que hace más tiempo que se encuentra disponible. Si todos los agentes que tienen la habilidad *h* como primaria se encuentran ocupados, se repite el proceso con los que la tienen como secundaria y así sucesivamente hasta encontrar un agente libre o en su defecto, si no hay agente libre que pueda atender dicha llamada, encolar la misma.

2. Para el caso en que un agente queda libre, si no hay llamadas en la cola, el mismo queda libre. Si no es así, el agente busca en la cola alguna llamada de las habilidades que posee (la acción humana es evidente en este proceso, lo que hace inoperativo todo cálculo matemático, excepto si se plantea un sistema experto). La búsqueda se realiza según el orden de prioridad de las habilidades que tiene, empezando por la habilidad primaria, luego la secundaria y así consecutivamente hasta encontrar una llamada para atender. De no haber una llamada que pueda atender, también queda libre. Esto mantiene un cierto

[19] Longest-Idle-Agent-Routing.

paralelismo con un recorrido de las colas de cada habilidad *h*, según el orden de prioridad del agente, donde para cada cola el servicio es del tipo *FIFO*.

2.3.6.2. Matriz de prioridades.

Para una asignación de agentes adaptada a cada tipo de llamada, puede (y debe) definirse una matriz de prioridades. Esta matriz debe ser de igual tamaño que la de agentes/habilidades. En este caso las filas se corresponden también con los agentes pero las columnas se corresponden con los tipos de llamadas. De esta forma, la entrada de la matriz *Ri,k* indica el nivel de prioridad de la llamada de tipo *k* para el agente *i*. Si el nivel de prioridad toma valores de 1 a infinito hay que elegir un valor especial (por ejemplo cero) para el caso en que el agente no atienda ese tipo de llamadas.
Esta estructura permite la convivencia de distintos tipos de llamadas con igual nivel de prioridad para un determinado agente. Por otro lado el enrutamiento es más complejo, debido a que si los niveles de prioridad son iguales, es necesario un segundo algoritmo para distinguir esta equidad. Después de identificar la llamada a atender, la elección del agente es análoga a lo descrito en el epígrafe anterior.

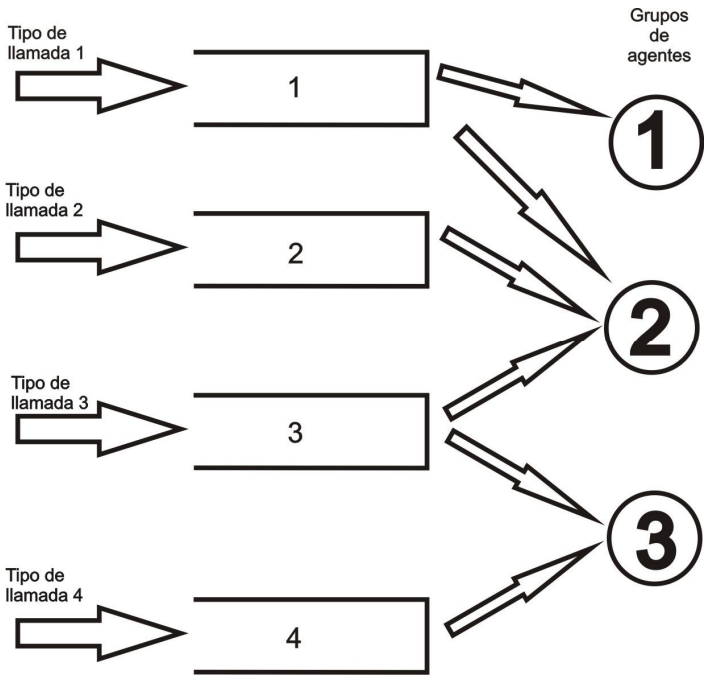


Fig. 2.3.3. Modelo Erlang-A.

2.3.6.3. Modelado del sistema.

Los sistemas complejos se diferencian de los simples en que estos últimos solo tienen un tipo de llamadas, por lo que el estudio analítico de aquellos se vuelve bastante complicado. En la figura 2.3.3 puede verse el modelo particular considerando colas según el modelo Erlang-A. La complejidad de los grandes sistemas actuales excede los límites de la teoría, lo que obliga a analizar estos sistemas de dos formas:

- Utilizar la teoría de colas con modelos simplificados.
- Usar herramientas de simulación para análisis de sistemas complejos.

2.3.6.4. Bloques canónicos.

Los bloques canónicos se deben entender como piezas básicas para la construcción de un *call center* con SBR. En la figura 2.3.4 se representan dichos bloques. Aún para diseños simples, el análisis debe realizarse mediante simulación en la mayoría de los casos debido a que el estado del arte del estudio analítico se demuestra insuficiente.

2.3.6.5. Routers.

En este contexto, un *router*[20] es el sistemna encargado de *escuchar* las llamadas y asignarlas bien a un grupo de agentes bien a una cola de espera. El *router* además *escucha* los fines de servicio, para saber cuando un agente queda libre, y asignarle un nuevo contacto. Este *software* define la política de enrutamiento con la que se trabaja, pero deben configurarse las diferentes políticas predefinidas.

2.3.7. Simulación.

La otra fórmula mencionada para el análisis de sistemas complejos corresponde al uso de herramientas de simulación. Estas herramientas ofrecen

[20] Distribuidor automático de llamadas (ACD).

la ventaja de evitar los límites del estudio analítico, permitiendo abordar sistemas de gran complejidad [Novasim 2015].

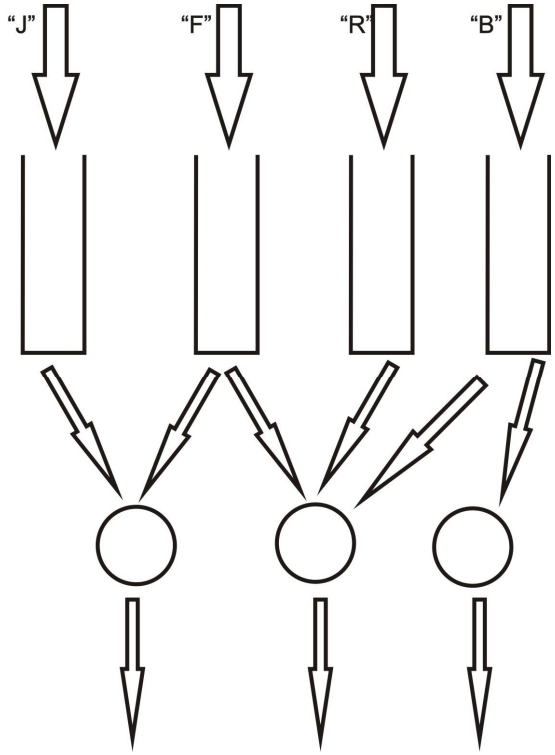


Fig. 2.3.4. Bloques canónicos de un *call center* con enrutamiento basado en habilidades.

A día de hoy existen algunas herramientas de simulación para el análisis de *call centers* que son de difícil acceso y a precios que solo grandes corporaciones pueden permitirse.

2.3.8. Otras consideraciones.

Habitualmente los modelos de *call center* establecen períodos de 15 a 60 minutos, por lo que para realizar una simulación se deben definir P períodos en los que opera el *call center,* siendo necesario agregar dos períodos adicionales: uno al principio de la jornada y otro al final[21]. Esto se hace para que la simulación comience antes de que abra el *call center* y termine después de que cierre. También se deben definir colectores estadísticos, que se encargan de

[21] Esta consideración debe ser diferente en los casos de que el call center atienda, por ejemplo, llamadas desde Europa y América estando ubicado en Latinoamérica, caso habitual.

recabar la información necesaria para calcular los indicadores de calidad deseados.

2.3.9. Caso práctico. Call center bilingüe.New

Un interesante caso particular que nos permite entender el modelo expuesto es el del caso de un *call center* bilingüe. En él se atienden dos tipos de llamadas, que pueden ser en inglés y en español. Para este caso tendremos agentes unilingües (que atenderán llamadas en un solo idioma) y agentes bilingües (que atenderán ambas). Este tipo de sistemas han sido estudiados de manera analítica utilizando los denominados cuasi-procesos de nacimiento y muerte.

2.3.9.1. Modelo.

Como se observa en la figura 2.3.5, el modelo que se propone tiene llamadas entrantes de tasas iguales para cada idioma. Definimos también los tiempos de servicio de los grupos de agentes con tasas iguales de igual valor tanto para los unilingües como para los bilingües. El valor de *paciencia* es de igual valor para ambos casos.

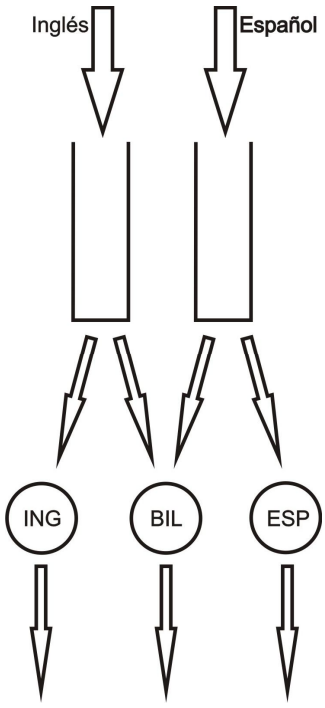


Fig. 2.3.5. Modelo de *call center* bilingüe.

Si se considera el orden de izquierda a derecha de la figura 2.3.5 para numerar los grupos de agentes, tenemos que los que atienden solo en inglés es el 1, bilingües el 2 y solo español el 3, quedando la matriz agentes-habilidades como sigue:

$$A = \begin{pmatrix} ING \\ ESP / ING \\ ESP \end{pmatrix}$$

Las filas se corresponden con los grupos de agentes y las columnas con las prioridades, por lo que hay una sola columna en este ejemplo simplificado. También debe considerarse que la matriz no cumple las hipótesis planteadas en su definición, ya que los agentes bilingües tienen igual prioridad para ambos tipos de llamadas, lo que impide utilizar esta matriz para definir la asignación de llamadas. Sin embargo, utilizando la matriz de prioridades sí es posible hacerlo, quedando ésta de la siguiente forma:

$$R = \begin{pmatrix} 1\,0 \\ 1\,1 \\ 0\,1 \end{pmatrix}$$

Donde las filas se corresponden con los grupos de agentes y las columnas con los tipos de llamadas. La prioridad 1 indica que atienden dichas llamadas y 0 que no lo hacen.

2.3.9.2. Observaciones al modelo.

Queda de la mano del empresario el fijar como objetivo minimizar la cantidad total de agentes a contratar, lo que se logra claramente considerando que todos los agentes sean bilingües. Pero ya que los mismos constituyen el personal más capacitado, son más difíciles de encontrar. Por ello se busca cumplir con los objetivos minimizando el número de agentes bilingües. En una situación más real, es probable que los agentes bilingües cobren más que los unilingües, por lo que debe ser tenido en cuenta para minimizar el costo total en agentes.

2.3.10. Comentarios finales.

Para concluir y a modo de resumen:

- Los *call centers* modernos son sistemas complejos, cuyo diseño y gestión implican un gran desafío, pero también un área de investigación muy interesante.
- El modelo Erlang-A, que considera abandonos, es una ampliación del tradicional modelo de Erlang-C, utilizado para el estudio de *call centers*. Dada la importancia que tienen las llamadas perdidas en la definición de la calidad de los *call centers*, debe considerarse trabajar con el modelo A para un mejor dimensionamiento del sistema y su análisis siempre que se pueda disponer de datos relativos a los abandonos.
- Para el estudio de *call centers* complejos es necesario utilizar herramientas de simulación para analizarlos.
- Los *call centers* con enrutamiento basado en habilidades permiten un mejor aprovechamiento de los recursos.

Capítulo 3

Equipos VoIP

Este capítulo se centra en el conjunto de equipos (centralita y teléfonos) que deben formar parte de un entorno laboral del que se va a evaluar su tráfico entrante. Los equipos reseñados en esta memoria forman parte del equipamiento del Laboratorio de Transmisión por Línea de la EITE-ULPGC.

3.1. Introducción.

En este epígrafe vamos a definir los rudimentos *hardware* necesarios para entender el escenario técnico de la telefonía IP.

3.1.1. Central telefónica IP.

Una central telefónica IP, como es el caso estudiado, es un equipo diseñado para ofrecer servicios de comunicación a través de redes de datos. A esta aplicación particular se le conoce como voz sobre IP[22]. La dirección IP del dispositivo lo identifica en la red. Con los componentes adecuados, la centralita permite administrar vídeo, troncales digitales o servicios de VoIP (*SIP trunking*) para llamadas a bajo costo. Los aparatos telefónicos necesarios para un uso eficiente de esta tecnología son los denominados teléfonos IP o SIP. Adicionalmente, estas centralitas permiten la convivencia de tecnologías, permitiendo líneas normales de las redes telefónicas públicas, y anexos analógicos para teléfonos estándar (fax, teléfonos inalámbricos, contestadores automáticos, etc.).

3.1.2. Teléfono IP.

Hoy en día, empresas y entornos laborales colaborativos de todo el mundo han optado por la tecnología IP en sus comunicaciones de voz, amén de las de datos. La creciente existencia de sistemas operativos y *software* gratuitos han hecho proliferar muchas centrales IP, algunas de las cuales utilizan un ordenador como *hardware* de centralita. Pero la seguridad es un asunto clave en estos equipos, ya que al estar conectados al Internet pueden estar

[22] *Internet Protocol* (Protocolo de Internet).

sometidos ataques. Más adelante se podrá observar que Avaya dispone de estándares de seguridad extremadamente eficientes.

La tecnología VoIP tiene un alto nivel de desarrollo y permite crear con relativa facilidad una amplia gama de aplicaciones de telefonía y servicios, incluyendo los de una centralita PBX[23] con diversas pasarelas (*gateways*) de VoIP.

Una de las ventajas clave en la incorporación de telefonía IP en un entorno de trabajo es que no hay necesidad de instalar cableado telefónico más allá del cableado estructurado existente en una red local. Los teléfonos IP o SIP[24] utilizan la red de datos existente, son fáciles de instalar y se manejan de forma intuitiva en la mayoría de los casos.

Esta facilidad permite que los trabajadores de una oficina puedan cambiar de un despacho a otro sin hacer cambios en el cableado o en la configuración de la centralita. La existencia de competencia permite elegir entre teléfonos SIP de varios fabricantes disponibles en el mercado sin necesidad de mantener equipos de una determinada marca. VoIP tiene un alto nivel de estandarización, lo que ha permitido su crecimiento. VoIP permite recibir y realizar llamadas a través de la red de telefonía móvil, utilizando las anteriormente mencionada pasarelas (*gateways*) de VoIP.

Para finalizar, cabe destacar que las centrales IP tienen el correo de voz incorporado, lo que habilita operadoras automáticas con mensajes de bienvenida y diferentes menús de opciones, que desvían llamadas automáticamente a diferentes destinos, etc.

[23] *Private Branch Exchange* o ramal privado de conmutación automática hace referencia a una centralita secundaria privada automática, lo que viene a ser cualquier central telefónica conectada directamente a la red pública de telefonía por medio de líneas troncales.

[24] Protocolo de inicio de sesión.

3.2. Centralita Avaya System Server IP Office 500.

El sistema IP Office 500, ofrece un uso eficiente de canales VCM[25] siendo un dispositivo adecuado para aplicaciones de telefonía IP. Este sistema se gestiona gracias al software IP Office Standard Edition. Además, IP *Office* 500 se diferencia de IP406V2 en que proporciona una mayor capacidad de expansión de enlaces de hasta cuatro interfaces PRI[26] (máximo de 96/120 enlaces). El equipo *IP Office 500* de que dispone el laboratorio de Transmisión por Línea incluye:

- 4 ranuras para alojar una variedad de tarjetas de expansión y VCM.
- Tarjeta *Digital Station* 8.
- Tarjetas Phone 2 y Phone 8. Tarjetas VCM-32 y VCM-64.
- Compatibilidad opcional de tarjeta secundaria de enlace.
- Tarjeta *Analog Trunk Module* 4.
- Tarjetas BRI-4 y BRI-8 (2 canales 2B+D y 4 canales 2B+D respectivamente).
- Puerto DTE de 9 pines para mantenimiento.
- Módulos de teléfono (8, 16, 30).
- Módulos *Digital Station* (16, 30).
- Módulo *Analog Trunk* 16.
- Módulo So8.
- Socket O/P externo compatible con dos puertos de conmutación de activación/desactivación de relé, por ejemplo, para sistemas de puertas de entrada.
- Puerto de entrada de audio para fuente externa de música en espera.
- 48 canales de datos.

[25] Módulo de Compresión de Voz.

[26] PRI es una configuración RDSI que funciona como un circuito troncal.

- Hasta 30 puertos *Voicemail Pro*[27].
- Dos puertos *Ethernet* conmutados 10/100 (Capa 3).

Figura 3.1. Centralita IP Office 500.

IP *Office* incluye una utilidad de registro de llamadas, IP *Office* SMDR, que se utiliza como fuente de obtención de datos. Esta utilidad permite enviar los detalles de todas las llamadas a un archivo en un formato legible por un ordenador, en particular en formato CSV[28]. De hecho, Avaya permite con esta herramienta que aplicaciones de terceros puedan utilizar estos datos para asignar costes a departamentos, analizar la capacidad de los enlaces, generar informes de uso para determinados códigos de cuentas y, sobre todo, diversos análisis sobre la calidad del servicio. Sin embargo, la utilidad IP *Office* SMDR no proporciona informes ni análisis gráficos del uso telefónico, asunto que implementamos en este PFC, permitiendo evaluar el grado de servicio. Para configuraciones de IP *Office* en múltiples emplazamientos, se requiere una aplicación IP *Office* SMDR en cada uno. Hay que destacar también que existen diversas herramientas en el mercado de análisis de ficheros SMDR, pero están orientados el control de gasto, esto es, al tráfico saliente. Este PFC se centra

[27] *Voice Mail Pro* ofrece prestaciones de Correo Vocal avanzado y escalable que permite disponer de un sistema de buzón de voz centralizado capaz de atender simultáneamente hasta 40 llamadas. *Voice Mail Pro* alerta e informa sobre nuevos mensajes, por ejemplo enviando por email mensajes de contestador.

[28] Valores Separados por Comas.

en el tráfico entrante, lo que permite tomar decisiones en base a momentos críticos en servicios de emergencias, por ejemplo.

3.3. *IP Office Server Edition*.

IP Office Server Edition [Avaya IP Office 8.1] proporciona a IP *Office* capacidades de telefonía, comunicaciones unificadas, movilidad y cooperación, alta disponibilidad, facilidad de uso y bajo coste. El sistema está centrado en compañías de tamaño medio, permitiendo hasta 32 ubicaciones, 1000 usuarios y una gran robustez integral.

La solución *IP Office Server Edition* se basa en una red en topología de estrella fija que proporciona y administra funciones globales. Proporciona una arquitectura modular y flexible que puede aumentar el número de usuarios y sitios al conectar en red varios servidores. Los componentes actúan como una única unidad lógica mediante un sistema de administración integrado.

3.3.1. Funciones.

IP *Office Server Edition* proporciona las siguientes funciones:

- Un solo servidor Primario de *Server Edition*.
- *Voicemail Pro* y Avaya *one-X® Portal*.
- De manera opcional, puede añadir un segundo servidor de *Server Edition* para aumentar la capacidad y garantizar mayor robustez.
- La solución IP *Office Server Edition* es compatible con hasta 30 sistemas de expansión que pueden proporcionar capacidades adicionales, compatibles con interfaces analógicas o digitales y sitios remotos.
- El Sistema de expansión *Server Edition* puede ser un servidor IP500 V2 o IP *Office Server Edition* ya preexistente.
- Los usuarios en la solución IP *Office Server Edition* se pueden configurar en el mismo servidor o en el Sistema de expansión *Server Edition*.
- La distribución de *software* de IP *Office Server Edition* incluye entre otros los siguientes componentes:
    - *Server Edition Manager*,

- *System Status Application* (SSA),
- Cliente *Voicemail Pro* y
- *SoftConsole*.

- Se puede configurar un secundario de *Server Edition* o Sistema de expansión *Server Edition* de manera central o desde un lugar remoto.
- De manera opcional, se puede configurar un servidor de aplicación separado dedicado a Avaya *one-X® Portal* para proporcionar más capacidad, superando las limitaciones del primario de *Server Edition*.
- Se pueden añadir nuevos servidores y sistemas de expansión en cualquier momento.

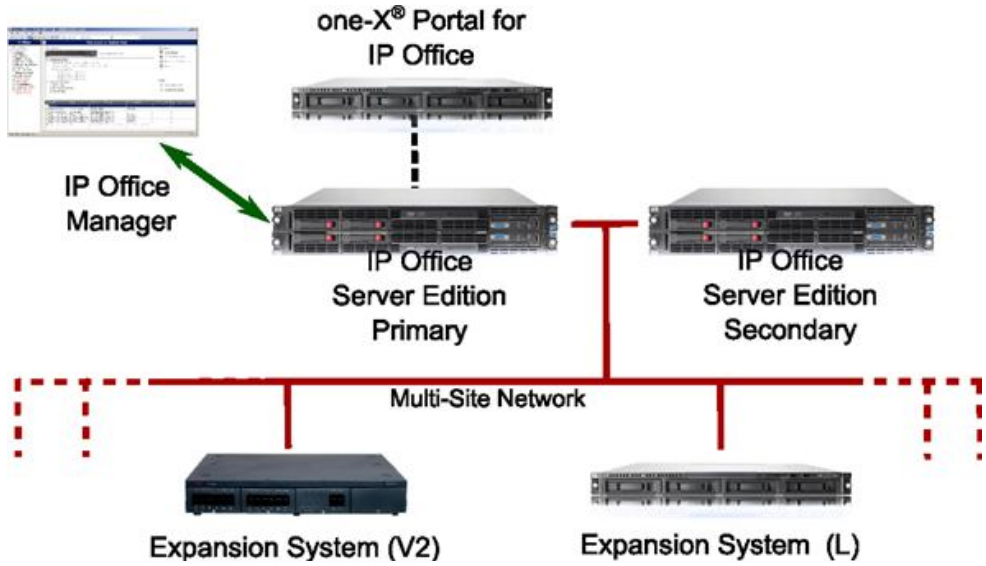


Figura 3.2. Arquitectura IP *Office*.

IP *Office Server Edition* ofrece funciones de continuidad para mantener un alto nivel de servicio en caso de cualquier falla de red o dispositivo a fin de garantizar un funcionamiento sin interrupciones. Una combinación de las funciones de acceso remoto, resistencia y redundancia garantizan la continuidad del trabajo.

3.3.2. Administración del sistema ante fallos.

En caso de fallo en alguno de los elementos del sistema, se puede seguir administrando y gestionando el servidor IP *Office Server Edition* y los

dispositivos en una red IP *Office Server Edition* instalados en el servidor Secundario de *Server Edition*. Esto proporciona una administración sin funciones fuera de línea y facilita el realineamiento de la configuración después de resolver los problemas en ella. La función de resincronización destaca el cambio en la configuración de fuente y hora además de permitir al administrador decidir qué conjunto de cambios se desea mantener. Además, se puede administrar directamente cada dispositivo y aplicación para permitir una configuración determinada mientras estos están aislados. Incluso se puede utilizar la función de resincronización para realinear las configuraciones después de conectar los dispositivos.

3.3.3. Copia de seguridad.

La solución IP *Office Server Edition* es compatible con multitud de funciones de copias de seguridad y restauración que permiten preservar y, eventualmente, recuperar configuraciones y datos en caso de una falla en los equipos. El primario de *Server Edition* ofrece una solución para generar copias de seguridad de manera manual y segura en un servidor externo que, opcionalmente, puede ser el mismo primario de *Server Edition*. Cada servidor y sistema de expansión se puede configurar para que copie la configuración, *firmware* y datos binarios de teléfonos de manera local y periódicamente; tanto *Voicemail Pro* como Avaya *one-X® Portal for IP Office* poseen funciones de copias de seguridad y restauración.

3.3.4. Acceso remoto.

Se puede acceder a la solución IP *Office Server Edition* de manera remota. Algunos de los componentes a los que se puede acceder de manera remota son:

- *System Status Application*
  - Nombre de usuario y contraseña RBAC[29].

[29] *Role-Based Access Control* (RBAC) ofrece a los clientes la capacidad de crear cuentas de administración basadas en los roles definidos por el mismo cliente. Las funciones definidas por

  - La contraseña se transfiere de manera segura.
  - El puerto SSA se puede desactivar.
- *System Monitor*.
  - Acceso después de intercambiar contraseña
- Servicios de soporte de IP *Office* o SSL[30] VPN[31].
  - El sistema utiliza HTTPS y canal TLS.
  - La contraseña se transfiere de manera segura.
  - Solicita que los datos se re-ingresen periódicamente.
- *Mobility* y clientes Avaya *one-X® Portal.*
  - El nombre de usuario y contraseña es la configuración de IP *Office*, administrados mediante IP *Office Server Edition Manager*.

3.3.5. Capacidades y escalabilidad.

La solución IP Office Server Edition es compatible con un máximo de 1000 usuarios o extensiones.

- Máximo de usuarios de la solución: 1000.
- Máximo de usuarios por servidor/expansión: 500/250 por cada Primario, 500 por cada Secundario, 500 por cada expansión Linux, 384 por cada expansión IP500 V2.
- Extensiones totales: 1000 en toda la solución.
- Red multisitio. Máximo de dispositivos: 32.
- Líneas troncales SIP registradas: 4250.
- Máximo posible de canales activos para líneas troncales SIP: 4352.
- Procesamiento de llamadas. Capacidad de llamadas del servidor DL360 (BHCC[32]): 14400 sin usuarios one-X Portal activos. 7200 con usuarios one-X Portal activos.

---

el cliente se pueden adaptar para dar a cada administrador solo los privilegios de acceso que se necesitan para realizar el trabajo de dicho administrador.

[30] *Transport Layer Security* (en español seguridad de la capa de transporte).

[31] Red privada virtual.

[32] *Busy-hour call completion*.

- Capacidad de llamadas del servidor DL120 (BHCC): 10800 sin usuarios *one-X Portal* activos. 7200 con usuarios *one-X Portal* activos.
- Capacidad de expansión de llamadas IP500 V2 (BHCC): 7200.
- Capacidad general de llamadas (BHCC): 14400/7200.
- Llamadas VoIP concurrentes: medios directos: 1000.
- Llamadas VoIP concurrentes: sin medios directos: 256 por servidor (128 en un servidor pequeño). 128 por cada expansión Linux. 120 por cada expansión V2 (VCM[33]). 60 por cada expansión V2 (relevador RTP). Máximo 4352.
- Canales de transcodificación/VCM: 256 cada servidor (128 en un servidor pequeño). 128 por cada Sistema de expansión Server Edition (L). 148 Sistema de expansión Server Edition (V2). Máximo 4352.
- Máximo de grupos de búsqueda: 200.
- Tamaño del grupo de búsqueda: 500 usuarios.
- Total de miembros del grupo de búsqueda: 2000.
- Canales de conferencia: 4096.
- Tamaño de la conferencia: 64.
- Canales de grabación: 100.
- Tamaño del grupo de paginación: 64.
- Buzones: 1000.
- Canales del buzón de voz/auto asistentes: 100 (40 en un servidor pequeño).
- Capacidad de almacenamiento de mensajes: 1000 hrs.
- Canales TTS: 40.
- Correo electrónico por voz IMAP/SMTP/MAPI: 1000 usuarios.
- Integración con *Exchange* MAPI: 490 usuarios.
- Clientes activos de *One-X Portal*: 100, 200 o 500.
- Capacidad de usuarios/ extensión de copias de seguridad del servidor: 500 o 250.

[33] *Voice Compression Modules*.

- Tiempo de conmutación por error para teléfonos individuales: 2-3 minutos.
- Tiempo de conmutación por error completo del servidor Menos de 10 minutos
- Capacidad de directorio del sistema: 5000 entradas.
- Capacidad del directorio personal: 100 por usuario. En total 10800 por dispositivo.

3.4 Teléfono IP modelo 1616.

El teléfono 1616 se conecta con el sistema telefónico mediante una conexión IP y admite varias funciones adicionales:

- Utiliza una conexión *Ethernet* 10/100 para conectarse con el sistema telefónico a través de una red IP.
- Incluye una conexión a PC a través de un puerto Ethernet 10/100 que también se puede utilizar para conectar un ordenador a la red a través de la misma conexión que el teléfono. Si no se admite este puerto, el administrador del sistema podrá desactivar su utilización.
- El teléfono podrá recibir alimentación a través de la red si está disponible la función alimentación a través de *Ethernet* (PoE[34]). De lo contrario, necesitará su propia unidad de fuente de alimentación.

Entre las muchas funciones de este aparato, destacan los códigos de cuenta, con los que el sistema telefónico puede almacenar varios códigos de cuenta. Estos códigos se pueden utilizar para realizar un seguimiento de las llamadas con relación a actividades o clientes en particular. También se pueden utilizar para realizar un seguimiento de todas las llamadas para usuarios o conjuntos de usuarios. Cuando se ingresa un código de cuenta mientras se realiza una llamada o durante ella, el sistema incluye ese código de cuenta en la salida del registro de llamadas. Las cuentas que se ingresan se controlan con aquellas almacenadas en el sistema del teléfono. Si se ingresa un número no válido, se solicitará el código de cuenta nuevamente. El administrador del sistema puede

[34] *Power over Ethernet*.

configurar determinados números o tipos de números para que soliciten que se ingrese un código de cuenta para poder continuar con la llamada a ese número. El administrador del sistema también puede cambiar la configuración para que el operador deba ingresar un código de cuenta para poder realizar llamadas externas.

Figura 3.3. Teléfono IP modelo 1616.

3.5. Teléfono digital modelo 5420.

El télefono digital 5420 de Avaya es un terminal telefónico especialmente diseñado para recepcionistas o usuarios avanzados, cuyas principales características son:

10 teclas de funciones fijas: Conferencia, Transferencia, Interrupción, Llamada en espera, Rellamada, Silencio, Subir/Bajar volumen, Altavoz y *Voice Mail*.

- 24 teclas programables para presentación de llamadas/funciones (distribuidas en 3 páginas de display de 8 teclas que coinciden con los 8 botones físicos del display).
- Equipo funcional de primera calidad con registro de llamadas locales y directorio de marcado rápido para mayor productividad.
- Interfaz de usuario avanzado.

- Costos de instalación y mudanza reducidos; no lleva etiquetas de papel.
- Protección de la inversión con *firmware* descargable.
- Soporte ajustable de escritorio, tanto para escritorio como para ser fijado a la pared.
- Totalmente listo para el mercado global (se usan iconos para indicar la funcionalidad de los botones fijos).
- Display de 7x29 caracteres.
- Altavoz y micrófono manos libres de dos vías.
- Opción para deshabilitar el altavoz (vía Administración del Sistema).
- Función manos libres para escuchar la llamada en forma grupal.
- 9 teclas para funciones fijas debajo del *display*.
- 7 teclas de navegación del *display* (4 teclas multifunción, 3 teclas fijas).
- Indicador de gran tamaño para mensajes en espera.
- Botón exclusivo para recuperar voicemails.
- Conexión para microteléfono.
- Adaptable al idioma local (opción de elegir el idioma para el menú del teléfono local).
- Enchufe para usarlo con el módulo de expansión EU24 de 24 botones.

Figura 3.4. Teléfono digital modelo 5420.

Capítulo 4

Estudio del protocolo SMDR.

4.1. Definiciones generales.

4.1.1. Servicio. Se entiende servicio como la capacidad que tiene un dispositivo de realizar tareas y al tiempo ofrecerlas a otros. En una red de comunicaciones, el servicio lo ofrece una aplicación que se ejecuta en un dispositivo. A su vez, otro dispositivo conectado a esa misma red puede hacer uso de este servicio. Normalmente, un diseñador que necesita desarrollar una aplicación de red tiene que definir en primer lugar el interfaz entre las entidades que van a formar parte de la comunicación. El proceso de definición de un interfaz incluye la definición de un lenguaje común, denominado protocolo, y el diseño y formatos de los mensajes que se intercambiarían. Al no existir un modelo universal para la realización de estas tareas, los distintos protocolos y formatos desarrollados hasta la fecha carecen de homogeneidad. Además, normalmente es necesario configurar un dispositivo antes de poder usar un servicio. Por ejemplo, configurar correctamente la dirección de red, comprobar que el servicio está operativo o saber cuantas instancias del mismo servicio se encuentran actualmente en la red son funciones relevantes para las aplicaciones. Por lo tanto, se necesita un administrador de red que resuelva todos estos problemas. Ya que se puede esperar que cada usuario tenga el conocimiento de un experto en redes, estas tareas de configuración deben ser asumidas por las propias aplicaciones. En este punto es donde entra en juego el descubrimiento de servicios.

4.1.2. Descubrimiento. En general, los *frameworks* de descubrimiento de servicios son un conjunto de herramientas que sirven para desarrollar aplicaciones de red. En lugar de introducir nuevos conceptos, estos tratan de organizar y estandarizar la forma en que se diseñan, desarrollan e implementan dichas aplicaciones. Sea una cámara digital con conexión de red, por ejemplo. Para compartir fotos, se necesita conectar la cámara a una red. Después de esto, la cámara todavía necesitaría conocer su dirección de red. Esto puede ocurrir de varias formas, ya sea a través de un servidor central o de forma distribuida. Posteriormente, se debe usar un protocolo de transporte común para poder transmitir la información (fotografías) a otros dispositivos. Esto

normalmente implica la instalación de cierto *software* (controladores) en la máquina que se comunicaría con la cámara. Para hacer todo este proceso automático, las soluciones de descubrimiento de servicios existentes en la actualidad incorporan herramientas que permiten realizar estas tareas, tales como asignación de direcciones de red o la instalación de controladores.

4.1.3. SLP. Hasta que no se produjo el auge de Internet, no se había desarrollado un protocolo de descubrimiento de servicios exclusivo para redes IP. El Grupo de Trabajo Srvloc de la *Internet Engineering Task Force* (IETF) creó el protocolo SLP[35] que, junto a *Salutation*, fueron las dos primeras soluciones de descubrimiento de servicios para redes IP. Cuando estas soluciones se combinan con el protocolo DHCP[36], obtenemos asignación automática de direcciones IP, además del descubrimiento de servicios. Aun así, estas soluciones todavía no liberan al usuario de la necesidad de instalar controladores.

4.1.4. SLPv2. El protocolo *Service Location Protocol* v2 forma parte de una familia de protocolos de descubrimiento de servicios diseñados para redes IP por el Grupo de Trabajo Srvloc de la IETF durante la segunda mitad de la década de los 90. Fue definido originalmente como SLP en el documento RFC 2165, que posteriormente fue sustituido por SLPv2 descrito por el RFC 2608. Es una solución limitada a redes que soporten el conjunto de protocolos TCP/IP. Al mismo tiempo, SLPv2 es totalmente independiente del lenguaje de programación y del sistema operativo usados para la programación y ejecución de las aplicaciones.

4.1.5. SAP. Un punto de acceso al servicio, del inglés *Service Access Point* (SAP), se define dentro de la arquitectura *Open Systems Interconnection* (OSI) y es una abstracción de la conexión entre entidades pertenecientes a capas contiguas de un sistema de comunicaciones. Así, los puertos en una red TCP/IP enlazan las capas de transporte y aplicación, permitiendo que un mensaje pueda ser correctamente encaminado a la aplicación de destino.

[35] *Service Location Protocol.* Protocolo de ubicación de servicio.

[36] *Dynamic Host Configuration Protocol*. Protocolo de configuración dinámica de *host*.

4.2. Protocolo SMDR.

La unidad de control (centralita) puede enviar registros SMDR (informe de detalles de mensajes de estación) al puerto y la dirección IP especificada en la configuración Parámetros avanzados[37]. Normalmente, un registro SMDR es un informe de cada llamada entre dos interlocutores cuando se completa la llamada. En algunos escenarios, por ejemplo, transferencias y conferencias, se pueden generar registros SMDR separados para cada porción de la llamada. Cada registro SMDR contiene información de la llamada con el formato de valores separados por coma (CSV), es decir campos de ancho variable, cada uno separado por comas (con la excepción del primer parámetro, separado por un espacio del siguiente).

4.2.1. Registros SMDR. Se genera un registro SMDR para cada llamada entre dos dispositivos del sistema IP *Office* (aunque otros fabricantes[38] han adoptado también este formato de registro[39]). Los dispositivos son, entre otros, extensiones, líneas de troncales (o canales de una troncal), canales de correo de voz, canales de conferencias y tonos de IP *Office*. Las llamadas que no se presentan a otro dispositivo no generan un registro SMDR. Por ejemplo, el código de acceso de marcado de usuarios internos que simplemente cambia un parámetro de configuración. El registro SMDR se genera cuando la llamada finaliza. Por lo tanto, el orden de generación de los registros SDMR no coincide con las horas de inicio de las llamadas. Cada registro contiene una ID de llamada que aumenta un número por cada llamada subsiguiente. Cuando una llamada se mueve de un dispositivo a otro, se genera un registro SMDR para la primera parte de la llamada y se generará un registro SMDR adicional para la porción subsiguiente de la llamada (el camino seguido por estas llamadas entre varias extensiones es irrelevante a efectos de este estudio). Cada uno de estos

[37] Puede observarse este proceso con detalle en el anexo correspondiente al final de este documento.

[38] Panasonic, Mitel, Polycom, Samsung.

[39] Con algunas variaciones.

registros tendrá la misma ID de llamada. Cada registro de una llamada indica en el campo Continuación si habrá más registros para ella. Las llamadas de despertador producen un registro SMDR incluso si la extensión objetivo estaba ocupada al momento de la llamada. El Interlocutor1 se muestra como Llamada de despertador.

4.2.1.1. Tiempos de llamada. Cada registro SMDR puede incluir valores para la duración del timbre (tiempo de espera en ser atendido, parámetro crítico en nuestro estudio), la duración de la conexión, el tiempo de retención y la duración del estacionamiento. La duración total de un registro SMDR es la suma de estos valores. El tiempo en el que una llamada no está en ninguno de los estados anteriores, por ejemplo, cuando un interlocutor de la llamada se ha desconectado, no se mide ni se incluye en los registros SMDR. Donde se usan anuncios, la duración de la conexión para una llamada comienza ya sea cuando se la atienden o cuando el primer anuncio comienza. Todos los tiempos se redondean al segundo más cercano. Cada registro SMDR tiene una hora de inicio de llamada tomada de la hora del reloj del sistema. Para las llamadas transferidas o sujetas a división, cada uno de los diversos registros SMDR tendrá la misma hora de inicio de llamada que la llamada original.

4.2.1.2. Campos SMDR. El informe de SMDR contiene los siguientes campos. Los valores de hora se redondean al segundo más cercano.

4.2.1.2.1. Inicio de la llamada. Hora de inicio de la llamada en formato AAAA/MM/DD HH:MM:SS. Para todos los segmentos de llamadas transferidas es la hora en la que se inició la llamada. Por ello, cada segmento de la llamada tiene la misma hora de inicio.

4.2.1.2.2. Duración de conexión. Duración de la porción conectada de la llamada en formato HH:MM:SS. No incluye la duración del timbre, la retención ni el estacionamiento. Las llamadas perdidas o erróneas tendrán una duración de 00:00:00. La duración total de un registro se calcula como Duración de conexión + Duración de timbre + Tiempo de retención + Duración de estacionamiento.

4.2.1.2.3. Duración de timbre. Duración de la porción de timbre de la llamada en segundos. Para las llamadas entrantes, representa el intervalo entre la

llamada que llega al conmutador y el momento en que se la atiende, no el tiempo que timbra en una extensión individual. Este parámetro es de especial relevancia en el presente proyecto. Para las llamadas salientes, indica el intervalo entre el inicio de la llamada y el momento en que se la atiende en el equipo remoto si el tipo de troncal la admite. Las troncales analógicas no pueden detectar respuestas remotas y, por lo tanto, no pueden proporcionar una duración de timbre para llamadas salientes.

4.2.1.2.4. Llamante. El número de la persona que llama. Si la llamada se originó en una extensión, será ese número de extensión. Si la llamada se origina externamente, será la CLI de quien llama si está disponible. De lo contrario, se dejará en blanco.

4.2.1.2.5. Dirección. La dirección de la llamada:

- E (I) para las llamadas entrantes, o
- S (O) para las llamadas salientes.

Las llamadas internas se representan como S (O) por "saliente". Este campo puede usarse en conjunto con Es interna a continuación para determinar si la llamada es interna, externa saliente o externa entrante.

4.2.1.2.6. Número al que se llamó. Este es el número al que IP *Office* llamó. Para una llamada que se transfiere, este campo muestra el número al que se llamó originalmente, no el número del interlocutor que transfirió la llamada.

- Llamadas internas: La extensión, el grupo o el código de acceso al que se llamó.
- Llamadas entrantes: El DDI marcado por quien realiza la llamada si está disponible.
- Llamadas salientes: Los dígitos marcados.
- Correo de voz: Llamadas a un buzón de correo de voz del usuario.

4.2.1.2.7. Número marcado. Para las llamadas internas y salientes es idéntico a lo estipulado en Número al que se llamó indicado anteriormente. Para las llamadas entrantes, es el DDI de quien realiza la llamada entrante.

4.2.1.2.8. Cuenta. Es el último código de cuenta adjuntado a la llamada. Nota: Los códigos de cuenta de IP *Office* puede contener caracteres alfanuméricos.

4.2.1.2.9. Es interna. Puede valer 0 ó 1, lo que denota si ambos interlocutores de la llamada son internos o externos (1 equivale a una llamada interna). Las llamadas a destinos de SCN[40] se indican como internas.

Dirección	Es interna	Tipo de llamada
E	0	Entrante externa
S	1	Interna
S	0	Saliente externa

4.2.1.2.10. ID de llamada. Es un número que comienza en 1.000.000 y aumenta de 1 en 1 por cada llamada única. Si la llamada generó varios registros SMDR, cada registro tendrá la misma ID de llamada. La ID de llamada usada se reinicia a partir de 1.000.000 si se reinicia IP *Office*.

4.2.1.2.11. Continuación. 1 si existe otro registro para esa ID de llamada, 0 en caso contrario.

4.2.1.2.12. DispositivoInterlocutor1. Es el dispositivo número 1. Generalmente es el que inicia la llamada, aunque en algunos escenarios, como en las conferencias, esto puede variar. Si el grupo de búsqueda/extensión participa en la llamada, sus detalles tendrán prioridad sobre una troncal. Incluye destinos de SCN remotos.

[40] Small Community Network o red comunitaria pequeña.

Tipo	Dispositivo de interlocutor	Nombre de interlocutor
Número interno	E<número de extensión>	<nombre>
Correo de voz	V<9500 + número de canal>	Canal VM <número de canal>
Conferencia	V<1><número de conferencia>+<número de canal>	Canal CO <número de conferencia.número de canal>
Línea	T<9000+número de línea>	Línea <número de línea>.<canal si corresponde>
Otro	V<8000 + número de dispositivo>	U<clase de dispositivo> <número de dispositivo>. <canal de dispositivo>
Desconocido/ Tono	V8000	U1 0.0

4.2.1.2.13. NombreInterlocutor1. El nombre del dispositivo para una extensión o agente. Es el nombre de usuario.

4.2.1.2.14. DispositivoInterlocutor2. El otro interlocutor del registro SMDR de este segmento de llamada. Consulte DispositivoInterlocutor1 en epígrafe anterior.

4.2.1.2.15. NombreInterlocutor2. El otro interlocutor del registro SMDR de este segmento de llamada. Consulte NombreInterlocutor1 anteriomente indicado.

4.2.1.2.16. Tiempo de retención. La cantidad de tiempo en segundos en el que la llamada se retuvo durante este segmento de llamada.

4.2.1.2.17. Duración de estacionamiento. La cantidad de tiempo en segundos en el que la llamada se estacionó durante este segmento de llamada.

4.2.1.2.18. Validación de autorización. Este campo se usa para los códigos de autorización. Este campo muestra 1 para las autorizaciones válidas o 0 para las que no son válidas.

4.2.1.2.19. Código de autorización. Este campo muestra el código de autorización usado o n/a si no se usó ningún código de autorización.

4.2.1.2.20. Usuario cargado. Este campo y los subsiguientes se usan para el aviso de cargo (AoC) de ISDN. El usuario al que se le asignaron los cargos de la llamada. No necesariamente es el usuario que participa en la llamada.

4.2.1.2.21. Cargo de llamada. El cargo total de la llamada calculado usando el costo de la línea por unidad y marcación de usuario.

4.2.1.2.22. Divisa. Indica la divisa utilizada. Es un parámetro de todo el sistema en la configuración de IP *Office*.

4.2.1.2.23. Monto en cambio de último usuario. El monto de AoC en el cambio de usuario.

4.2.1.2.24. Unidades de llamada. Indica el número de unidades totales de llamada.

4.2.1.2.25. Unidades en cambio de último usuario. Las unidades de AoC actuales en el cambio de usuario.

4.2.1.2.26. Costo por unidad. Este valor se establece en la configuración de IP *Office* en comparación con cada línea en la que se establece la señalización Aviso de cargo. Los valores son 1/10.000 de una unidad de moneda. Por ejemplo, si el costo de la llamada por unidad es de £1,07, en la línea debe establecerse un valor de 10700.

4.2.1.2.27. Marcado. Indica el valor de marcado establecido en la configuración de IP *Office* para el usuario al que se le cobra la llamada. Las unidades del campo son 1/100, por ejemplo, una entrada de 100 es un factor de marcado de 1.

4.2.1.2.28. Causa de direccionamiento externo. Es el campo que indica quién o qué causó la llamada externa y un código de motivo. Por ejemplo, U FU indica que la llamada externa se originó debido a la configuración Remisión incondicional de un usuario.

Destinado por		Código de motivo	
HG	Grupo de búsqueda	fb	Remitir si ocupado.
U	Usuario	fu	Remisión incondicional.
LINE	Línea	fnr	Remitir si no hay respuesta.
AA	Operadora automática	fdnd	Remitir si DND[41].
ICR	Ruta de llamada entrante	CfP	Llamada de propuesta (consulta) de conferencia
RAS	Servicio de acceso remoto	Cfd	Conferencia
?	Otro	MT	Hermanamiento de teléfonos móviles.
		TW	Trabajador remoto (Modo Teletrabajador de *Phone Manager*).
		XfP	Llamada de propuesta (consulta) de transferencia.
		Xfd	Llamada transferida.

4.2.1.2.29. ID de direccionamiento externo. El nombre asociado de quien establece el destino indicado en el campo *Causa de direccionamiento externo*. Para los grupos de búsqueda y usuario será su nombre en la configuración de IP *Office*. Para una ruta de llamada entrante será la etiqueta si se la establece. De lo contrario, será ICR[42].

4.2.1.2.30. Número de destino externo. Este campo se usa para llamadas direccionadas de rutas de llamadas entrantes remitidas y hermanamiento móvil a una línea externa. Muestra el número externo al que IP *Office* llama como resultado de la determinación de destino fuera del conmutador donde otros campos del llamante dan el número de marcado original.

[41] Do not Disturb o no molestar.

[42] Intelligent Customer Routing o enrutado de cliente inteligente.

4.2.2. Ejemplos SMDR.

4.2.2.1. Llamada entrante perdida.

En este registro, la duración de la llamada es cero y el campo Continuación está en 0, lo que indica que la llamada nunca se conectó. En Duración de timbre, se muestra que sonó por 9 segundos antes de finalizar.

28/06/2008 09:28:41,**00:00:00**,9,8004206,I,4324,4324,,0,1000014155,0,
E4324,José Ruiz,T9161,LÍNEA 5.1,0,0

4.2.2.2. Llamada externa.

El campo Es interna en 0 muestra que esta es una llamada externa. El campo Dirección muestra que fue una llamada entrante. El valor de Duración de timbre fue de 7 segundos y el total de Duración de conexión fue de 5 segundos.

01/08/2008 15:14:19,**00:00:05,7**,01707299900**,I,**23,390664,,0,1000013,0,E23,
Extn23,T9001,Línea 1.2,0,0,,,,,,,,,,,,,

4.2.2.3. Llamada estacionada.

En este ejemplo, el primer registro tiene una duración de estacionamiento que muestra que la llamada se estacionó. El campo *Continuación* indica que la llamada no finalizó de este modo y que existen más registros. El segundo registro tiene la misma ID de llamada y muestra un cambio en el NombreInterlocutor2, lo que indica que el interlocutor anuló el estacionamiento de la llamada. Observe también que ambos registros comparte la misma hora de inicio de llamada.

20/10/2008 07:18:31,0:00:12,3,215,O,210,210,,1,38,1,E15,
Extn15,E10,Extn10,0,7
20/10/2008 07:18:31,0:00:10,0,215,O,210,210,,1,38,0,E15,Extn15,E11,
Extn11,0,0

4.2.2.4. Estacionar y no estacionar.

El estacionamiento y la anulación del estacionamiento de una llamada en la misma extensión simplemente se muestra en el campo *Duración de estacionamiento* del consiguiente registro SMDR. Del mismo modo, las llamadas retenidas y cuya retención se anula en la misma extensión se muestran en el campo *Duración de retención* del consiguiente registro SMDR para la llamada. No obstante, los registros a continuación muestran una llamada estacionada en una extensión y luego con el estacionamiento anulado en otra. Los registros muestran una llamada de 17 a 13. 13 luego estaciona la llamada mostrada por *Duración de estacionamiento*. 11 anula el estacionamiento de la llamada y el primer registro se indica como continuado en el campo Continuación. La ID de llamada coincidente indica el registro subsiguiente para la llamada.

```
09/07/2008 16:39:11,00:00:00,2,17,O,13,13,,1,1000052,1,E17,
Extn17,E13,Extn13,0,4
09/07/2008 16:39:11,00:00:02,0,17,O,13,13,,1,1000052,0,E207,
Extn17,E11,Extn11,0,0
```

4.2.2.5. Llamada externa con repetición de encaminamiento.

En este ejemplo, se ha vuelto a enrutar una llamada entrante externa fuera del conmutador, esto está evidenciado por los campos Interlocutor 1 e Interlocutor 2 que contienen los detalles de una línea externa. La causa de direccionamiento externo muestra que una ruta de llamada entrante (ICR) volvió a encaminar la llamada entrante. La ID de direccionamiento externo en este caso es la identificación establecida para la ruta de llamada entrante. El número de destino externo es la llamada del número externo real.

```
…  08:14:27,00:00:03,5,392200,I,9416,200,,0,1000073,0,T9005,Línea
5.1,T9005,Línea 5.2,0,0,,,,0000.00,,0000.00,0,0,618,0.01,
ICR,ICR principal,416,
```

Capítulo 5

Desarrollo del *software*

5.1. Herramienta de desarrollo.

Para el desarrollo de la herramienta de *software* que constituye el eje central de este proyecto se ha utilizado un entorno de desarrollo denominado *Lazarus*, basado en *Free Pascal*, heredero del Pascal y del Turbo Pascal y que se ha constituido un entorno alternativo a Embarcadero Delphi.

5.1.1. Descripción de *Free Pascal* y Lazarus.

Free Pascal (FPC) es un compilador de Pascal [Wiki Lazarus 2015] de código abierto con dos características notables: un alto grado de compatibilidad con Delphi y disponibilidad en una variedad de plataformas, incluyendo OS X, Windows, Mac, y Linux. La compatibilidad de *Free Pascal* con Delphi incluye no solo la ayuda para el mismo lenguaje de programación *Object Pascal* que utiliza Delphi, sino también para muchas de las mismas bibliotecas de rutinas y de clases de gran alcance por las que Delphi es conocido. Esto incluye unidades habituales tales como System, SysUtils, StrUtils, DateUtils, Classes, Variants, Math, IniFiles y Registry, que se incluyen con *Free Pascal* en todas las plataformas soportadas. *Free Pascal* también incluye unidades tales como Windows, ShellAPI, BaseUnix, Unix y DynLibs para acceder a características específicas de un sistema operativo. Esta docena, más o menos, de unidades se denomina generalmente como la biblioteca de tiempo de ejecución de *Free Pascal* (RTL[43]).

Lazarus es un sistema de desarrollo de código abierto que trabaja sobre el compilador *Free Pascal* agregando un entorno integrado de desarrollo (IDE) que incluye un editor de código con resalte de sintaxis y un diseñador de formularios visual, así como una librería de componentes que es altamente compatible con la biblioteca de componentes visual de Delphi (VCL). La librería de componentes de *Lazarus* (LCL) incluye los equivalentes para muchos de los controles familiares de VCL tales como formas, botones, cajas de texto y más

[43] *Run Time Library*.

que se utilizan para crear aplicaciones que tienen un interfaz gráfico de usuario (GUI). Tanto *Free Pascal* como *Lazarus* están escritos en Pascal. El código fuente completo está disponible no solamente para el compilador de *Free Pascal* y el IDE de *Lazarus*, sino también para todas las unidades que construyan *Free Pascal* RTL y *Lazarus* LCL.

Como Delphi, tanto *Free Pascal* como *Lazarus* son herramientas de programación de uso general, significando que se puede desarrollar una variedad amplia de programas con ellos, incluyendo lo siguiente:

5.1.2.1. Aplicaciones de consola.

Las aplicaciones de consola no tienen un GUI. En su lugar se lanza la consola, se lee su entrada de la consola, y se escribe generalmente su salida en la consola. En *Windows* la consola es generalmente la ventana del aviso de comando. En OS X y Linux la consola es la ventana terminal. Las aplicaciones de consola incluyen cosas como utilidades pequeñas tales como el programa de *Windows* FC (File Compare) o los comandos de Unix cd y cp. Las aplicaciones de consola pueden también ser utilizadas por los programas de proceso de datos que no necesitan un GUI porque son arrancados por otros programas o desde archivos por lotes. El compilador *Free Pascal* y los programas utilitarios incluidos con él son todos aplicaciones de consola, lo que significa que pueden ejecutarse desde la consola, desde un archivo por lotes, o desde el IDE de *Lazarus*.
Se puede crear una aplicación de consola sin otra cosa que un editor de textos y el compilador *Free Pascal*. No tiene se que utilizar *Lazarus* para desarrollar aplicaciones de consola. Sin embargo, si se prefiere trabajar en un ambiente integrado, se puede utilizar *Lazarus* para crear un proyecto de aplicación de consola y editar y compilar el código en el IDE de *Lazarus*.

5.1.2.2. Bibliotecas cargables dinámicamente.

Una biblioteca cargable dinámicamente es generalmente una colección de funciones compiladas que se pueden llamar por otros programas. Como el

nombre sugiere, la biblioteca no se enlaza con su ejecutable en tiempo de compilación, sino que por el contrario se carga en el tiempo de ejecución. En *Windows*, un archivo de la librería tiene una extensión .dll (*dynamic-link library* o biblioteca de enlace dinámico, también conocida como DLL). En OS X, un archivo de biblioteca tiene una extensión .dylib (*dynamic shared library*, o librería compartida dinámica). En Linux, un archivo de la librería tiene una extensión .so (*shared object library* o librería de objetos compartidos). Las biblioteca cargables dinámicamente se utilizan típicamente para desarrollar complementos para otros programas, desarrollar biblioteca que se pueden llamar por los programas escritos en otras lenguajes tales como C y C++, o para descomponer proyectos grandes en trozos de modo que los desarrolladores del proyecto no se entorpezcan unos a otros. *Windows* se compone de centenares de DLLs, al igual que muchos otros programas grandes tales como *OpenOffice*.org.

Igual que para las aplicaciones de consola, solamente se necesita un editor de textos y el compilador *Free Pascal* para desarrollar una biblioteca, aunque se puede también crear un proyecto de *Lazarus* para una biblioteca y desarrollarla en el IDE de *Lazarus*.

En *Windows*, las DLLs tienen reputación de ser complejas e inestables. Esto tiene más que ver con la manera en que se instalan a veces, que con las propias DLLs. Realmente, el paso de la información "a" y "desde" DLLs se hace normalmente con los tipos de datos típicos, simples (más que con los objetos o las estructuras específicos de un lenguaje), este requisito obliga a los programadores a prestar más atención a lo que se está haciendo. Si se hace de forma correcta, se obtendrán programas mejores, más estables.

5.1.2.3. Aplicaciones con GUI.

La mayoría de los programas que utilizamos a diario son aplicaciones con GUI, incluyendo los procesadores de textos, navegadores Web, programas de hoja de cálculo, incluso muchas herramientas de desarrollo. *Lazarus* y *Delphi* son ambos buenos ejemplos de aplicaciones completamente provistas de GUI. Al desarrollar una aplicación con GUI con *Lazarus*, se crean no solamente

unidades en código Pascal, también diseña los formularios que contienen controles visuales tales como botones o cajas de lista. Como *Delphi*, el diseño del formulario en *Lazarus* se hace visualmente. Las características del control se pueden fijar en el IDE o mediante código.

Puesto que los controles de LCL están disponibles en todas las plataformas soportadas, una aplicación con GUI desarrollada en una plataforma (por ejemplo, *Windows*) se puede compilar en otra plataforma (por ejemplo, OS X o Linux) sin ningún cambio en el diseño del formulario o en los archivos de su código.

5.1.2. Arquitectura de Lazarus.

Los programas escritos en *Lazarus* respetan la arquitectura indicada en la figura 5.1.

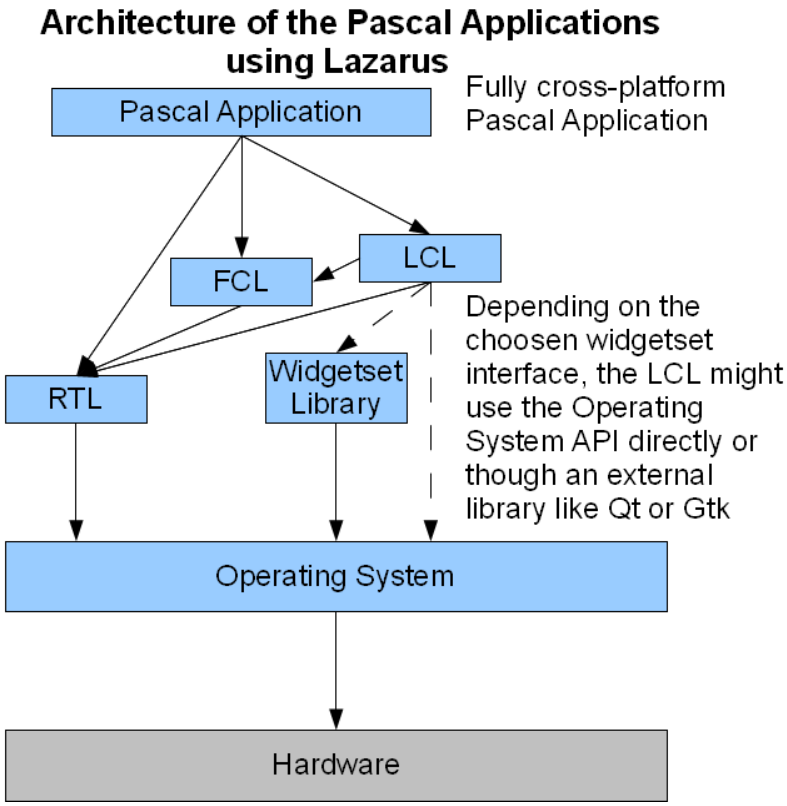


Figura 5.1. Arquitectura *Lazarus*.

Y la propia LCL sigue la arquitectura mostrada en la figura 5.2.

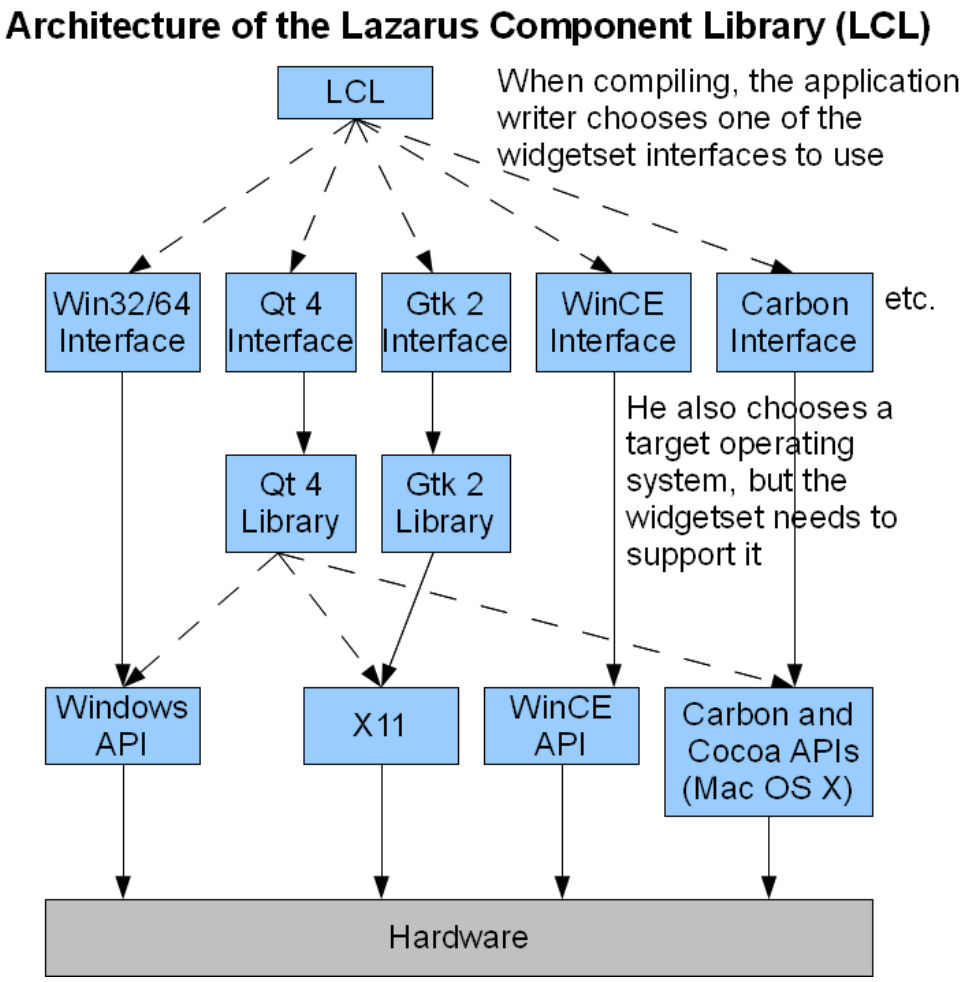


Figura 5.2. Arquitectura LCL.

La LCL cuenta con unidades independientes de la plataforma, al igual que los controles, formularios, botones y rejillas. Estos no pueden trabajar por su cuenta. Son abstractos y requieren de un servidor. La LCL ofrece muchas interfaces diferentes que se comunican con *backends* o "*Widgetsets*" (Win32, GTK, QT, carbon, el cocoa). Un motor será compilado por la mera utilización del paquete LCL y la adición de la unidad de "interfaces" como una de las primeras unidades de cada programa. Al crear una aplicación con el IDE de *Lazarus* esto se hace automáticamente. El motor se determina en tiempo de compilación y no se puede cambiar en tiempo de ejecución.

5.2. Analizador SMDR.

5.2.1. Análisis funcional.

El objetivo de todo análisis funcional es describir las funcionalidades del sistema/aplicativo mediante modelos o documentos de análisis. El presente análisis identifica las interacciones con los diversos elementos externos y documenta las estructuras de información necesarias para completar el desarrollo. Este análisis ha permitido:

- Explicar las necesidades de la fase de desarrollo y qué funcionalidades ha sido necesario que implementar.
- Estimar el esfuerzo que se ha debido realizar para obtener la solución.
- Plantear las pruebas necesarias para comprobar que lo desarrollado es lo que se buscaba.
- Garantizar el mantenimiento y la funcionalidad de la aplicación, con lo que los evolutivos o correctivos no serán traumáticos.

Si bien estos aspectos son importantes, cabe destacar que este Análisis Funcional es una fase dentro del ciclo de vida del desarrollo, por lo que la labor desarrollada de análisis no finalizó cuando acabó tal fase, sino que el proceso es circular, replanteando esta fase (al igual que las siguientes) constantemente con el objetivo de realizar mejoras a las funcionalidades planteadas. Para este desarrollo se ha utilizado uno de los métodos más extendidos en la ingeniería del *software* para realizar el análisis funcional, los Casos de Uso. Casos de Uso ha permitido capturar requisitos y guiar la construcción del *software*. En este caso particular, el caso de uso ha tenido una meta bien establecida, una estructura entendida por todos los involucrados en el proyecto (redactor, tutores), pero también un conjunto de exigencias que el sistema debía satisfacer.

El caso de uso, que desmenuza el problema principal en otros más pequeños, ha permitido formar un elemento de trabajo de valor claro para el usuario. Las diferentes subdivisiones de casos de uso son requisitos y pruebas y evolucionan para incluir las diferentes partes necesarias a través del diseño, la

implementación y las pruebas. Así, las porciones de caso de uso han posibilitado:

- Que los casos de uso fueran divididos en unidades de trabajo más pequeñas e independientemente implementables.
- Que los requisitos fuesen ordenados, priorizados y tratados en paralelo.
- Enlazar los distintos modelos desarrollados (requisitos, análisis, diseño, implementación y pruebas) usados en el desarrollo.

Para ello, lo primero ha sido identificar la función más útil que la aplicación debía realizar con el objeto de centrar el trabajo en ello. Esta función principal se dividió en porciones más pequeñas, de entre las cuales se seleccionó la más importante, de valor transversal en el proceso de desarrollo. En ese momento se empezó a construir. Posteriormente se añadieron casos de prueba previamente definidos para esa porción.

Este proceso se repitió para todas las porciones y casos de uso que fue necesario dividir. De esta forma, la aplicación fue creciendo por incrementos, de tal forma que cada incremento se construyó sobre el incremento previo para añadir más funcionalidad y mejorar la calidad de lo que se había hecho antes.

5.2.1.1. Requisitos Funcionales frente a Casos de Uso.

En este desarrollo se ha realizado una Toma de Requisitos bastante detallada para posteriormente diseñar un Análisis Funcional basado en los requisitos recogidos. De esta forma se capturaron y describieron los requisitos y después se trasladó a un modelo de casos de uso, donde se ha vuelto a describir la funcionalidad deseada. Este proceso ha permitido un enfoque más afinado, orientado al usuario y con una estructura más clara.

5.2.1.2. Resultado del análisis.

Esta aplicación de análisis de ficheros funciona de la siguiente manera:

El proceso comienza con la apertura de un fichero de datos. Se trata de un fichero con formato CSV, esto es, con valores separados por comas. De este fichero solo se utilizarán los primeros valores (se cuenta el número de comas

hasta llegar al valor “I”, que identifica a las llamadas entrantes. El aspecto del fichero de datos es parecido al siguiente:

```
2002/05/28 00:30:00,00:01:00,3,8004206,I,4324,4324,,0,1000014155,0,E4324,Joe Bloggs,T9161,LINE 5.1,0,0
2002/05/28 07:21:00,00:00:30,5,01707392200,I,299999,299999,,0, ,1,E4750,John Smith,T9002,LINE 1.2,11,0
2002/05/28 07:22:00,00:02:00,8,8004206,I,4324,4324,,0,1000014155,0,E4324,Joe Bloggs,T9161,LINE 5.1,0,0
2002/05/28 07:23:00,00:01:30,5,01707392200,I,299999,299999,,0, ,0,V9502,VM Channel 2,T9002,LINE 1.2,0,7
2002/05/31 03:55:02,00:01:51,9,4797,O,08000123456,08000,,0,1000014129,0,E4797,Joe Bloggs,T9001,LINE 1.1,0,0
2002/05/31 04:28:41,00:00:01,9,8004206,O,4324,4324,,0,1000014155,0,E4324,Joe Bloggs,T9161,LINE 5.1,0,11
2002/05/31 23:45:00,00:00:19,3,4966,I,VoiceMail,VoiceMail,,1, ,0,E4966,John Smith,V9501,VM Channel 1,0,0
```

En este ejemplo sucinto se observan 7 llamadas realizadas en diferentes días de mayo de 2015. Las fechas con las que se trabaja mantienen el formato aaaa/mm/dd. El siguiente dato es la hora de la llamada. El siguiente su duración y el siguiente el tiempo de timbre en segundos. Dos campos más adelante se observa un carácter en mayúsculas, que puede ser una “I” o una “O” dependiendo de si se trata de una llamada entrante o saliente. Estos aspectos se detallan en el capítulo de análisis de los campos de los ficheros SMDR.

Tras la apertura del fichero de datos y en caso de que no contenga errores, pasamos a analizar seleccionando la opción correspondiente. Las flechas nos guían en el proceso para hacerlo intuitivo. Los ficheros que no contienen llamadas entrantes no se valoran.

El componente *TPageControl* nos abre una serie de pestañas que nos permiten seleccionar tanto el visionado de datos generales como el de datos en valores medios así como diferentes gráficas, que describiremos más adelante.

Finalizado el visionado de estos valores y gráficas podemos grabar un informe en un fichero, cerrar el fichero de datos abierto actualmente o cerrar la aplicación.

5.2.2. Análisis orgánico (diseño).

El análisis orgánico ha sido una parte importante del proceso de programación y diseño propuesto. Este análisis partió de un análisis previo descriptivo del programa a diseñar, que a su vez derivó en el análisis funcional anteriormente

detallado. El análisis previo ha permitido describir el comportamiento del programa y las necesidades a cubrir. Por su parte, el análisis orgánico ha dado lugar a la estructura del programa, esto es, la forma en la que diseñar para cumplir lo que el análisis funcional ha descrito. Este análisis ha permitido determinar la estructura del programa y cómo implementar cada una de las funciones en el mismo.

Para un análisis adecuado, y sabiendo lo que se quería diseñar, se buscó la respuesta a las preguntas clave ¿Qué herramientas hay accesibles para hacerlo? ¿Cuál de las existentes será la más apropiada? ¿Como será mejor hacerlo?

El proyecto de *software* se dividió en varias unidades orgánicas, que fueron estudiadas exhaustivamente. Tras esta fase se decidió qué herramienta emplear (en este caso, *Lazarus Free Pascal*).

Una vez terminado el análisis orgánico, se procedió a la programación [Menkaura 2015] siguiendo la estructura y las indicaciones obtenidas en el análisis orgánico.

La aplicación no requiere de instalación y puede ejecutarse en *Microsoft Windows* versión 7 y versión 8.1, sistemas operativos de 64 bits.

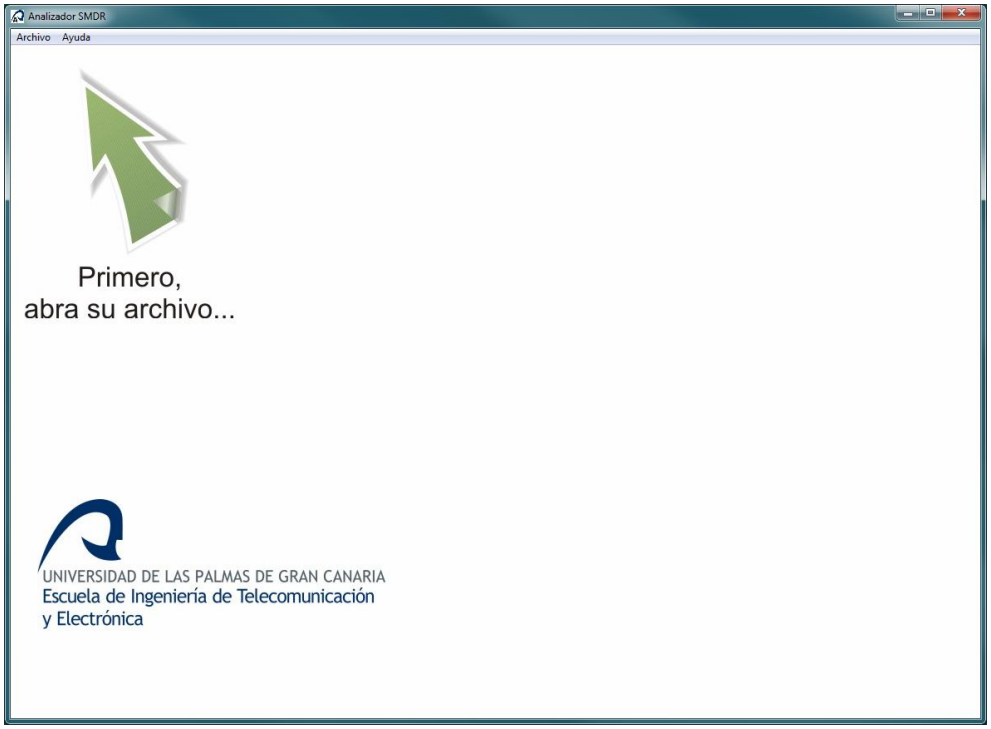


Figura 5.2.1. Pantalla de inicio.

La primera pantalla de la aplicación (fig. 5.2.1) nos lleva de forma intuitiva a la apertura del fichero de datos SMDR. Este fichero puede tener la extensión txt o csv. También tenemos acceso a la opción “Acerca de…”, que detalla los créditos de la aplicación.

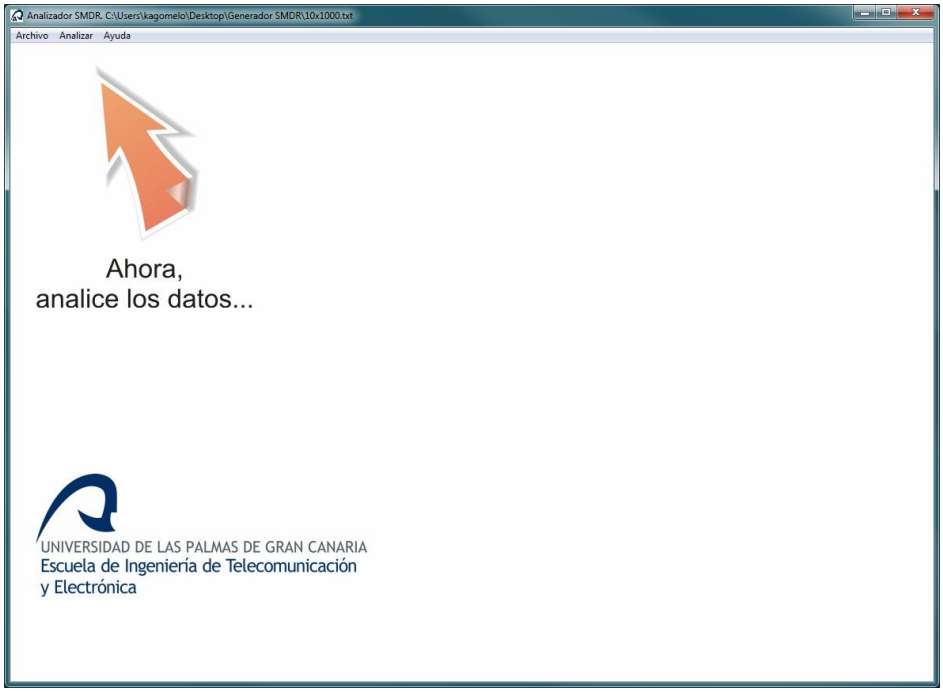


Figura 5.2.2. Pantalla tras carga de datos.

Tras la apertura del fichero de datos, una nueva flecha de color anaranjado nos invita a empezar el análisis. Si el fichero supera los 100.000 registros, esta operación puede tardar unos segundos dependiendo de *hardware* utilizado. En esta operación quedan registradas en memoria las llamadas entrantes, lo cual permite un rápido cálculo de las diferentes variables y parámetros necesarios.
Si el fichero de datos no contiene llamadas entrantes, nos informa con un mensaje de tal hecho y nos devuelve al punto de partida para que abramos un nuevo fichero de datos.

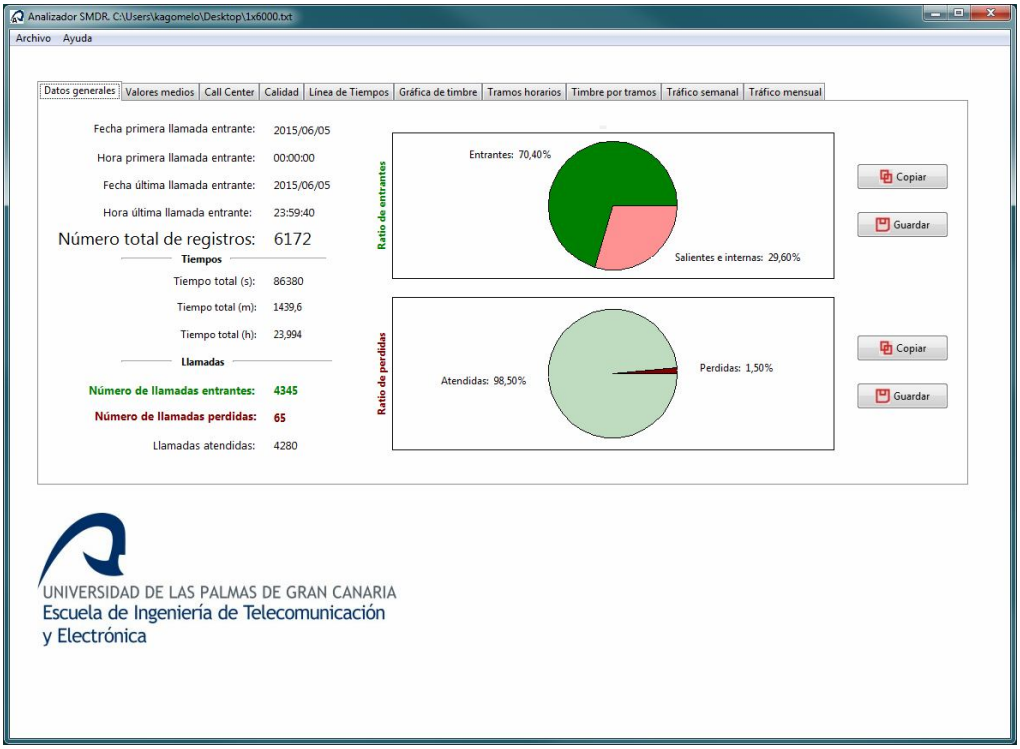


Figura 5.2.3. Datos generales.

La organización de la información ofrecida se organiza en pestañas tal y como se muestra en la figura. La primera pestaña muestra los datos generales observados en el fichero. A saber:

- Fecha de la primera llamada entrante, que se corresponde con la primera llamada entrante existente en el fichero.
- Hora de la primera llamada entrante que consta en el fichero SMDR.
- Fecha de la última llamada entrante.
- Hora de la última llamada entrante.
- El tiempo total transcurrido entre la primera llamada y la última (este parámetro nos permitirá calcular el grado de uso de la centralita) en segundos.
- Ídem en minutos.
- Ídem en horas.
- Número total de registros que contiene el fichero.
- Número de llamadas entrantes en total.
- Número de llamadas perdidas, esto es, de duración cero.

- Número de llamadas atendidas, que resulta de la resta de las entrantes menos las perdidas.

A esta serie de valores le acompañan dos gráficas del tipo tarta. La primera muestra el ratio de llamadas entrantes enfrentado al resto del tráfico, esto es, llamadas salientes e internas. La segunda indica el tráfico perdido, o sea, las llamadas entrantes de duración cero. Cada gráfica dispone a su derecha de dos botones. Uno de ellos copia la misma en el portapapeles. La otra permite guardarla en formato JPG. Estos botones están presentes en todas las gráficas y permiten la creación de informe gráfico junto al informe de datos que genera esta aplicación.

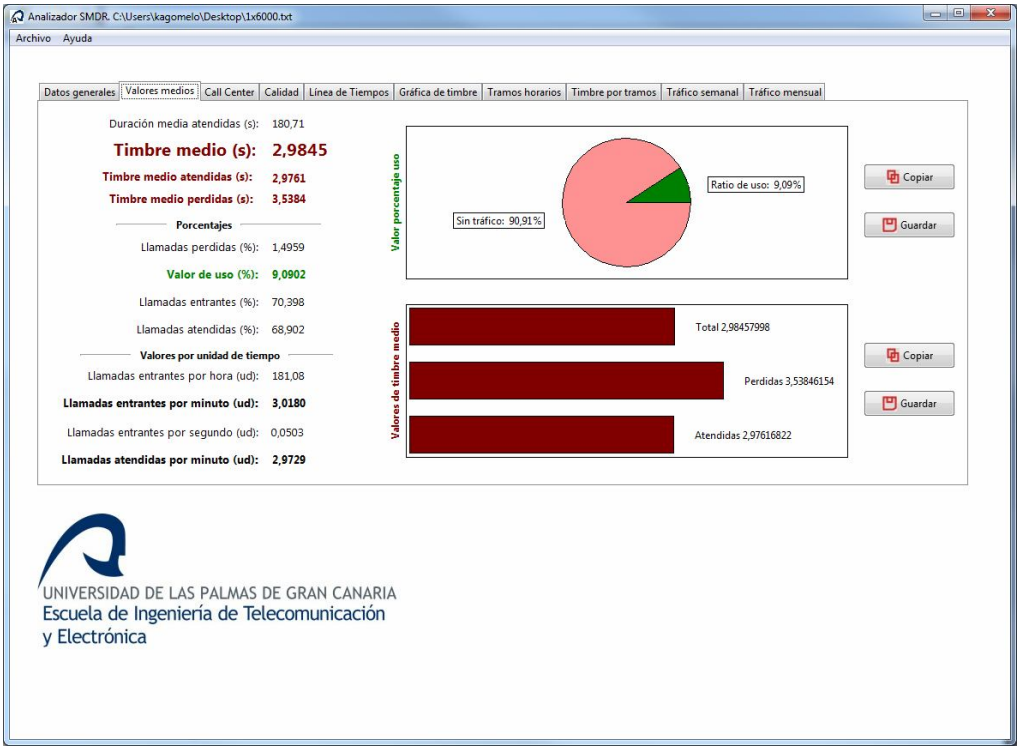


Figura 5.2.4. Valores medios.

La pestaña de valores medios nos ofrece una serie de parámetros calculados en base a los datos aportados por el fichero SMDR. Estos son:

- Duración media de las llamadas entrantes.
- Timbre medio general de todas las llamadas entrantes. Hay que recordar que timbre es el tiempo en segundos que tarda una llamada en ser atendida.

- Timbre medio de las llamadas atendidas, que es el tiempo medio que se ha tardado en responder a una llamada entrante no perdida. Este parámetro nos dará información clave en la mejora del sistema en el proceso de toma de decisiones que resulta de la evaluación de estos parámetros.
- Timbre medio de las llamadas perdidas, que viene a ser el tiempo que un usuario ha esperado en que se atienda su llamada hasta desistir. Este es otro parámetro esencial para evaluar mejoras en el sistema.
- Llamadas perdidas nos indica el porcentaje de llamadas de duración cero. También es un parámetro fundamental, especialmente en servicios de emergencias, pero también en cualquier empresa o institución.
- Valor de uso es el grado de amortización en términos de tiempo de nuestro equipamiento. Establece un ratio entre la duración de las llamadas y el tiempo total transcurrido entre el primer y último registro. Suele ser un valor bajo, pero da una perspectiva interesante a la hora de racionalizar recursos.
- El porcentaje de llamadas entrantes se calcula sobre el total de llamadas del fichero. Este dato nos da medida de cual es el perfil de nuestra centralita, esto es, más orientada a la atención a terceros o preferiblemente oprientada a tráfico interno y/o saliente.
- El porcentaje de llamadas atendidas nos indica el grado de éxito de las llamadas entrantes.
- Los valores por unidad de tiempo son indicadores nos dan una visión general del nivel de tráfico de nuestro sistema: Número de llamadas entrantes por hora, minuto y segundo, así como número de llamadas atendidas por minuto. Este último parámetro será necesario para calcular parámetros en Erlang.

A esta serie de valores analíticos les acompaña una gráfica del tipo tarta con la indicación del grado de uso, ya explicado anteriormente.

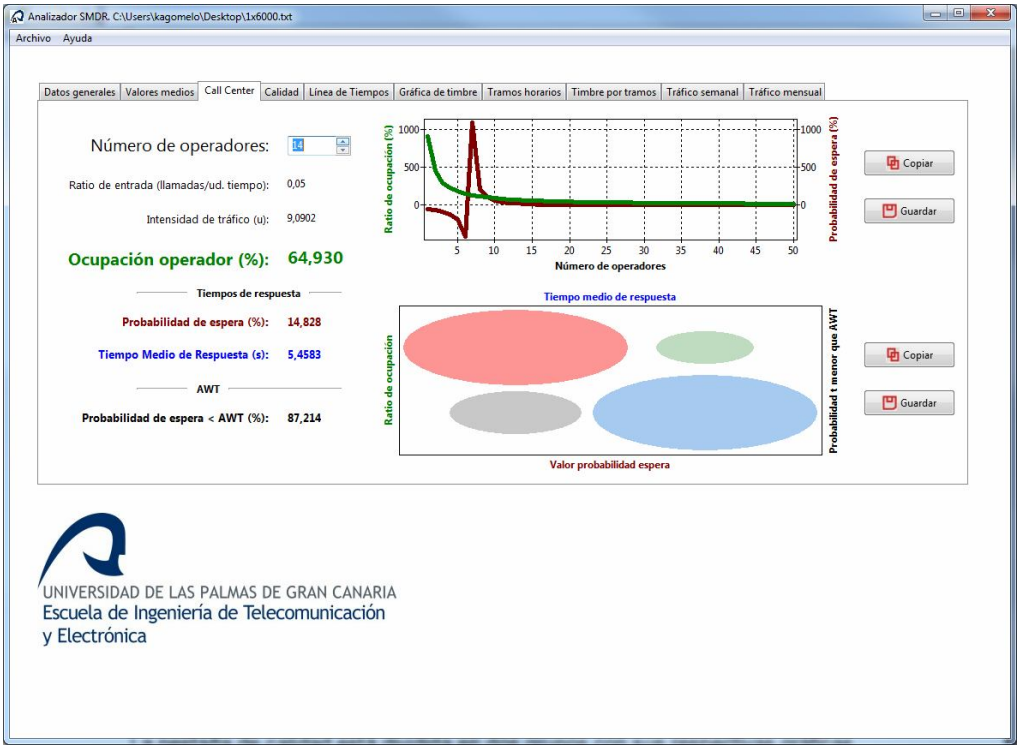


Figura 5.2.5. Parámetros de *call center*.

En esta pestaña disponemos del cálculo de parámetros que se ha denominado "*call center*". Este grupo de parámetros se calcula en base a cálculos directos y otros dependen del valor que vayamos asignando al número de operadores que deseamos asignar para que el sistema dé una respuesta adecuada al tráfico entrante. Así, obtenemos como valores fijos:

- El ratio de entrada (λ), que es el cálculo del tráfico entrante por unidad de tiempo y
- La intensidad de tráfico entrante, que es λ por la duración media en segundos de las llamadas.

Y como valores variables en función del número de operadores:

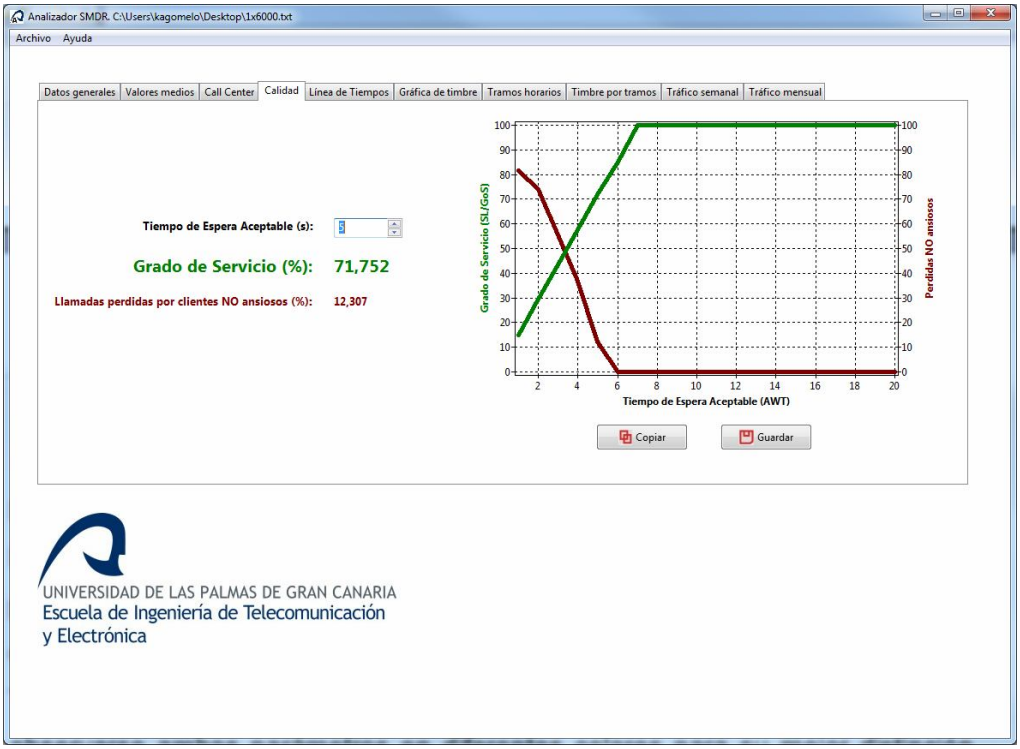
- El ratio de ocupación del operador en porcentaje, que nos indica el nivel de tiempo que el operador dedica a atender llamadas respecto al total de su jornada laboral.
- La probabilidad de espera para un usuario. Este parámetro genera valores inconsistentes para un número demasiado bajo de operadores cuando el nivel de tráfico es elevado. En la gráfica de la parte superior

derecha se plasma como variaciones negativas de probabilidad. Se calcula en base al modelo Erlang C.

- El tiempo medio de respuesta, cuyo acrónimo es ASA (*Average Speed of Answer*), que nos da otro parámetro de calidad del sistema. Este parámetro también genera valores inconsistentes para un número demasiado bajo de operadores cuando el nivel de tráfico es elevado. Para este PFC se han calculado los tiempos medios de respuesta de la Policía Municipal de Las Palmas de Gran Canaria, de la Policía Nacional y del servicio de emergencia 112, dando como valores 7 segundos para los dos primeros casos y menos de dos segundos para el 112.
- El último parámetro en calcular es la probabilidad de que las llamadas entrantes esperen un tiempo menor al AWT, calculado en primer lugar en esta pestaña de parámetros. Genera un valor en términos porcentuales. También hay que indicar que este parámetro también genera valores inconsistentes para un número demasiado bajo de operadores cuando el nivel de tráfico es elevado.

En la parte derecha acompañan a los datos dos gráficas. En la parte superior se enfrentan el número de operadores a su ratio de ocupación (en color verde) y a la probabilidad de espera.

En la parte inferior derecha se ha hecho una abstracción mediante una gráfica de las denominadas “de burbujas”. Se representan cuatro elipses en cuatro colores según el patrón de colores del texto de la izquierda: verde para el ratio de ocupación de operador, rojo oscuro para la probabilidad de espera, azul para el tiempo medio de respuesta y negro para el tiempo medio de respuesta menor que el AWT. Las elipses van cambiando de tamaño en función del número de operadores asignado, pero el paso a considerar es el del color inicial a un color más claro, tipo pastel. Cuando la elipse se aclara significa que entra en un margen de valores positivo en el diseño del *call center*. El caso óptimo es cuando las cuatro elipses adoptan un color claro.



La pestaña de calidad permite una simulación del Grado de Servicio y el ratio de llamadas perdidas por clientes no ansiosos. Estos dos parámetros se calculan en función del Tiempo de Espera Aceptable, conocido como AWT (*Aceptable Waiting Time*) en este contexto. En la gráfica que este grupo tiene a su derecha pueden observarse ambos parámetros en diferentes colores para su mejor distinción siguiendo el patrón de colores del texto a su izquierda. Puede utilizarse el botón izquierdo del ratón para hacer un zoom sobre una zona determinada, así como el botón derecho del ratón para arrastrar la vista actual. Para volver al gráfico inicial basta con hacer *click* sobre la gráfica con el botón izquierdo del ratón.

Figura 5.2.6. Línea de tiempos (detalle).

La pestaña Línea de tiempos muestra una gráfica de barras donde constan todas las llamadas entrantes. Este gráfico aporta información inmediata para ficheros con un número menor aproximado a las 500 llamadas, pero requiere del uso de las ayudas creadas para ficheros de gran tamaño. Entre estas ayudas cabe destacar las conocidas (por mencionadas) de zoom y arrastre con los botones del ratón, a las que se suman en la parte superior derecha un pequeño mapa que nos ubica una vez realizado el zoom. Este pequeño mapa contiene un recuadro que también se puede arrastrar para modificar la vista actual. Las barras de desplazamiento inferior y derecha permiten también desplazarnos por la imagen de forma cómoda.

En la parte derecha se encuentran varios botones, que a continuación describimos:

- “Copiar” nos almacena en la memoria del portapapeles la imagen que es este momento estamos realizando. Esta herramienta es especialmente práctica cuando se desea hacer un informe gráfico en el que se desse destacar una información determinada dentro del conjunto de llamadas, esto es, después de hacer un zoom.

- “Guardar” permite guardar en formato JPEG la imagen que en este momento se está observando en la gráfica.
- El botón “Todo” nos permite volver a la situación inicial después de haber hecho una selección de fechas.
- Los siguientes dos controles permiten hacer una búsqueda entre fechas. Indicando en el control bajo el término “Desde” una fecha de inicio y en el siguiente una fecha final, la nueva gráfica se mostrará en pantalla. Esta opción es especialmente valiosa para ficheros con gran cantidad de datos. Una vez seleccionadas las nuevas fechas, basta con pulsar en el botón “Redibujar” para obtener la nueva vista.

La leyenda indica la representación en la gráfica del valor medio de la duración de cada llamada (línea gruesa de color negro) y de la desviación estándar (que nos da el grado de dispersión de datos, esto es, de la duración de las llamadas).

Las barras de desplazamiento inferior y derecha también ayudan a una mejor selección de la vista deseada.

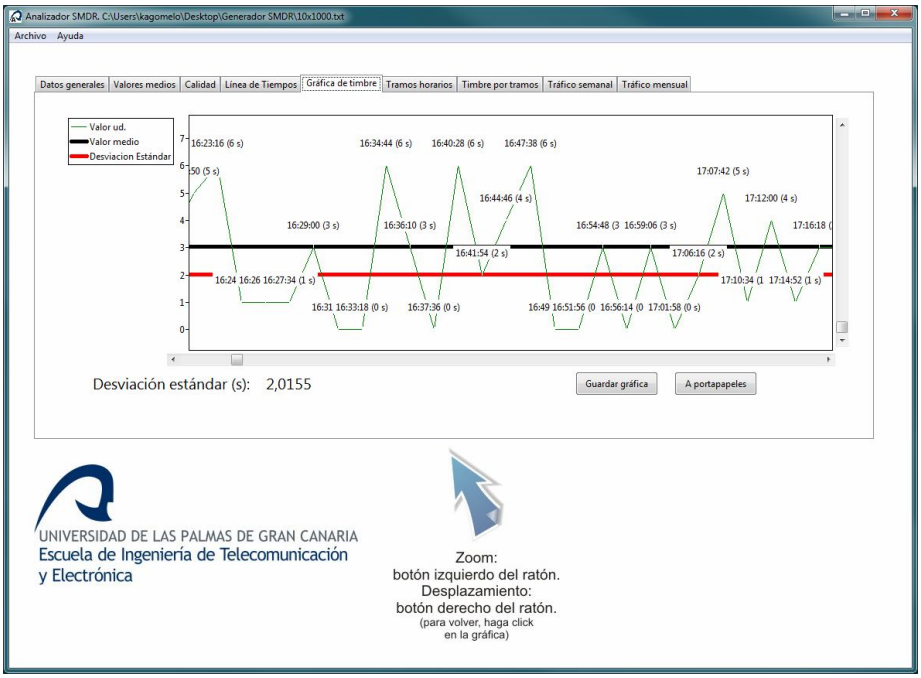


Figura 5.2.7. Gráfica de timbre.

La gráfica de timbre guarda cierto paralelismo con la de línea de tiempos, con la diferencia de que muestra el tiempo de timbre en la escala vertical. Con esta gráfica observamos los tiempos de espera de todas las llamadas entrantes, tanto las atendidas como las que no.

Adicionalmente se han agregado dos valores que atraviesan la gráfica. Por un lado es la media aritmética del timbre, mostrada en color negro. Por otro, el valor de la desviación estándar, en color rojo. Este parámetro permite valorar el grado de dispersión de los datos mostrados, lo cual puede ser de gran valor para encontrar tramos horarios o de fechas donde el tiempo en atender llamadas sea especialmente destacado.

Acompañan a la gráfica los botones y controles de fechas similares a los de la gráfica anterior.

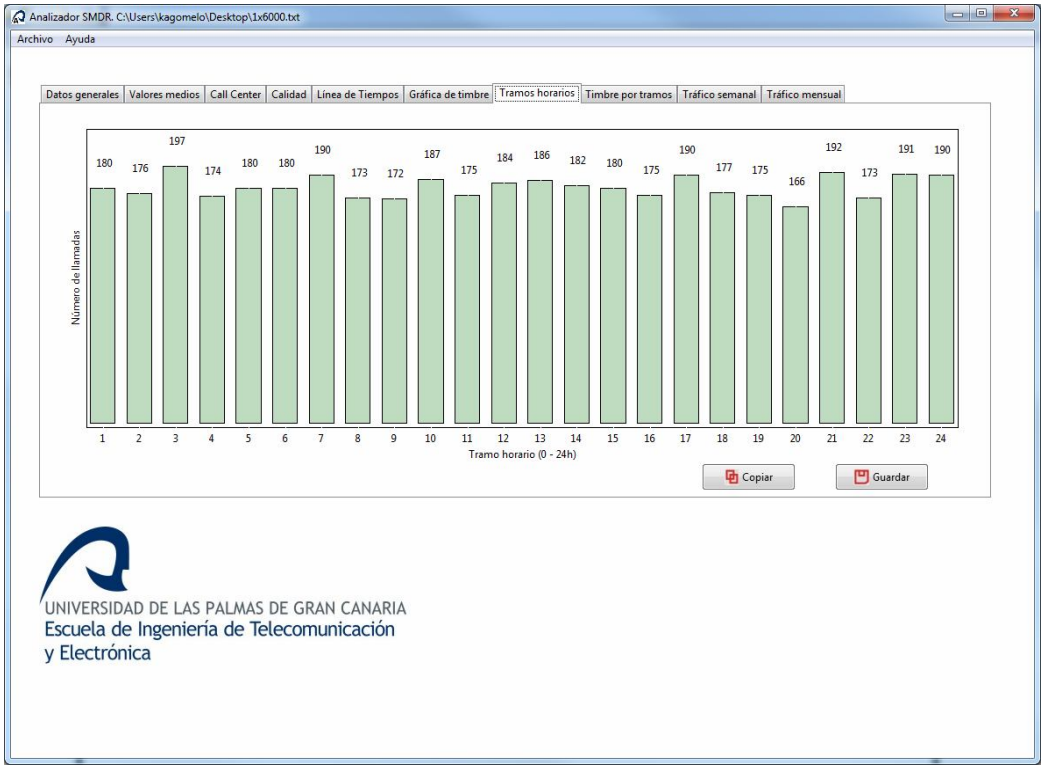


Figura 5.2.8. Gráfica de tráfico por tramos horarios.

La pestaña “Tramos horarios” da muestra del tráfico (en número de llamadas) dependiendo del tramo horario del que se trate. Así, puede observarse en esta figura que cada columna hace mención al tramo horario justo anterior: El “1”

para el tramo entre las 00:00 y las 00:59, el “2” para el tramos entre la 1:00 y la 1:59 y así en adelante. Esta gráfica permite identificar en qué tramos horarios se concentra el tráfico en nuestra centralita y donde deben redoblarse esfuerzos en atender esas llamadas.

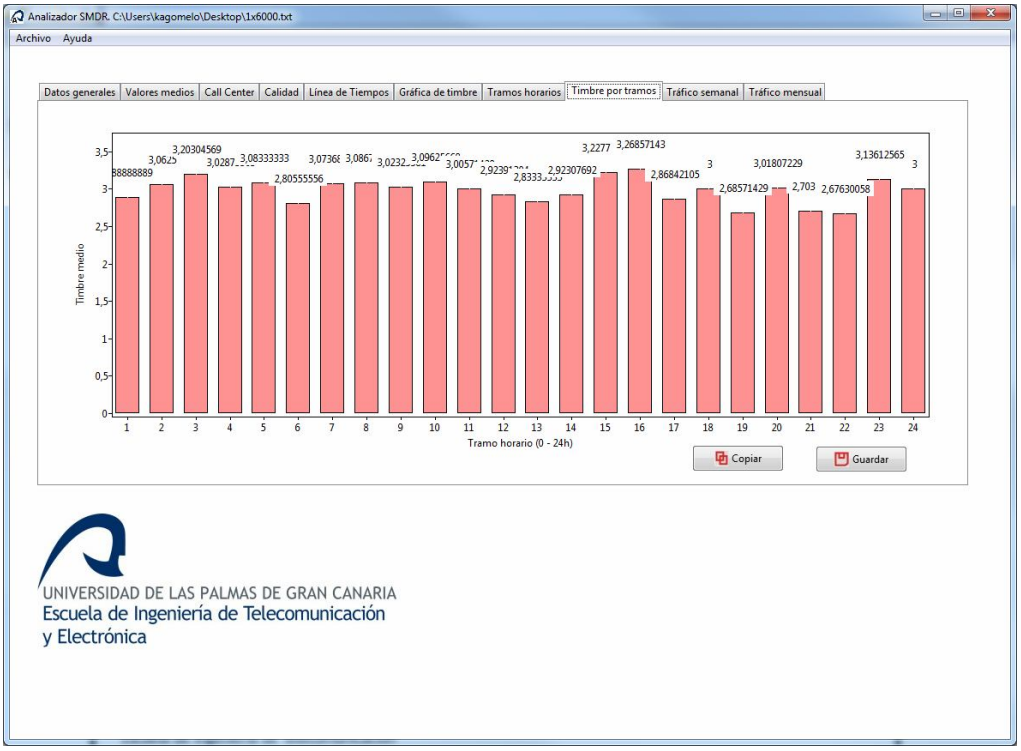


Figura 5.2.9. Gráfica de timbre por tramos horarios.

La gráfica de timbre por tramos utiliza el mismo criterio para identificar los tramos que el caso anterior, pero dando valor en vertical al parámetro del timbre.

Esta gráfica ofrece información sobre el tiempo de espera de las llamadas entrantes en función de la hora del día de que se trate, aspecto esencial ante una toma de decisiones vinculada a la mejora del sistema.

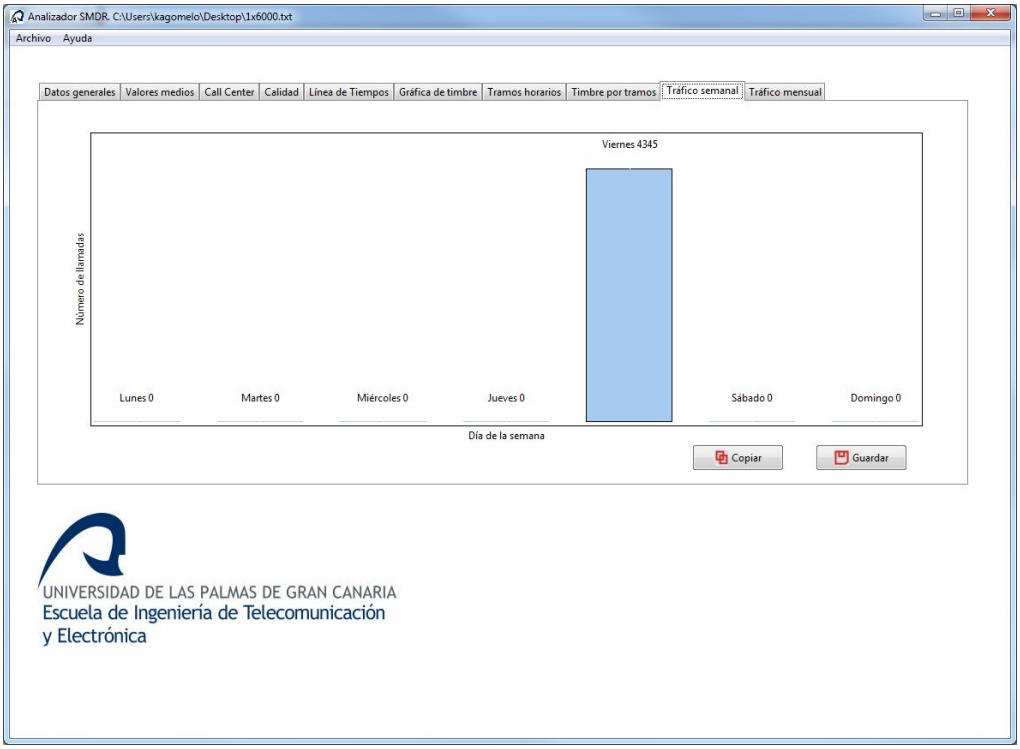


Figura 5.2.10. Tráfico semanal.

La pestaña “Tráfico semanal” ofrece información sobre la cantidad de llamadas que entran al sistema de telefonía en función del día de la semana de que se trate. Esta gráfica permite conocer los picos de tráfico en función del día de la semana de que se trate.

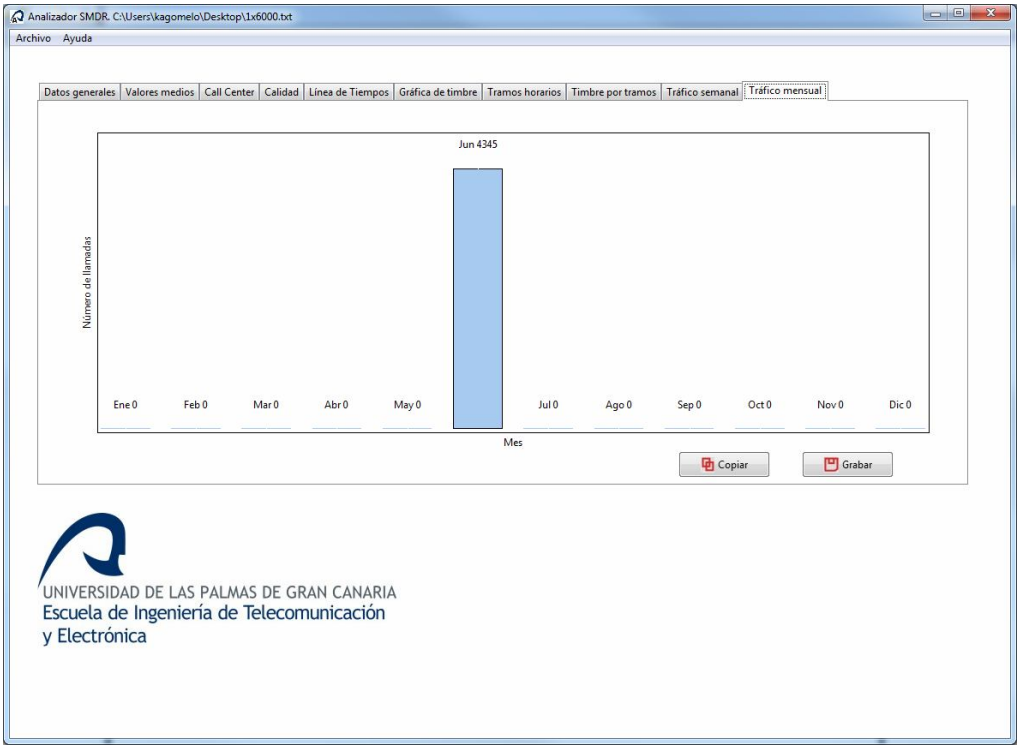


Figura 5.2.11. Tráfico mensual.

La gráfica de tráfico mensual permite observar el tráfico entrante en cada mes del año de que se trate. Da una visión de conjunto bastante adecuada para conocer los comportamientos del flujo de datos entrante a lo largo del año.
En los anexos que se adjuntan al presente documento pueden observarse el listado de código tanto del generador de ficheros SMDR pseudoaleatorios como el del analizador.

5.2.3. Pruebas. Obtención y auditado de datos.

En el capítulo de entradas/salidas de esta aplicación ya se ha reseñado todo lo referente a la entrega de datos visuales, así como las opciones de grabación en fichero JPEG o en el portapapeles.
De forma añadida, el analizador de ficheros SMDR tiene la opción de generar un reporte, un fichero de datos que resume todas las variables obtenidas de forma analítica y visibles con la operativa del mismo.
En la figura 5.2.12 pueden observarse los contenidos del informe.

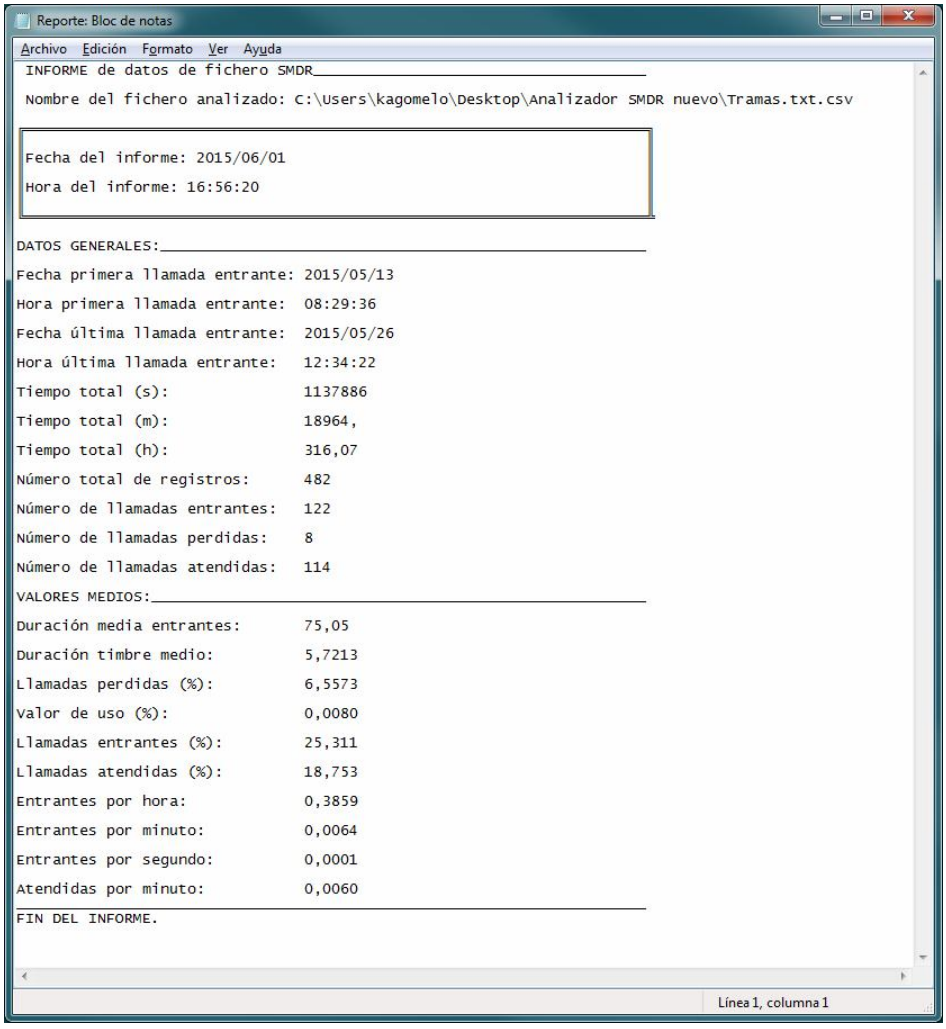


Figura 5.2.12. Informe de datos.

Por otro lado, el acceso a archivos se circunscribe a la lectura de ficheros SMDR siguiendo la nomenclatura de equipos Avaya (tipo C).

5.3. Herramienta de generación de ficheros SMDR pseudoaleatorios.

5.3.1. Análisis funcional.

Siguiendo el mismo proceso de análisis seguido para el desarrollo de la aplicación de análisis, la aplicación para generar ficheros SMDR construye un fichero separado por comas con datos variables de fecha, hora, duración, timbre y carácter de entrada o salida. El fichero toma como fecha de inicio el primer día de 2000 y a partir de esta fecha genera el fichero con un número de registros equivalente de forma aproximada al resultado de multiplicar el número de días por el número de llamadas por día que se introduzca. Se ha utilizado una variable aleatoria para limitar la duración de las llamadas por debajo de 9 minutos de duración y una media aproximada del 60% de llamadas entrantes.

5.3.2. Análisis orgánico.

De igual forma en cuanto a metodología se refiere, el análisis orgánico nos lleva a los siguientes datos. La aplicación solo requiere la entrada de dos parámetros para generar el fichero deseado. Estos ficheros cubren las necesidades de análisis de ficheros para hacer pruebas con la aplicación de análisis, especialmente cuando se requiere una cantidad importante de registros. Se puede tomar como referencia la media de llamadas del servicio de emergencias de Canarias, teléfono 112, que supera las 5.000 llamadas por día.
La pantalla de inicio (única) de este generador de ficheros es la siguiente:

Figura 5.3.1. Pantalla de inicio (única).

Al final de este documento, en anexo, se detalla el código de la aplicación para generar ficheros SMDR pseudo-aleatorios.

5.3.3. Pruebas. Obtención y auditado de datos.

El generador crea un fichero como el que se muestra en la figura 5.3.2.

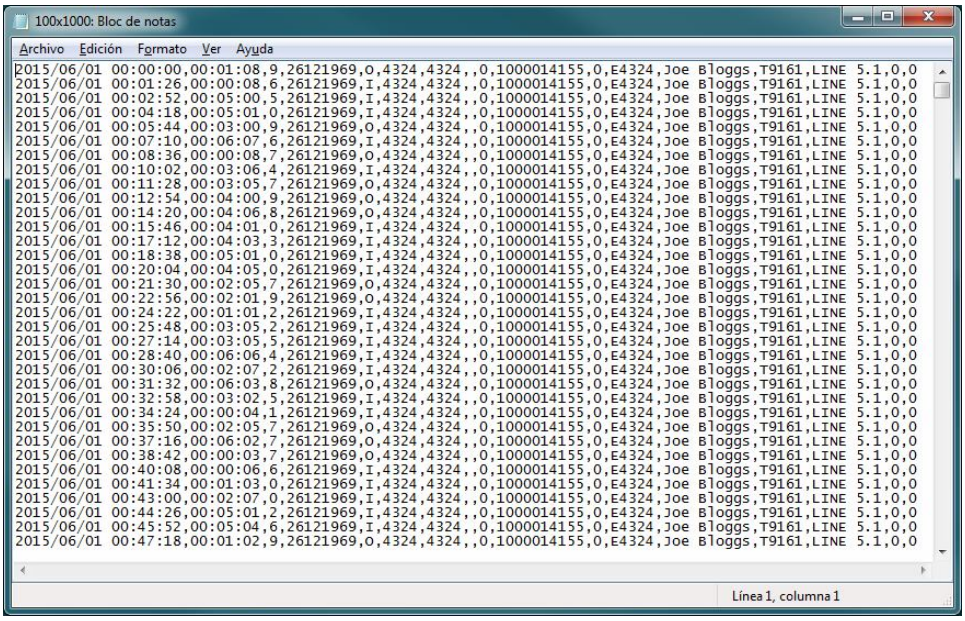


Figura 5.3.2. Fichero SMDR.

En este caso, se trata de un fichero con datos de 100 días a razón de 1000 llamadas por día (no se muestran todos sus registros). Cabe destacar que el tiempo de generación de este fichero ha sido de menos de un segundo, disponiendo de unos 100.000 registros.

Capítulo 6

Conclusiones y líneas futuras

6.1. Conclusiones.

En resumen, este proyecto ha implicado:

- Un amplio estudio documental de los fundamentos matemáticos, así como de la situación donde es potencialmente aplicable el presente desarrollo *software*.
- El desarrollo *software* de una aplicación que permite evaluar tráfico entrante en una centralita IP Avaya con el objeto de adoptar decisiones que permitan aminorar los tiempos de respuesta en el ámbito del que se trate. Se da la circunstancia de que tiene un enfoque claro a ámbitos donde el tiempo de respuesta es crítico, tales como servicios de emergencia o *call centers*.

Adicionalmente hay que reseñar que este proyecto ha requerido trabajar con documentación relativa a equipos de voz, vídeo e Internet como herramientas de trabajo para estudiar productos y servicios que las empresas necesitan, así como saber qué otros productos hay en el mercado relacionados.

Para el desarrollo de las herramientas de *software* realizadas, ha sido necesario un uso extensivo de herramientas de desarrollo *software* y equipos informáticos *hardware*. El conocimiento de la herramienta *Lazarus*, así como el trabajo de análisis han permitido un código compacto a la par que operativo, con un resultado de gran fluidez.

Este proyecto ha significado un importante esfuerzo documental para apuntalar los desarrollos expuestos, buscando fórmulas innovadoras para resolver problemas reales del sector de las telecomunicaciones. Esto ha significado una mejora en la capacidad de organización documental y gestión de tiempos, habida cuenta de las muchas tareas que se han debido realizar, complementadas con la labor de seguimiento de los tutores.

Este proyecto ha reforzado ampliamente habilidades relacionadas con la preparación de documentos técnicos complejos, abriendo también el camino a una mejora en la capacidad de resolver problemas de alto nivel técnico, lo que ha implicado un trabajo metódico y científico.

Finalmente hay que decir que este proyecto ha significado una mejora en la capacidad de automotivación de quien lo suscribe, fortaleciendo la capacidad de aprender a aprender, ampliando la capacidad para fijar el aprendizaje mediante la reflexión y el pensamiento razonado en el ámbito de las tecnologías de las telecomunicaciones.

El desarrollo de este proyecto ha conllevado, además, los siguientes logros:

- Comprender los fundamentos de la tecnología VoIP.
- Identificar los elementos básicos en el desarrollo de aplicaciones prácticas relativas a la tecnologóa VoIP.
- Interpretar el estado del arte, incluyendo el estudio de las centralitas Avaya y equipos relacionados.
- Desmenuzar el protocolo SMDR, las particularidades de su formato y los resultados que se derivan de su estudio.
- Implementar *software* que evalúe tráfico entrante en centralitas que utilicen el estándar SMDR y que permita identificar problemáticas existentes o potenciales.
- Priorizar parámetros críticos en el análisis de tráfico entrante en centralitas que utilizan tecnología IP que permitan interpretar datos de forma rápida e intuitiva.
- Valorar potenciales amenazas en entornos laborales donde el tráfico telefónico entrante es esencial.
- Inventariar productos comerciales existentes con los que propone el presente proyecto.
- Exponer posibles desarrollos y mejoras sobre el presente proyecto.

6.3. Líneas futuras.

Este PFC deja un amplio margen para desarrollos futuros, siempre orientados a un análisis de tráfico entrante en centralitas VoIP, ya que las aplicaciones comerciales existentes se centran mucho más en aspectos relacionados con el tráfico saliente y el análisis de costes derivado del mismo.

Por un lado, puede ser objeto de un Proyecto Fin de Carrera el diseño de un interfaz que conecte el puerto serial RS-232 de que disponen las centralitas

Avaya, que se puede conectar a un smartphone a través de su conector USB universal, que bien puede disponer de una aplicación basada en *Android* para el estudio del tráfico.

Esta aplicación permitiría analizar datos en tiempo real del tráfico existente en una centralita, si bien su verdadera utilidad práctica estaría en el análisis de datos contenido en un fichero de datos SMDR generado por la propia centralita.

Otro aspecto potencialmente interesante podría ser el desarrollo de una aplicación soportada en Internet y que pudiera acceder de forma remota a una centralita, obtener datos SMDR y evaluarlos en los mismos términos que realiza la presente aplicación.

El equipo de desarrolladores de *Lazarus* ha permitido que cualquier desarrollo realizado con este entorno sea transportable a otras plataformas y sistemas operativos con pocas variaciones en el código, lo cual permite abrir una puerta adicional a la adaptación de este aplicativo a otros entornos y sistemas operativos para un mayor y más amplio espectro de uso. Sin embargo, es en el escenario de los *call centers* donde puede adquirir una mayor relevancia cualquier desarrollo. La sinergia entre lo humano y lo tecnológico adquiere una dimensión extraordinariamente interesante en el campo de la ingeniería de telecomunicaciones. Esto implica un campo abierto al desarrollo de aplicativos orientados a ayudar al proceso de selección de personal con el objeto de confeccionar las matrices de habilidades/operador, así como el resto de herramientas necesarias para el adecuado dimensionamiento de un centro de atención de llamadas.

Un sistema experto que ayude en el proceso de toma de decisiones que considere todos los datos de tráfico podría convertirse en una excelente herramienta para todo gestor o diseñador de centros de llamadas.

Capítulo 7

Bibliografía,

Pliego de condiciones y

Presupuesto.

7.1. Bibliografía.

[Landivar 2009] Landivar, E., *Comunicaciones unificadas con Elastix, Volumen I.* GNU Free Documentation License, 2009.

[Silicon 2013] http://www.siliconnews.es/2013/02/15/un-tercio-de-las-llamadas-a-nivel-mundial-se-hicieron-por-skype-en-2012/

[Davidson 2001] Davidson, J., Peters, J., *Fundamentos de voz sobre IP*, Cisco Press, 2001.

[Huidobro 2006] Huidobro, J., Roldán, D. *Comunicaciones en redes Wifi, VoIP, Multimedia y Seguridad*, Creaciones Copyright, SL., 2006.

[Verma 2011] Verma, P., Wang, L., *Voice over IP Networks*, Springer, 2011.

[Manual Avaya 2010] Avaya, *IP Office Release 6.1*, 2010.
http://support.avaya.com/css/appmanager/public/support

[Avaya IP Office 8.1] *Guía de descripción general de IP Office Server Edition Solution IP Office 8.1*.

[Chee 2008] NG Chee Hock, *Queueing modelling fundamentals*, Willis.

[Wiki Lazarus 2015]
http://wiki.lazarus.freepascal.org/Overview_of_Free_Pascal_and_Lazarus/es

[Menkaura 2015] Menkaura Abiolla-Ellison, *Getting Started with Lazarus and Free Pascal*, Mka Publishing, 2015.

[Novasim 2015] NovaSim. ccProphet - *simulate your call center´s performance. Available online via* <www.novasim.com/CCProphet/>.

7.2. Pliego de condiciones.

Para la realización de este proyecto se ha hecho uso de un conjunto de herramientas *software* y equipos *hardware* cuyas características y versiones se detallan a continuación:

- Equipo informático.
    - Ordenador de sobremesa con microprocesador Intel® S3 marca Acer modelo Aspire X3950 con 4GB de RAM y 300GB de disco duro.
    - Ordenador portátil Acer Aspire S5 con Microsoft *Windows*® 7 *Home Premium*, procesador Intel Core i5-3317U Dual-core 1,70 GHz, 4 GB, DDR3 SDRAM y 128 GB de disco duro de estado sólido.
    - Dos monitores Philips de 23 pulgadas.
    - Impresora Brother DCP-7065DN.
    - Impresora Brother HL-L8250CDN.

Para el desarrollo de este proyecto se han utilizado las siguientes herramientas *software*:

- *Microsoft Word* del paquete *Office* 2003 utilizado para la elaboración de la memoria.
- *Microsoft PowerPoint* del paquete *Office* 2003, utilizado para la elaboración de la presentación del proyecto.
- *Microsoft Windows* 7, sistema operativo bajo el que se ha hecho el proyecto.

7.3. Presupuesto.

Juan Francisco Rodríguez Báez, autor del presente Proyecto Fin de Carrera, tutorizado por D. Carlos M. Ramírez Casañas y D. Adán San Gil Martín, por medio de la presente

Declara

Que el Proyecto Fin de Carrera con título “Desarrollo *software* para control de tráfico telefónico por medio del protocolo SMDR”, desarrollado en la Escuela de Ingeniería de Telecomunicación y Electrónica (EITE), de la Universidad de Las Palmas de Gran Canaria, en el periodo de seis meses, tiene un coste total de desarrollo de 21.168,97 € (veintiunmil ciento sesenta y ocho con noventa y siete céntimos), correspondiente a la suma de los importes que a continuación se detallan.

Las Palmas de Gran Canaria.
Junio 2015.

Firma: _____

7.3.1. Desglose.

Para la realización del presupuesto se han seguido las recomendaciones del Colegio Oficial de Ingenieros Técnicos de Telecomunicación (COITT) sobre los baremos orientativos mínimos para trabajos profesionales en 2014. El presupuesto se ha desglosado en varias secciones en las que se han indicado los distintos costes asociados al desarrollo del proyecto. En otro apartado se detallan los costes de una centralita Avaya similar a la existente en los laboratorios de la EITE.

El tipo de costes de este proyecto son recursos materiales (Equipos y *software*), horas de trabajo, costes de redacción del proyecto, material fungible, derechos de visado del colegio, impuestos y coste de centralita.

7.3.1.1. Recursos materiales.

Equipos.

A continuación se detalla el equipamiento informático utilizado para dar soporte a las herramientas de *software* utilizadas. Se ha estipulado un coste de amortización lineal para equipamiento informático de 4 años a razón de:

$$C = \frac{V_c - V_f}{A_u}$$

Donde C es la cuantía por año, V_c el valor de compra, V_f el valor al final de la vida útil y A_u los años de vida útil.

De forma detallada, el equipamiento utilizado ha sido:

Hardware	**Precio de mercado**	**Valor a 4 años**	**Amortizado 1º año**
Ordenador de sobremesa con microprocesador Intel® S3 marca Acer modelo Aspire X3950 con 4GB de RAM y 300GB de disco duro.	400,00 €	100,00 €	75,00 €
Ordenador portátil Acer Aspire S5 con Microsoft Windows® 7 Home Premium, procesador Intel Core i5-3317U Dual-core 1,70 GHz, 4 GB, DDR3 SDRAM y 128 GB de disco duro de estado sólido.	1.200,00 €	400,00 €	200,00 €
Dos monitores Philips de 23 pulgadas.	300,00 €	100,00 €	50,00 €
Impresora Brother DCP-7065DN.	150,00 €	50,00 €	25,00 €
Impresora Brother HL-L8250CDN.	300,00 €	100,00 €	50,00 €
		Total	**400,00 €**

Hardware	**Precio de mercado**	**Valor a 10 años**	**Amortizado 1º año**
Centralita Avaya IP Office 500 con 5 teléfonos digitales/IP	1.300,00 €	600,00 €	150,00 €
		Total	**150,00 €**

Software.

Entre los recursos materiales utilizados para la realización de este proyecto, se incluyen la herramienta *software* de desarrollo *Lazarus*, los paquetes de ofimática utilizados para la redacción de la memoria, y el sistema operativo bajo el que se ejecutó el trabajo:

- *Lazarus* IDE.

- *Microsoft Office*® 2003.
- *Microsoft Windows*® 7.

Software	**Precio mercado**	**Valor a 1 año**
Lazarus IDE	0	0
Microsoft Office® 2003	150,00 €	50,00 €
Microsoft Windows® 7	300,00 €	100,00 €
	Total	**150,00 €**

7.3.1.2. Horas de trabajo.

Se han invertido 6 meses en todas las tareas relacionadas con el presente proyecto: documentación, diseño e implementación, así como las pruebas necesarias para comprobar el adecuado funcionamiento de los diferentes elementos *software*.

Para el cálculo de los costes por horas de trabajo se ha tenido en cuenta que el Ministerio de Economía y Hacienda remitió a todos los colegios profesionales una nota en la que se recordaba que, siguiendo directivas europeas, se debían eliminar los baremos orientativos de honorarios que tradicionalmente publicaban los colegios profesionales. Por ello, y haciendo notar que los honorarios son libres y responden al libre acuerdo entre el profesional y su cliente, el COITT sugiere que se tengan en cuenta algunos conceptos importantes:

- Costes directos del ingeniero y de sus colaboradores. Se tiene que considerar el coste de las horas de todos lo integrantes del equipo de trabajo y las empleadas en trabajos técnicos, reuniones con el promotor, instalador, arquitecto, constructor, organismo públicos y en el caso de la dirección de obra las visitas a realizar.
- Viajes, dietas, hoteles, delineación, mecanografía, reproducción y encuadernación, etc. Imputables al trabajo encomendado.
- Porcentaje del total año de gastos generales que se repercuten a cada trabajo concreto como los derivados de impuestos del trabajo personal, alquiler de local, amortización de equipos, seguridad social, intereses de préstamos, etc.

- Derechos de visado y tasas administrativas si procede (Ministerio, Jefatura, Ayuntamiento, etc.).
- El número de horas a emplear en cada trabajo dependerá de la experiencia del ingeniero y de las herramientas y bases de datos que disponga.
- Otro factor a tener en cuenta es el volumen de actividad con cada cliente. Lo indicado en los puntos anteriores supone únicamente una orientación al ingeniero para el cálculo de sus honorarios, sin que de ninguna manera pueda presuponerse su carácter oficial.

Siguiendo por tanto las recomendaciones del COITT y estableciendo como referencia valores sugeridos en años anteriores, los honorarios en función de las horas empleadas en la realización del proyecto se calculan a razón de:

$$C = f \times (58 \times H_l)$$

Donde C es la cuantía, f un factor de corrección y H_l el número de horas trabajadas en horario laboral.
Para la realización de este proyecto han sido necesarias 720 horas a razón de 6 horas diarias durante 5 días a la semana en un plazo de 6 meses.
El Factor de corrección tiene un valor determinado en función del número de horas empleadas. Teniendo en cuenta que el número de horas ha sido de 720, el factor tiene un valor de 0,45. Así:

$$C = 0{,}45 \times (58 \times 720) = 18.792{,}00€$$

7.3.1.3. Costes de redacción del proyecto.

Se ha estimado que, dada la cuantía anterior, el importe por redacción del proyecto es irrelevante.

7.3.1.4. Material fungible.

Los gastos en materiales fungibles necesarios en la realización del proyecto han sido los siguientes:

- Papel: 10,00 €.
- Tóner de impresora láser: 300,00 €.
- Material y gastos de encuadernación: 30,00 €.
- Otro material de papelería: 10,00 €.

El coste total en fungibles es de 350,00 €.

7.3.1.5. Derechos de visado.

Los gastos de visado (V) del colegio se tarifican según la siguiente fórmula:

$$V = 0{,}007 \times P \times f'$$

Donde P es el presupuesto y f' el factor de corrección.
Para presupuestos menores de 20.000 €, el COITT estima:

$$f' = 0{,}7$$

Quedando:

Visado = 0,007 x 18.792,00 € x 0,7 = 92,08 €

7.3.1.6. Impuestos.

Siguiendo los cálculos anteriores, las cuantías son:

Coste del proyecto = 18.792,00 € + 550,00€ + 350,00 € + 92,08 € = 19.784,08 €

El Impuesto General Indirecto Canario de aplicación en este caso es del 7%, con lo que el presupuesto final del proyecto es:

Coste total del proyecto = 19.784,08 € + 0,7% (IGIC) = 21.168,97 €

El presupuesto total del proyecto asciende a la cantidad de veintiunmil ciento sesenta y ocho con noventa y siete céntimos.

Anexos

Anexo 1. Erlang.

El Erlang es una unidad adimensional que da medida estadística del volumen de tráfico. El tráfico de un Erlang corresponde a un recurso (circuito, canal, etc.) utilizado de forma continua, o dos recursos utilizados al 50%, y así sucesivamente. De forma alternativa, un Erlang puede ser considerado como "multiplicador de utilización" por unidad de tiempo, así un uso del 100% corresponde a 1 Erlang, una utilización de 200% son 2 Erlangs, y así sucesivamente. En general, si la tasa de llamadas entrantes es de λ por unidad de tiempo y la duración media de una llamada es *h*, entonces el tráfico *A* en Erlangs es:

$$A = \lambda \times h$$

Este parámetro nos permitirá determinar si un sistema está sobredimensionado o dispone de muy pocos recursos asignados. De esta forma, podremos calcular cuántas líneas (troncales) deberán utilizarse durante las franjas horarias de mayor ocupación.

El tráfico medido en Erlangs se utiliza para calcular el nivel (o grado) de servicio (GoS o SL). Hay diferentes fórmulas para calcular el tráfico, entre ellas Erlang B, Erlang C y la fórmula de Engset. Esto será expuesto a continuación, y cada uno puede ser derivado como un caso especial de Procesos de tiempo continuo de Markov conocido como *birth-death process*.

Erlang B.

Erlang-B, también conocida como la fórmula de pérdida de *Erlang*, deriva de la probabilidad de bloqueo de la distribución de *Erlang* para describir la probabilidad de pérdida de llamada en un grupo de circuitos (en una red de circuitos conmutados, o equivalente). Este parámetro se usa en la planificación de las redes telefónicas aunque la fórmula no se limita a estas redes, ya que describe una probabilidad en un sistema de colas (aunque se trata de un caso

especial con un número de servidores, pero sin espacios de búfer para las llamadas entrantes que esperan a que un servidor quede libre). La fórmula se aplica partiendo del supuesto de que una llamada sin éxito, debido a que la línea está ocupada, no se pone en cola o se vuelve a intentar, sino que se pierde sin solución de continuidad. Se entiende que los intentos de llamada llegan conforme a un proceso de *Poisson*, por lo que las llegadas de cada llamada son independientes. Además, se supone que las longitudes de los mensajes están exponencialmente distribuidas (correspondiéndose con un sistema *Markoviano*). A pesar de esto, se puede aplicar en otras distribuciones de tiempo.

El *Erlang* es una cantidad adimensional que se calcula como la tasa promedio de llegada, λ, multiplicada por la longitud media de la llamada, *h*. La fórmula de *Erlang-B* asume una población infinita de fuentes (por ejemplo, abonados de teléfonos), que ofrecen conjuntamente el tráfico a *N* servidores (tales como enlaces en una ruta). La tasa de la llegada de nuevas llamadas es igual a λ y es constante, no en función del número de fuentes, debido a que el número total de fuentes se asume que es infinito. La tasa de salida de la llamada es el número de llamadas en curso dividido por *h*, la llamada de duración de tiempo media. La fórmula calcula la probabilidad de bloqueo en un sistema de pérdida, donde si una solicitud no es atendida inmediatamente cuando intenta utilizar un recurso, se anula[44]. En las solicitudes, por tanto, no espera. El bloqueo se produce cuando hay una nueva solicitud de una fuente, pero todos los servidores ya están ocupados. La fórmula asume que el tráfico bloqueado inmediatamente está desactivado.

La fórmula proporciona el GoS (grado de servicio) que es la probabilidad *Pb* de que una nueva llamada que llega sea rechazada debido a que todos los servidores (circuitos u operadores) están ocupados:

[44] Escenario inaceptable en algunos casos, como puede ser el de un servicio de emergencias.

$$P_b = B(A,m) = \frac{\frac{A^m}{m!}}{\sum_{i=0}^{m} \frac{A^i}{i!}}$$

donde:

- P_b es la Probabilidad de bloqueo.
- *m* es el número de recursos tales como servidores o circuitos en un grupo.
- *A* = λ*h* es la cantidad total de tráfico ofrecido en erlangs.

Esto puede ser expresado recursivamente como sigue, en un formulario que se utiliza para simplificar el cálculo de tablas de la fórmula de *Erlang-B*:

$$B(A,0) = 1$$

$$B(A,\ j) = \frac{AB(A,\ j-1)}{AB(A,\ j-1) + j} \forall j = 1,2...,m\,.$$

Por lo general, en lugar de *B*(*A*, *m*), la inversa 1/*B*(*A*, *m*) se calcula en a fin de garantizar la estabilidad:

$$\frac{1}{B(A,0)} = 1$$

$$\frac{1}{B(A,\ j)} = 1 + \frac{j}{A}\frac{1}{B(A,\ j-1)} \forall j = 1,2,...,m$$

La fórmula de *Erlang-B* se aplica a sistemas de pérdida, tales como los sistemas de telefonía en redes fijas y móviles, que no proporcionan almacenamiento en búfer de tráfico. Se supone que las llamadas entrantes pueden ser modeladas por un proceso de *Poisson*, pero también se acepta como válida cualquier distribución estadística de llamada con un tiempo medio de duración finita. *Erlang-B* es una herramienta de dimensionado de rutas de conmutación de circuitos para tráfico de voz y su fórmula es decreciente convergente en *m*.

Erlang B extendido.

Erlang-B extendido se utiliza cuando las llamadas que encuentran a los servidores ocupados no se pierden, si no que se reintentan. Es un cálculo iterativo, en lugar de una fórmula, que agrega un parámetro adicional, el factor de repetición, que define la proporción de rellamadas. Los pasos a seguir en el proceso de cálculo son las siguientes. Calcular:

$$P_b = B(A, m)$$

como se indica arriba para Erlang B. Calcular el número probable de llamadas bloqueadas:

$$B_e = A \times P_b$$

Calcular el número de rellamadas, R asumiendo un Factor de Repetición, R_f:

$$R = B_e \times R_f$$

Calcular el nuevo tráfico ofrecido:

$$A_{i+1} = A_0 + R$$

Donde A_o es el nivel inicial de tráfico. Hecho esto debe volverse al primer paso y repetir hasta que se obtenga un valor estable de A.

Erlang C.

La Formula de Erlang C también asume una infinita población de fuentes, las cuales ofrecen en conjunto, un trafico de *A* Erlangs hacia *N* servidores. Sin embargo, si todos los servidores están ocupados cuando una petición llega de una fuente, la petición es introducida en la cola. Un sin fin de números de peticiones podrían ir a la cola en este modo simultáneamente. Esta fórmula calcula la probabilidad de la cola ofrecido en el tráfico, asumiendo que las llamadas que fueron bloqueadas se quedaran en el sistema hasta que se puedan atender. Esta formula es usada para determinar la cantidad de agentes o representantes de clientes, que necesitará en un *Call Center* para conocer la probabilidad de permanecer en la cola.

$$P_W = \frac{\frac{A^N}{N!}\frac{N}{N-A}}{\sum_{i=0}^{N-1}\frac{A^i}{i!} + \frac{A^N}{N!}\frac{N}{N-A}}$$

Donde:

- *A* es la intensidad total del trafico ofrecido en unidades de Erlangs.
- *N* es la cantidad de servidores [número de troncales].
- P_W es la probabilidad de que un cliente tenga que esperar para ser atendido.

Se asume que las llamadas entrantes pueden ser modeladas usando una distribución de Poisson y que el tiempo de espera de las llamadas son descritas por una distribución exponencial negativa.

Engset.

La fórmula Engset, está relacionada con las anteriores, pero se utiliza con una población finita de *S* orígenes en lugar de la población infinita de orígenes que asume *Erlang*.

Una empresa que instale una centralita necesita saber el número mínimo de circuitos de voz que es preciso contratar hacia y desde la red telefónica. Un enfoque aproximado es utilizar la fórmula de Erlang-B. Sin embargo, si la empresa tiene un pequeño número de extensiones, debe en su lugar utilizar un cálculo más exacto, proporcionado por la fórmula de Engset, que refleja el hecho de que las extensiones que ya está en uso no hará llamadas simultáneas adicionales. Lógicamente, para una población de usuario grande, el cálculo por Engset y por Erlang B dará el mismo resultado:

$$E(N, A, S) = \frac{A^N \binom{S}{N}}{\sum_{i=0}^{N} A^i \binom{S}{i}}$$

Esto puede ser expresado recursivamente del siguiente modo, en una forma que es usada para calcular las tablas de la fórmula Engset:

$$E(0, A, S) = 1$$

$$E(N, A, S) = \frac{A(S - N + 1)E(N - 1, A, S)}{N + A(S - N + 1)E(N - 1, A, S)}$$

donde:

- *E* es la probabilidad de bloqueo.
- *A* es el tráfico en Erlangs generado por cada origen cuando está desocupado.
- *S* es el número de orígenes.
- *N* es el número de servidores.

De nuevo, se asume que las llamadas que llegan pueden ser modeladas por una distribución Poisson. Sin embargo, debido a que hay un número finito de servidores, la tasa de llegada de las nuevas llamadas decrece a medida que nuevos orígenes (como abonados telefónicos) pasan a estar ocupados y, por lo tanto, no pueden originar nuevas llamadas. Cuando *N* = *S*, la fórmula se reduce a una distribución binomial.

Anexo 2. Cisco CDR.

Cisco *Unified Communications Manager* produce dos tipos de registros de historial de llamadas, que registra la información de diagnóstico, de la siguiente manera:

- Registros de detalles de llamadas (CDR), que son los registros de datos que contienen información sobre cada llamada que fue procesada por *CallManager*.
- Gestión de llamadas (CMR), que son los registros de datos que contienen datos sobre la calidad de servicio (QoS) o información de diagnóstico sobre la llamada. También se conoce como registros de diagnóstico.

Ambos, CDRs y CMR juntos, se conocen como datos de CDR. Datos CDR proporciona un registro de todas las llamadas que se han realizado o recibido por los usuarios del sistema CallManager. Datos CDR es útil principalmente para generar registros de facturación; sin embargo, también puede ser utilizado para seguimiento de la actividad de llamadas, el diagnóstico de ciertos tipos de problemas y la planificación de capacidad.

CDR contienen información acerca del origen de las llamadas, destino de la llamada, la fecha y hora en que se inició la llamada, el tiempo de la conexión y otros datos. Una llamada se considera iniciada cuando la persona que llama descuelga el teléfono. La llamada se considera terminada cuando ya sea la persona que llama o la parte llamada ha colgado. CMR contienen información acerca de la cantidad de datos enviados y recibidos, *jitter*, latencia y pérdida de paquetes.

Registro de llamadas de duración nula (ZDF).

Un parámetro de servicio *CallManager* llamado 'CDR: Iniciar llamadas con indicador de duración cero' determina si se generan CDRs para llamadas que nunca fueron realizadas o duraron menos de un segundo. Por ejemplo, cuando un usuario descuelga y luego cuelga el teléfono sin completar una llamada.

CallManager siempre genera datos de CDR para las llamadas que tienen una duración de 0 segundos y terminan debido a algún tipo de error, independientemente de la configuración de ZDF en los CDR. Las llamadas a destinos ocupados o números de teléfono erróneos son ejemplos de este tipo de llamada. La duración de estas llamadas es 0 porque nunca se llegan a realizar, pero sus CDRs se generan de todos modos. Cuando se habilita la generación de CDR, *CallManager* genera un CDR para cada llamada de 1 segundo de duración o más.

Parámetros de empresa. Intervalo de tiempo.

Este parámetro especifica el intervalo de tiempo para la recolección de datos de CDR. Por ejemplo, si este valor se establece en 1, cada archivo contendrá de 1 minuto de CDR de datos (CDR y CMR, si está activado). La base de datos de CDR no recibirá los datos de cada archivo hasta que haya transcurrido el intervalo, por lo que se debe considerar la rapidez con que desea el acceso a los datos CDR cuando se decide qué intervalo configurar para este parámetro. El valor mínimo es 1 (que es la opción por defecto). El máximo es de 1440. La unidad de medida para este campo obligatorio representa un minuto. En la figura A.2.1 puede observarse la arquitectura de CDR.

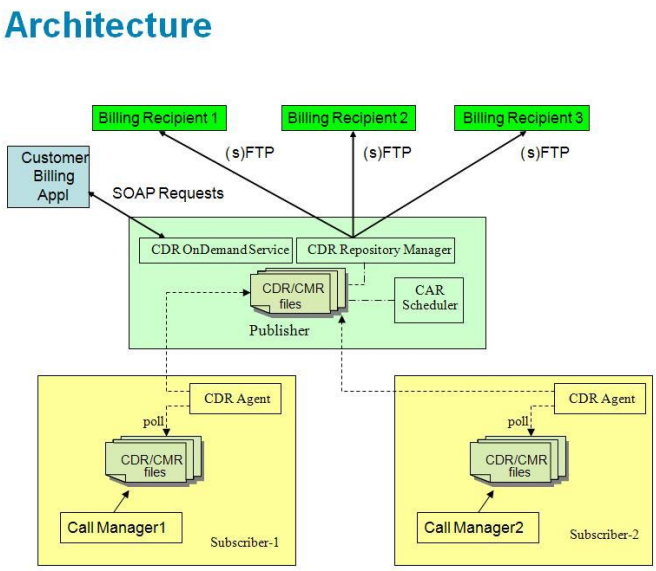


Figura A.2.1. Arquitectura de CDR

Servicios.

CallManager puede ser activado en cualquier nodo del clúster, realizando el procesamiento de llamadas y registrando los datos CDR y los datos de RMC en archivos planos. Los archivos planos se generan a una frecuencia determinada por el Intervalo de tiempo.

Agente CDR

Se ejecuta como un servicio de red en cada nodo de un clúster (incluyendo el Editor). Se sondea el directorio local para CDR/CMR creando archivos planos cada 6 segundos. Al éxito de la transferencia, el sistema elimina la copia local del archivo.

CDR *Repository Manager*

- Se ejecuta como un servicio de red en todos los nodos de un clúster. Pero, en realidad, sólo el *Repository Manager* CDR realiza alguna acción. En todos los demás nodos, el servicio simplemente se pone en marcha, pero luego se queda en modo "*sleep*".
- Crea la estructura de directorios utilizada por los servicios de CAR.
- Gestiona los archivos planos que se reciben de otros nodos. Cuando el archivo llega al nodo Repositorio CDR, el *Repository Manager* CDR lo detecta. El sistema almacena el fichero en un directorio en el que la fecha se registra con la marca de tiempo.
- Mantiene archivos CDR por un máximo de 30 días.
- Comprueba periódicamente el uso del disco y borra archivos viejos. Los umbrales se configuran mediante la gestión de CDR.
- Genera alarmas de entrega y de uso del disco.

Programador CAR.

Se ejecuta como un servicio de red en todos los nodos de un clúster. Pero, en realidad, sólo el servicio Programador CAR en el publicador realiza alguna

acción. En todos los demás nodos, el servicio simplemente se pone en marcha, pero luego pasa a estado "*sleep*".
Basado en el programa de carga de CDR, el programador CAR accede a los archivos CDR/CMR en la estructura de directorios que crea el servicio Administrador de Repositorio CDR, procesa estos archivos e inserta la información de CDR en la base de datos CAR.
El tamaño máximo de la base de datos CAR es de 6 Gb (no configurable). El programador CAR purga los datos de la base de datos CAR si excede 6 Gb o si el número de registros es superior a 2 millones. El límite de dos millones de registro incluye datos tanto de control de gasto como de fallos en el mismo.

Servicio Web CAR

Sólo se puede activar en el *Publisher*. Se ejecuta como un servicio de entidades (Centro de control - Función de Servicios). Debe estar en ejecución con el fin de acceder al análisis CDR y a la herramienta *Reporting* (CAR).

Servicio CDR *on Demand*.

El Servicio CDR *on Demand* es un servicio basado en HTTPS SOAP[45] que se ejecuta en el nodo Repositorio CDR (es decir, el *Publisher*). Recibe solicitudes SOAP para listas de nombres de archivos CDR desde un servidor de terceros basado en un intervalo de tiempo especificado por el usuario (hasta un máximo de 1 hora) y devuelve todas las listas que se ajusten a la duración que especifica la solicitud.
El Servicio CDR *on Demand* también puede gestionar las solicitudes de entrega de un archivo CDR específico a un destino indicado a través de FTP. El sistema puede activar el servicio CDR *on Demand* en el nodo Repositorio CDR, ya que tiene acceso a los archivos de CDR en el mismo.

[45] Simple Object Access Protocol es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

Servidores de facturación.

El sistema puede enviar archivos CDR hasta a tres destinos preconfigurados (servidores de facturación) utilizando FTP/SFTP. El Administrador de CDR es responsable de transferir los archivos CDR a los servidores de facturación.
En la figura A.2.2 pueden observarse las capturas de pantallas de configuración.

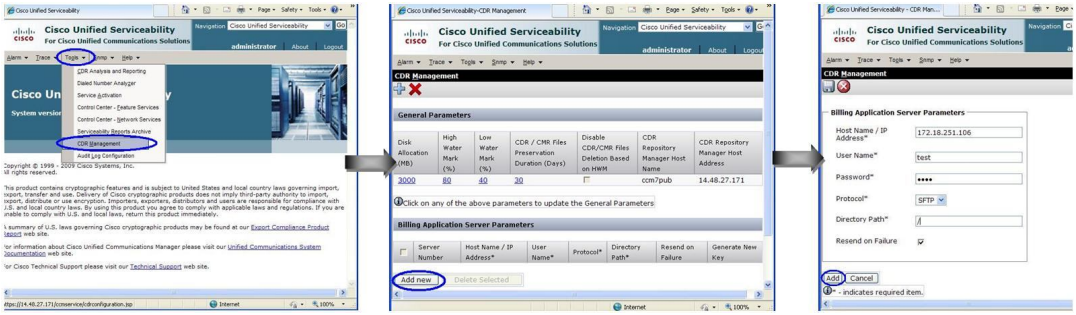


Figura A.2.2. Pantallas de configuración de servidores de facturación.

En la figura A.2.3 puede verse una captura de pantalla del proceso de carga del CDR *Scheduler* y en la figura A.2.4 la del fichero de datos CDR.

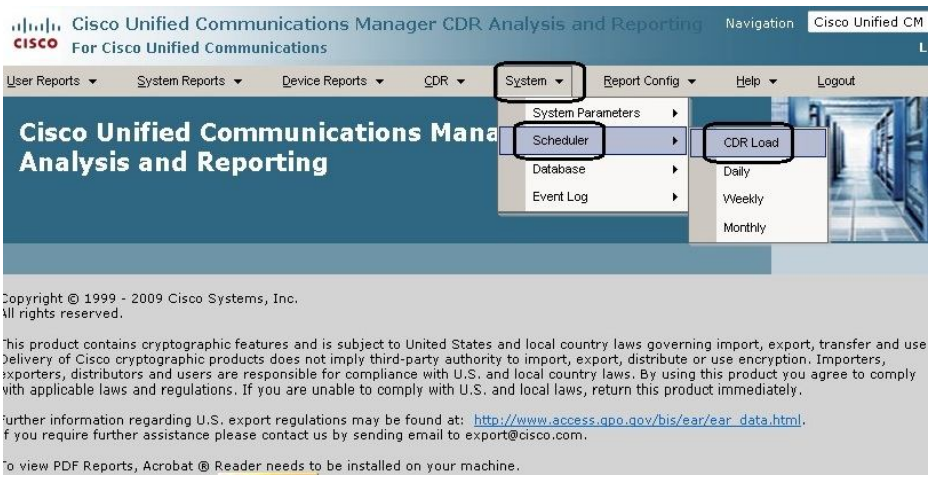


Figura A.2.3. Carga del CDR *Scheduler*.

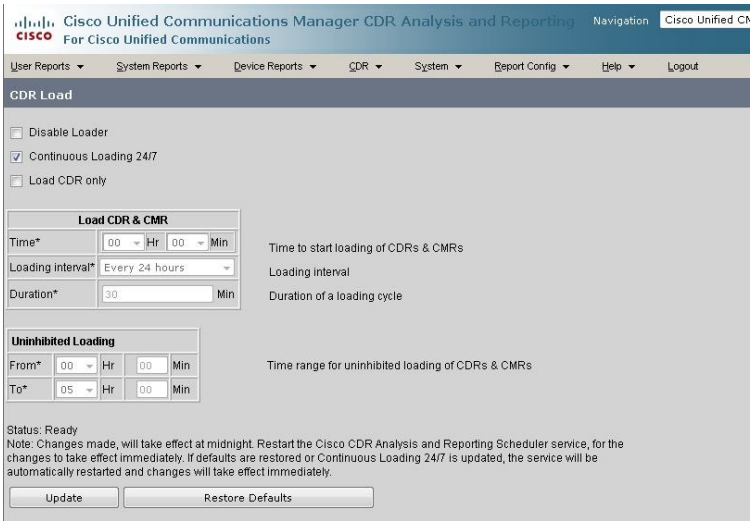


Figura A.2.4. . Carga del CDR.

Desactivado de carga de datos de CDR.

Se utiliza esta opción si no se desea que los datos de CDR se carguen en la base de datos CAR. Los cambios entran en vigor a medianoche. Se necesitará utilizar esta opción si se va a realizar operación de purga manual. Se puede forzar el cambio para que surta efecto inmediatamente al detener y reiniciar el servicio Programador CAR.

Carga contínua 24/7.

Activa el cargador CDR para que funcione continuamente las 24 horas del día, 7 días a la semana actualizando la base de datos CAR. Esta elección representa la configuración predeterminada para el Programador de carga CDR.

Anexo 3. Listado del código del generador SMDR pseudoaleatorio.

```
unit unit_generador_SMDR_aleatorio;

{$mode objfpc}{$H+}

interface

uses
  Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs,
StdCtrls,
  ExtCtrls, Spin, DateUtils;

type

  { TForm1 }

  TForm1 = class(TForm)
    Button1: TButton;
    DialogoGuardarFichero: TSaveDialog;         // Cuadro de diálogo
para guardar fichero
    Image1: TImage;
    NumDias: TSpinEdit;
    Label2: TLabel;
    NumeroDias: TLabel;
    ValorLlamadasPorDia: TSpinEdit;
    procedure Button1Click(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var
  Form1: TForm1;          // Formulario principal (y único)

implementation

{$R *.lfm}

{ TForm1 }

procedure TForm1.Button1Click(Sender: TObject);
  Var
    FicheroSMDR:textfile;             // Declaración del fichero, de
texto
    NombreInforme:string;             // Nombre del fichero
    Timbre:integer;
    Orden:string;
    NumLlamadasDia:integer;
    NumeroDiasTotal:integer;
    ContadorRegistros:integer;
    Fecha:tdatetime;
    Hora:tdatetime;
```

```
    Duracion:string;
    CHora:string;
    incremento:integer;
    contador:integer;
    ContadorDias:integer;
  begin
    if DialogoGuardarFichero.Execute then
       NombreInforme:= DialogoGuardarFichero.FileName;// Apertura de
cuadro de diálogo para asignar nombre
    NombreInforme:=NombreInforme+'.txt';              // Extensión por
defecto, txt
    Assignfile (FicheroSMDR, NombreInforme);
    randomize;                // Arrancar la generación de aleatorios
    try                       // Realiza estas operaciones en ausencia
de error
      Rewrite (FicheroSMDR); // Crea fichero para insertar datos
      NumeroDiasTotal:=strtoint(NumDias.text);
      ContadorRegistros:=1;
      NumLlamadasDia:=strtoint(ValorLlamadasPorDia.text);
      Incremento:=86400 div NumLlamadasDia;
      DateSeparator := '/';
      ShortDateFormat := 'yyyy/mm/dd';
      LongDateFormat := 'yyyy/mm/dd';
      Fecha:=date;
      Hora:=encodetime(00,00,00,00);    // Empezar a las 00:00 horas
      contador:=1;
      ContadorRegistros:=0;
      ContadorDias:=1;
      repeat // Bucle para el cambio de día
             repeat // Bucle para el cambio de hora
                    Timbre:=random(10);                  // El tiempo
en espera no supera los 10 segundos
                    CHora:=timetostr(Hora);
                    if CHora[2]=':' then CHora:='0'+CHora;  // Ajuste
de hora a 8 caracteres
                    Orden:='O';                          // Se
establece por defecto que la llamada es saliente
                    If Timbre<7 then Orden:='I';         // El 70%
(aprox) de las llamadas serán entrantes

Duracion:='00'+':'+'0'+inttostr(random(7))+':'+'0'+inttostr(random(9))
;
                    writeln (FicheroSMDR,datetostr(Fecha)+'
'+CHora+','+Duracion+','+inttostr(Timbre)+',26121969,'+Orden+',4324,43
24,,0,1000014155,0,E4324,Joe Bloggs,T9161,LINE 5.1,0,0');
                    // Writeln genera la entrada en el fichero con los
datos SMDR
                    Hora:=incsecond(Hora,Incremento);
                    contador:=contador+incremento;
                    inc(ContadorRegistros);
             until contador>=86400;
             inc(ContadorDias);
             Fecha:=Fecha+1;
             contador:=1;
             Hora:=encodetime(00,00,00,00);    // Empezar a las 00:00
horas
      until ContadorDias>=NumeroDiasTotal;
    finally
```

```
      CloseFile(FicheroSMDR);                // Cierre de fichero de
datos
      showmessage ('Fichero SMDR generado con éxito');
    end
end;

end.
```

Anexo 4. Listado del código del analizador SMDR.

```
unit unidadprincipal2;

{$mode objfpc}{$H+}{$I-}

interface

uses
  Classes, SysUtils, FileUtil, TAGraph, TASeries, TAMultiSeries,
TATools,
  TANavigation, DividerBevel, DateTimePicker, Forms, DateUtils,
Controls,
  Graphics, Dialogs, Menus, StdCtrls, ExtCtrls, Buttons, ComCtrls,
Spin,
  math, UnitDatosGenerales2;

type

  { TFormPrincipal }

  TFormPrincipal = class(TForm)
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    BitBtn3: TBitBtn;
    BitBtn4: TBitBtn;
    BitBtn5: TBitBtn;
    BitBtn6: TBitBtn;
    CopiarTraficoSemanal: TBitBtn;
    BotonCopiarTramosH: TBitBtn;
    BotonRedibujaTimbre: TBitBtn;
    BotonTodoTimbre: TBitBtn;
    BotonGuardarTimbre: TBitBtn;
    BotonCopiarTimbre: TBitBtn;
    BotonGraficaCompleta: TBitBtn;
    GuardarBurbujas: TBitBtn;
    GuardarRatioOcupacion: TBitBtn;
    CopiarBurbujas: TBitBtn;
    CopiarRatioOcupacion: TBitBtn;
    CopiarGraficaEntrantes: TBitBtn;
    GuardarGraficaEntrantes: TBitBtn;
    CopiarGraficaPerdidas: TBitBtn;
    GuardarGraficaPerdidas: TBitBtn;
    GuardarPorcentajeUso: TBitBtn;
    CopiarTimbreMedio: TBitBtn;
    GuardarTimbreMedio: TBitBtn;
    CopiarPorcentajeUso: TBitBtn;
    BotonRedibujarLineaTiempos: TBitBtn;
    GuardarGraficaGradoServicio: TBitBtn;
    CopiarCalidad: TBitBtn;
    GuardarLineaGraficaTiempos: TBitBtn;
    CopiarLineaTiempos: TBitBtn;
    CalendarioInicio: TDateTimePicker;
    CalendarioFinal: TDateTimePicker;
    CalendarioInicioT: TDateTimePicker;
    CalendarioFinalT: TDateTimePicker;
    DividerBevel4: TDividerBevel;
```

```
    DividerBevel6: TDividerBevel;
    GraficaBurbujas: TChart;
    GraficaBurbujasBubbleSeries1: TBubbleSeries;
    GradoServicio: TLabel;
    GraficaGradoServicio: TChart;
    GraficaGradoServicioLineSeries1: TLineSeries;
    GraficaGradoServicioLineSeries2: TLineSeries;
    GraficaDuracionMedia1: TLineSeries;
    GraficaDesviacionEstandar2: TLineSeries;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    DialogoGradoServicio: TSaveDialog;
    DialogoRatioEntrantes: TSaveDialog;
    DialogoPerdidas: TSaveDialog;
    DialogoPorcentajeUso: TSaveDialog;
    DialogoTimbresMedios: TSaveDialog;
    DialogoRatioOcupacion: TSaveDialog;
    DialogoBurbujas: TSaveDialog;
    ValorDesvEstLlamadas: TLabel;
    PanelNavGraficaTimbre: TChartNavPanel;
    GraficaTimbresMedios: TChart;
    GraficaPerdidas: TChart;
    GraficaCallCenterLineSeries2: TLineSeries;
    GraficaDeUso: TChart;
    GraficaDeUsoPieSeries: TPieSeries;
    GraficaCallCenter: TChart;
    GraficaCallCenterLineSeries1: TLineSeries;
    GraficaPerdidasPieSeries1: TPieSeries;
    GraficaDesvTimbreLineSeries2: TLineSeries;
    GraficaTimbresMediosBarSeries1: TBarSeries;
    PanelNavLineaTiempos: TChartNavPanel;
    ChartNavScrollBar1: TChartNavScrollBar;
    ChartNavScrollBar2: TChartNavScrollBar;
    ChartNavScrollBar3: TChartNavScrollBar;
    ChartNavScrollBar4: TChartNavScrollBar;
    ChartToolset1: TChartToolset;
    ChartToolset1PanDragTool1: TPanDragTool;
    ChartToolset1ZoomDragTool1: TZoomDragTool;
    GraficaTraficoMensual: TChart;
    GraficaTraficoMensualBarSeries1: TBarSeries;
    DividerBevel1: TDividerBevel;
    DividerBevel2: TDividerBevel;
    DividerBevel3: TDividerBevel;
    DividerBevel5: TDividerBevel;
    GraficaEntrantes1: TChart;
    GraficaEntrantesPieSeries2: TPieSeries;
    GraficaTraficoSemanal: TChart;
    GraficaTraficoSemanalBarSeries1: TBarSeries;
    HojaTraficoSemanal: TTabSheet;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    DialogoGuardarTimbre: TSaveDialog;
    DialogoGuardarLineaTiempos: TSaveDialog;
    DialogoGuardarTramosHorarios: TSaveDialog;
    DialogoTimbrePorTramos: TSaveDialog;
    DialogoTraficoSemanal: TSaveDialog;
    DialogoTraficoMensual: TSaveDialog;
```

```
    HojaCalidad: TTabSheet;
    PerdidasNoAnsiosos: TLabel;
    TiempoEsperaAceptable1: TLabel;
    ValorAWT: TSpinEdit;
    ValorGradoServicio: TLabel;
    ValorPerdidasNoAnsiosos: TLabel;
    ValorTimbreMAtendidas: TLabel;
    ValorTimbreMPerdidas: TLabel;
    Label9: TLabel;
    HojaTraficoMensual: TTabSheet;
    TimbrePorTramosBarSeries1: TBarSeries;
    TimbrePorTramos: TChart;
    HojaTimbrePorTramos: TTabSheet;
    TramosHorarios: TChart;
    TramosHorariosBarSeries1: TBarSeries;
    GraficaTimbre: TChart;
    GraficaTimbreLineSeries1: TLineSeries;
    GraficaLineaTiempos: TChart;
    GraficaLineaTiemposBarSeries1: TBarSeries;
    GraficaMediaTimbreLineSeries2: TLineSeries;
    ImagenHagaZoom: TImage;
    Label1: TLabel;
    Label2: TLabel;
    HojaTimbre: TTabSheet;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    HojaTramosHorarios: TTabSheet;
    ValorProbMenorAWT: TLabel;
    ValorASA: TLabel;
    ValorProbabilidadEspera: TLabel;
    ProbabilidadEspera: TLabel;
    ValorRatioOcupacionOperador: TLabel;
    RatioOcupacionOperador: TLabel;
    ValorIntensidadTrafico: TLabel;
    ValorLambda: TLabel;
    ValorAtendidasPorMinuto: TLabel;
    NumOperadores: TLabel;
    ValorNumOperadores: TSpinEdit;
    HojaCallCenter: TTabSheet;
    ValorDesviacionEstandar: TLabel;
    ValorTiempoTotalM: TLabel;
    ValorTiempoTotalH: TLabel;
    ValorLlamadasPorSegundo: TLabel;
    ValorLlamadasPorMinuto: TLabel;
    ValorLlamadasPorHora: TLabel;
    LlamadasPorSegundo: TLabel;
    LlamadasPorMinuto: TLabel;
    LlamadasPorHora: TLabel;
    ValorPorcentajeLlamadasCursadas: TLabel;
    LlamadasCursadas: TLabel;
    ValorNumLlamadasAtendidas: TLabel;
    LlamadasAtendidas: TLabel;
    MenuGuardarInforme: TMenuItem;
    HojaLineaTiempos: TTabSheet;
    DialogoGuardarInforme: TSaveDialog;
```

```
    ValorPorcentajeLlamadasEntrantes: TLabel;
    PorcentajeEntrantes: TLabel;
    ValorPorcentajeUso: TLabel;
    ValorPorcentajePerdidas: TLabel;
    ValorTimbreMedio: TLabel;
    ValorDuracionMedia: TLabel;
    LogoEITE: TImage;
    ImagenAnalizar: TImage;
    ImagenAbrir: TImage;
    FechaPrimeraEntrante: TLabel;
    FechaUltimaEntrante: TLabel;
    HoraPrimeraEntrante: TLabel;
    HoraUltimaEntrante: TLabel;
    PorcentajeValorUso: TLabel;
    ValorNumLlamadasPerdidas: TLabel;
    ValorNumLlamadasEntrantes: TLabel;
    ValorNumRegistros: TLabel;
    NumeroRegistros: TLabel;
    ValorTiempoTotal: TLabel;
    ValorHoraUltimaEntrante: TLabel;
    ValorFechaUltimaEntrante: TLabel;
    ValorHoraPrimeraEntrante: TLabel;
    DuracionMediaEntrantes: TLabel;
    TimbreMedio: TLabel;
    NumeroPerdidas: TLabel;
    TiempoTotal: TLabel;
    NumeroEntrantes: TLabel;
    PorcentajePerdidas: TLabel;
    HojaValoresMedios: TTabSheet;
    ValorFechaPrimeraEntrante: TLabel;
    MenuAnalisis: TMenuItem;
    MenuAyuda: TMenuItem;
    AcercaDe: TMenuItem;
    PanelGeneral: TPageControl;
    HojaDatosGenerales: TTabSheet;
    MenuPrincipal: TMainMenu;
    MenuArchivo: TMenuItem;
    SalirDelPrograma: TMenuItem;
    CerrarArchivo: TMenuItem;
    AbrirArchivo: TMenuItem;
    DialogoAbrirFichero: TOpenDialog;
    procedure AcercaDeClick(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn4Click(Sender: TObject);
    procedure BitBtn5Click(Sender: TObject);
    procedure BitBtn6Click(Sender: TObject);
    procedure CopiarTraficoSemanalClick(Sender: TObject);
    procedure BotonCopiarTramosHClick(Sender: TObject);
    procedure BotonRedibujaTimbreClick(Sender: TObject);
    procedure BotonGuardarTimbreClick(Sender: TObject);
    procedure BotonCopiarTimbreClick(Sender: TObject);
    procedure BotonGraficaCompletaClick(Sender: TObject);
    procedure BotonTodoTimbreClick(Sender: TObject);
    procedure CopiarBurbujasClick(Sender: TObject);
    procedure CopiarGraficaEntrantesClick(Sender: TObject);
    procedure CopiarGraficaPerdidasClick(Sender: TObject);
    procedure BotonRedibujarLineaTiemposClick(Sender: TObject);
```

```
    procedure CopiarCalidadClick(Sender: TObject);
    procedure CopiarLineaTiemposClick(Sender: TObject);
    procedure BotonRedibujarClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure CalendarioFinalChange(Sender: TObject);
    procedure CalendarioInicioChange(Sender: TObject);
    procedure CerrarArchivoClick(Sender: TObject);
    procedure CopiarPorcentajeUsoClick(Sender: TObject);
    procedure CopiarRatioOcupacionClick(Sender: TObject);
    procedure CopiarTimbreMedioClick(Sender: TObject);
    procedure GuardaGrafTimbreClick(Sender: TObject);
    procedure GuardaGrafTimbreTramosClick(Sender: TObject);
    procedure GuardaGrafTraficoSemanalClick(Sender: TObject);
    procedure GuardaGrafTramosHorariosClick(Sender: TObject);
    procedure GuardarBurbujasClick(Sender: TObject);
    procedure GuardarGraficaEntrantesClick(Sender: TObject);
    procedure GuardarGraficaGradoServicioClick(Sender: TObject);
    procedure GuardarGraficaPerdidasClick(Sender: TObject);
    procedure GuardarGrafTraficoMensualClick(Sender: TObject);
    procedure GuardarLineaGraficaTiemposClick(Sender: TObject);
    procedure GuardarPorcentajeUsoClick(Sender: TObject);
    procedure GuardarRatioOcupacionClick(Sender: TObject);
    procedure GuardarTimbreMedioClick(Sender: TObject);
    procedure HojaCalidadShow(Sender: TObject);
    procedure HojaCallCenterShow(Sender: TObject);
    procedure HojaLineaTiemposShow(Sender: TObject);
    procedure HojaTimbrePorTramosShow(Sender: TObject);
    procedure HojaTimbreShow(Sender: TObject);
    procedure HojaTramosHorariosShow(Sender: TObject);
    procedure HojaValoresMediosShow(Sender: TObject);
    procedure MenuAnalisisClick(Sender: TObject);
    procedure MenuGuardarInformeClick(Sender: TObject);
    procedure SalirDelProgramaClick(Sender: TObject);
    procedure AbrirArchivoClick(Sender: TObject);
    procedure HojaTraficoSemanalShow(Sender: TObject);
    procedure HojaTraficoMensualShow(Sender: TObject);
    procedure ValorAWTCCChange(Sender: TObject);
    procedure ValorAWTChange(Sender: TObject);
    procedure ValoresGeneralesClick(Sender: TObject);
    procedure ValorNumOperadoresChange(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var
  FormPrincipal: TFormPrincipal;
  NombreArchivo:string;
  UltimoRegistroEntrante:integer;
  NumEntrantes:integer;

implementation
```

```
type
  Lineas = record
    FechaLlamada:string;
    HoraLlamada:string;
    DuracionLlamada:string;
    TimbreLlamada:integer;
  end;

var
  Linea: array of Lineas;

{$R *.lfm}

{ TFormPrincipal }

//
//           Parametro, entrega el valor entre comas
//
function parametro (posicion:integer; linea:string):string;
var
  contador:integer;
  contador2:integer;
  vez:integer;
begin
  vez:=0;
  parametro:='';
  contador:=1;
  repeat
    if linea[contador]=',' then inc(vez);                    // Contador
de veces
    if vez=posicion then
      begin
        contador2:=contador+1;
        repeat
          parametro:=parametro+linea[contador2];             // Almacena
contenido parámetro
          inc(contador2);
        until linea[contador2]=',';                          // Hasta
encontrar la coma
        contador:=length(linea)-1;
      end;
    inc(contador);
  until contador=length(linea);   // Rastreo hasta final de la línea
end;


//
//                       Calcula número de registros y entrantes
//
procedure CalculaRegistros(var NumRegistros:integer);
var
  fichero:textfile;
  registro:string;
begin
     assignfile (fichero, NombreArchivo);// Asignar nombre de fichero
     NumRegistros:=0;
     NumEntrantes:=0;
     try
       reset (fichero);               // Abrir fichero datos SMDR
```

```
         while not eof(fichero) do
               begin
               readln(fichero, registro);// Lectura del registro del
fichero
               if parametro(4,registro)='I' then
NumEntrantes:=NumEntrantes+1;
               inc(NumRegistros);
               end;
         finally
           CloseFile(fichero);              // Cierre de fichero
         end;
end;

//
//                                  Abrir Archivo
//
procedure TFormPrincipal.AbrirArchivoClick(Sender: TObject);
var
    contador:integer;
    fichero:textfile;
    registro:string;
    NumRegistros:integer;
begin
    if DialogoAbrirFichero.Execute then              { Si se ejecuta
DialogoAbrir1 entonces hacer lo siguiente }
       begin
            NombreArchivo:= DialogoAbrirFichero.Filename; { Asignamos
a nombreArchivoo el valor de OpenDialog1.Filename }
            FormPrincipal.Caption:='Analizador SMDR. '+NombreArchivo;
            AbrirArchivo.Visible:=false;
            UltimoRegistroEntrante:=0;
            NumEntrantes:=0;
            NumRegistros:=0;
            CalculaRegistros(NumRegistros);
            setlength (Linea,NumRegistros+1);
            contador:=1;
            if NumEntrantes<>0 then
               begin
               assignfile (fichero, NombreArchivo);// Asignar nombre
de fichero
               try
                  reset (fichero);               // Abrir fichero
datos SMDR
                  while not eof(fichero) do
                     begin
                     readln(fichero, registro);  // Abre fichero para
lectura
                     if parametro(4,registro)='I' then   //
Comprobación de llamada entrante
                        begin
                        with Linea[contador] do
                             begin
                             FechaLlamada:=registro[1..10];    //
Almacena parámetros en tipo
                             HoraLlamada:=registro[12..19];
                             DuracionLlamada:=registro[21..28];

TimbreLlamada:=strtoint(parametro(2,registro));
                             UltimoRegistroEntrante:=contador;
```

```
                        end;
                             inc(contador);
                        end;
                    end;
                finally
                CloseFile(fichero);                          // Cierre de
fichero
                end;
                SalirDelPrograma.Visible:=true;
                CerrarArchivo.Visible:=true;
                dec (contador);
                MenuAnalisis.Visible:=true;
                ImagenAbrir.Visible:=false;
                ImagenAnalizar.Visible:=true;
                end else
                begin
                showmessage ('El fichero no contiene llamadas
entrantes');
                SalirDelPrograma.Visible:=true;
                CerrarArchivo.Visible:=false;
                AbrirArchivo.Visible:=true;
                MenuAnalisis.Visible:=false;
                ImagenAbrir.Visible:=true;
                ImagenAnalizar.Visible:=false;
                end;
       end;
end;

//
//                                Tráfico mensual
//
procedure TFormPrincipal.HojaTraficoMensualShow(Sender: TObject);
var
    ContTramos:integer;
    ContRegistros:integer;
    Tramo:array [1..12] of integer;
    NombreMes:array [1..12] of string;
    contador:integer;
begin
  ImagenHagaZoom.Visible:=false;
  GraficaTraficoMensualBarSeries1.Clear;
  ContTramos:=1;
  for contador:=1 to 12 do
      Tramo[contador]:=0;  // Inicializa contadores
  NombreMes[1]:='Ene';
  NombreMes[2]:='Feb';
  NombreMes[3]:='Mar';
  NombreMes[4]:='Abr';
  NombreMes[5]:='May';
  NombreMes[6]:='Jun';
  NombreMes[7]:='Jul';
  NombreMes[8]:='Ago';
  NombreMes[9]:='Sep';
  NombreMes[10]:='Oct';
  NombreMes[11]:='Nov';
  NombreMes[12]:='Dic';
  for ContRegistros:=1 to NumEntrantes do       // Rastrea matriz de
entrantes
      begin
```

```
      ContTramos:=strtoint(Linea[ContRegistros].FechaLlamada[6..7]);
      Tramo[ContTramos]:=Tramo[ContTramos]+1;
      end;
  for contador:=1 to 12 do                        // Muestra gráfica

GraficaTraficoMensualBarSeries1.addxy(contador,Tramo[contador],NombreM
es[contador],clSkyBlue);
end;

//
//                                Tráfico semanal
//
procedure TFormPrincipal.HojaTraficoSemanalShow(Sender: TObject);
var
    ContTramos:integer;
    ContRegistros:integer;
    Tramo:array [1..8] of integer;
    DiaDeLaSemana:array [1..7] of string;
    contador:integer;
  begin
      ImagenHagaZoom.Visible:=false;
      GraficaTraficoSemanalBarSeries1.Clear;
      ContTramos:=1;
      for contador:=1 to 8 do Tramo[contador]:=0;  // Inicializa
contadores
      for ContRegistros:=1 to NumEntrantes do       // Rastrea matriz
de entrantes
          begin

ContTramos:=dayofweek(strtodate(Linea[ContRegistros].FechaLlamada));
          if ContTramos=1 then
             ContTramos:=7
             else
             ContTramos:=ContTramos-1;
          Tramo[ContTramos]:=Tramo[ContTramos]+1;
          end;
      DiaDeLaSemana[1]:='Lunes';
      DiaDeLaSemana[2]:='Martes';
      DiaDeLaSemana[3]:='Miércoles';
      DiaDeLaSemana[4]:='Jueves';
      DiaDeLaSemana[5]:='Viernes';
      DiaDeLaSemana[6]:='Sábado';
      DiaDeLaSemana[7]:='Domingo';
      for contador:=1 to 7 do                         // Muestra gráfica

GraficaTraficoSemanalBarSeries1.addxy(contador,Tramo[contador],DiaDeLa
Semana[contador],clSkyBlue);
end;

//
//                                  Valores generales
//
procedure TFormPrincipal.ValoresGeneralesClick(Sender: TObject);
begin
  ImagenAnalizar.Visible:=false;
end;

//
//                                Salir del programa
```

```
//
procedure TFormPrincipal.SalirDelProgramaClick(Sender: TObject);
begin
    Close;
end;

//
//                                Acerca de...
//
procedure TFormPrincipal.AcercaDeClick(Sender: TObject);
begin
  showmessage ('Proyecto Fin de Carrera. Analizador de ficheros SMDR.
Autor: Juan Francisco Rodríguez Báez. Tutores: Carlos Ramírez Casañas
y Adán San Gil Martín.');
end;

//
//                         Guardar gráfica tramos horarios
//
procedure TFormPrincipal.BitBtn1Click(Sender: TObject);
var
  NombreFichero:string;
begin
  if DialogoGuardarTramosHorarios.Execute then
     begin
     NombreFichero:= DialogoGuardarTramosHorarios.FileName;
     NombreFichero:=NombreFichero+'.jpg';
     TRamosHorarios.SaveToFile(TjpegImage,NombreFichero);
     end;
end;

//
//        Copiar al portapapeles la gráfica de timbre por tramos
//
procedure TFormPrincipal.BitBtn2Click(Sender: TObject);
begin
  TimbrePorTramos.CopyToClipboardBitmap;
end;

//
//                         Guardar timbre por tramos horarios
//
procedure TFormPrincipal.BitBtn3Click(Sender: TObject);
var
  NombreFichero:string;
begin
    if DialogoTimbrePorTramos.Execute then
       begin
       NombreFichero:= DialogoTimbrePorTramos.FileName;
       NombreFichero:=NombreFichero+'.jpg';
       TimbrePorTramos.SaveToFile(TjpegImage,NombreFichero);
       end;
end;

//
//                         Guardar gráfica tráfico semanal
//
procedure TFormPrincipal.BitBtn4Click(Sender: TObject);
var
```

```
  NombreFichero:string;
begin
  if DialogoTraficoSemanal.Execute then
     begin
     NombreFichero:= DialogoTraficoSemanal.FileName;
     NombreFichero:=NombreFichero+'.jpg';
     GraficaTraficoSemanal.SaveToFile(TjpegImage,NombreFichero);
     end;
end;

//
//        Copiar al portapapeles la gráfica de tráfico mensual
//
procedure TFormPrincipal.BitBtn5Click(Sender: TObject);
begin
    GraficaTraficoMensual.CopyToClipboardBitmap;
end;

//
//                        Guardar gráfica tráfico mensual
//
procedure TFormPrincipal.BitBtn6Click(Sender: TObject);
var
  NombreFichero:string;
begin
  if DialogoTraficoMensual.Execute then
     begin
     NombreFichero:= DialogoTraficoMensual.FileName;
     NombreFichero:=NombreFichero+'.jpg';
     GraficaTraficoMensual.SaveToFile(TjpegImage,NombreFichero);
     end;
end;

//
//        Copiar al portapapeles la gráfica de tráfico semanal
//
procedure TFormPrincipal.CopiarTraficoSemanalClick(Sender: TObject);
begin
  GraficaTraficoSemanal.CopyToClipboardBitmap;
end;

procedure TFormPrincipal.BotonCopiarTramosHClick(Sender: TObject);
begin
  TramosHorarios.CopyToClipboardBitmap;
end;

//
//                                    Guardar gráfica timbre
//
procedure TFormPrincipal.BotonGuardarTimbreClick(Sender: TObject);
var
  NombreFichero:string;
begin
  if DialogoGuardarTimbre.Execute then
     begin
     NombreFichero:= DialogoGuardarTimbre.FileName;
     NombreFichero:=NombreFichero+'.jpg';
     GraficaTimbre.SaveToFile(TjpegImage,NombreFichero);
     end;
```

```
end;

//
//        Copiar al portapapeles la gráfica de Timbre
//
procedure TFormPrincipal.BotonCopiarTimbreClick(Sender: TObject);
begin
  GraficaTimbre.CopyToClipboardBitmap;
end;

//
//                                    Volver a gráfica completa linea
tiempos
//
procedure TFormPrincipal.BotonGraficaCompletaClick(Sender: TObject);
var
  contador:integer;
  Duracion:integer;
  NumARepresentar:integer;
  Titulo:string;
begin
  ImagenHagaZoom.Visible:=true;
  contador:=1;
     NumARepresentar:=NumEntrantes;
  Titulo:='De '+Linea[1].FechaLlamada+' a
'+Linea[NumARepresentar].FechaLlamada;
  GraficaLineaTiempos.BottomAxis.Title.Caption:=Titulo;
  for contador:=1 to NumARepresentar do
      begin
      Duracion:=strtoint(Linea[contador].DuracionLlamada[7..8])+
      strtoint(Linea[contador].DuracionLlamada[4..5])*60+
      strtoint(Linea[contador].DuracionLlamada[1..2])*3600;
      If (Duracion<>0) then   // Representa llamadas
         GraficaLineaTiemposBarSeries1.addxy(contador, Duracion,
         Linea[contador].FechaLlamada, clGreen)
      else                    // Representa perdidas
        GraficaLineaTiemposBarSeries1.addxy(contador, Duracion,
        Linea[contador].FechaLlamada, clRed);
      end;
  CalendarioInicio.Date:=strtodate(Linea[1].FechaLlamada);
  CalendarioInicio.Time:=strtotime(Linea[1].HoraLlamada);
  CalendarioFinal.Date:=strtodate(Linea[NumEntrantes].FechaLlamada);
  CalendarioFinal.Time:=strtotime(Linea[NumEntrantes].HoraLlamada);
  CalendarioInicio.MinDate:=strtodate(Linea[1].FechaLlamada);

CalendarioInicio.MaxDate:=strtodate(Linea[NumEntrantes].FechaLlamada);
  CalendarioFinal.MinDate:=strtodate(Linea[1].FechaLlamada);

CalendarioFinal.MaxDate:=strtodate(Linea[NumEntrantes].FechaLlamada);
end;

//
//        Copiar al portapapeles la gráfica de burbujas
//
procedure TFormPrincipal.CopiarBurbujasClick(Sender: TObject);
begin
  GraficaBurbujas.CopyToClipboardBitmap;
end;
```

```
//
//        Copiar al portapapeles la gráfica de Entrantes
//
procedure TFormPrincipal.CopiarGraficaEntrantesClick(Sender: TObject);
begin
  GraficaEntrantes1.CopyToClipboardBitmap;
end;

//
//        Copiar al portapapeles la gráfica de Perdidas
//
procedure TFormPrincipal.CopiarGraficaPerdidasClick(Sender: TObject);
begin
  GraficaPerdidas.CopyToClipboardBitmap;
end;

//
//        Copiar al portapapeles la gráfica de Grado de Servicio
//
procedure TFormPrincipal.CopiarCalidadClick(Sender: TObject);
begin
  GraficaGradoServicio.CopyToClipboardBitmap;
end;

//
//        Copiar al portapapeles la gráfica de línea de tiempos
//
procedure TFormPrincipal.CopiarLineaTiemposClick(Sender: TObject);
begin
  GraficaLineaTiempos.CopyToClipboardBitmap;
end;

procedure TFormPrincipal.BotonRedibujarClick(Sender: TObject);
begin

end;

procedure TFormPrincipal.Button1Click(Sender: TObject);
begin

end;


//
//                        Si se cambia fecha/Hora Final...
//
procedure TFormPrincipal.CalendarioFinalChange(Sender: TObject);
begin
  BotonRedibujarLineaTiempos.Enabled:=true;
end;

//
//                        Si se cambia fecha/Hora Inicial...
//
procedure TFormPrincipal.CalendarioInicioChange(Sender: TObject);
begin
  BotonRedibujarLineaTiempos.Enabled:=true;
end;
```

```
//
//        Copiar al portapapeles la gráfica de tramos horarios
//
procedure TFormPrincipal.Button5Click(Sender: TObject);
begin
  TramosHorarios.CopyToClipboardBitmap;
end;

procedure TFormPrincipal.Button6Click(Sender: TObject);
begin

end;

procedure TFormPrincipal.Button7Click(Sender: TObject);
begin

end;

procedure TFormPrincipal.Button8Click(Sender: TObject);
begin

end;

//
//                       Cerrar Archivo
//
procedure TFormPrincipal.CerrarArchivoClick(Sender: TObject);
begin
  PanelGeneral.Visible:=false;
  FormPrincipal.Caption:='Analizador SMDR.';
  ImagenAnalizar.Visible:=false;      // ocultar flecha analizar
  ImagenAbrir.Visible:=true;          // Mostrar flecha de abrir
  ImagenHagaZoom.Visible:=false;      // ocultar flecha zoom
  AbrirArchivo.Visible:=true;         // Mostrar opción de abrir
archivo
  GraficaLineaTiemposBarSeries1.Clear;
  MenuGuardarInforme.Visible:=false;
end;

//
//        Copiar al portapapeles la gráfica de porcentaje de uso
//
procedure TFormPrincipal.CopiarPorcentajeUsoClick(Sender: TObject);
begin
  GraficadeUso.CopyToClipboardBitmap;
end;

//
//        Copiar al portapapeles la gráfica de Call Center
//
procedure TFormPrincipal.CopiarRatioOcupacionClick(Sender: TObject);
begin
  GRaficaCallCenter.CopyToClipboardBitmap;
end;

//
//        Copiar al portapapeles la gráfica de timbres medios
//
procedure TFormPrincipal.CopiarTimbreMedioClick(Sender: TObject);
```

```
begin
  GraficaTimbresMedios.CopyToClipboardBitmap;
end;

procedure TFormPrincipal.GuardaGrafTimbreClick(Sender: TObject);
begin

end;

procedure TFormPrincipal.GuardaGrafTimbreTramosClick(Sender: TObject);
begin

end;

procedure TFormPrincipal.GuardaGrafTraficoSemanalClick(Sender:
TObject);
begin

end;

procedure TFormPrincipal.GuardaGrafTramosHorariosClick(Sender:
TObject);
begin

end;

//
//                        Guardar gráfica burbujas
//
procedure TFormPrincipal.GuardarBurbujasClick(Sender: TObject);
var
  NombreFichero:string;
begin
      if DialogoBurbujas.Execute then
         begin
         NombreFichero:=DialogoBurbujas.FileName;
         NombreFichero:=NombreFichero+'.jpg';
         GraficaBurbujas.SaveToFile(TjpegImage,NombreFichero);
         end;
end;

//
//                        Guardar gráfica Ratio Entrantes
//
procedure TFormPrincipal.GuardarGraficaEntrantesClick(Sender:
TObject);
var
  NombreFichero:string;
begin
      if DialogoRatioEntrantes.Execute then
         begin
         NombreFichero:= DialogoRatioEntrantes.FileName;
         NombreFichero:=NombreFichero+'.jpg';
         GraficaEntrantes1.SaveToFile(TjpegImage,NombreFichero);
         end;
end;

//
//                        Guardar gráfica Grado Servicio
```

```
//
procedure TFormPrincipal.GuardarGraficaGradoServicioClick(Sender:
TObject);
var
  NombreFichero:string;
begin
      if DialogoGradoServicio.Execute then
         begin
         NombreFichero:= DialogoGradoServicio.FileName;
         NombreFichero:=NombreFichero+'.jpg';
         GraficaGradoServicio.SaveToFile(TjpegImage,NombreFichero);
         end;
end;

//
//                        Guardar gráfica Perdidas
//
procedure TFormPrincipal.GuardarGraficaPerdidasClick(Sender: TObject);
var
  NombreFichero:string;
begin
  if DialogoPerdidas.Execute then
     begin
     NombreFichero:=DialogoPerdidas.FileName;
     NombreFichero:=NombreFichero+'.jpg';
     GraficaPerdidas.SaveToFile(TjpegImage,NombreFichero);
     end;
end;

procedure TFormPrincipal.GuardarGrafTraficoMensualClick(Sender:
TObject);
begin

end;

//
//                                    Guardar gráfica línea de tiempos
//
procedure TFormPrincipal.GuardarLineaGraficaTiemposClick(Sender:
TObject);
var
  NombreFichero:string;
begin
      if DialogoGuardarLineaTiempos.Execute then
         begin
         NombreFichero:= DialogoGuardarLineaTiempos.FileName;
         NombreFichero:=NombreFichero+'.jpg';
         GraficaLineaTiempos.SaveToFile(TjpegImage,NombreFichero);
         end;
end;

//
//                                    Guardar gráfica Porcentaje uso
//
procedure TFormPrincipal.GuardarPorcentajeUsoClick(Sender: TObject);
var
  NombreFichero:string;
begin
      if DialogoPorcentajeUso.Execute then
```

```
         begin
         NombreFichero:=DialogoPorcentajeUso.FileName;
         NombreFichero:=NombreFichero+'.jpg';
         GraficaDeUso.SaveToFile(TjpegImage,NombreFichero);
         end;
end;

//
//                         Guarda Grafica Ratio ocupación
//
procedure TFormPrincipal.GuardarRatioOcupacionClick(Sender: TObject);
var
  NombreFichero:string;
begin
      if DialogoRatioOcupacion.Execute then
         begin
         NombreFichero:=DialogoRatioOcupacion.FileName;
         NombreFichero:=NombreFichero+'.jpg';
         GraficaCallCenter.SaveToFile(TjpegImage,NombreFichero);
         end;
end;

//
//                         Guarda Grafica Timbres Medios
//
procedure TFormPrincipal.GuardarTimbreMedioClick(Sender: TObject);
var
  NombreFichero:string;
begin
      if DialogoTimbresMedios.Execute then
         begin
         NombreFichero:=DialogoTimbresMedios.FileName;
         NombreFichero:=NombreFichero+'.jpg';
         GraficaTimbresMedios.SaveToFile(TjpegImage,NombreFichero);
         end;
end;

//
//                         Calcula duración media entrantes
//
procedure CalculaDuracionMedia (var DuracionMediaEnSegundos:real);
var
  contador:integer;
begin
  DuracionMediaEnSegundos:=0;
  for contador:=1 to NumEntrantes do
      begin  //                         Calcula duración media
entrantes

DuracionMediaEnSegundos:=DuracionMediaEnSegundos+strtoint(Linea[contad
or].DuracionLlamada[7..8])
      +strtoint(Linea[contador].DuracionLlamada[4..5])*60+
      strtoint(Linea[contador].DuracionLlamada[1..2])*3600;
      end;
  DuracionMediaEnSegundos:=DuracionMediaEnSegundos/NumEntrantes;
end;

//
//            Redibujar linea de tiempos
```

```
//
procedure TFormPrincipal.BotonRedibujarLineaTiemposClick(Sender:
TObject);
var
  contador,ContadorI,ContadorF:integer;
  Duracion:integer;
  Titulo:string;
  Sumatoria:real;
  Desviacion:real;
  TiempoMedioLlamadas:real;
begin
    ContadorI:=1;
    ContadorF:=1;
    Contador:=0;
    repeat
      inc(contador);
    until
(Linea[contador].FechaLlamada=datetostr(CalendarioInicio.Date)) or
(contador=NumEntrantes);
    ContadorI:=contador;
    contador:=NumEntrantes+1;
    repeat
      dec(contador);
    until
(Linea[contador].FechaLlamada=datetostr(CalendarioFinal.Date)) or
(contador=1);
    ContadorF:=contador;
    GraficaLineaTiemposBarSeries1.Clear;
    Titulo:='De '+Linea[ContadorI].FechaLlamada+' a
'+Linea[ContadorF].FechaLlamada;
    GraficaLineaTiempos.BottomAxis.Title.Caption:=Titulo;
    CalculaDuracionMedia(TiempoMedioLlamadas);
    for contador:=ContadorI to ContadorF do
        begin
        Duracion:=strtoint(Linea[contador].DuracionLlamada[7..8])+
        strtoint(Linea[contador].DuracionLlamada[4..5])*60+
        strtoint(Linea[contador].DuracionLlamada[1..2])*3600;
        If (Duracion<>0) then   // Representa llamadas
          GraficaLineaTiemposBarSeries1.addxy(contador, Duracion,
          Linea[contador].FechaLlamada, clGreen)
        else                    // Representa perdidas
          GraficaLineaTiemposBarSeries1.addxy(contador, Duracion,
          Linea[contador].FechaLlamada, clRed);
        Sumatoria:=Sumatoria+sqr(Duracion-TiempoMedioLlamadas);
        end;
    BotonGraficacompleta.Enabled:=true;
    Desviacion:=sqrt(Sumatoria/(NumEntrantes-1));
    ValorDesvEstLlamadas.Caption:=floattostr(Desviacion)[1..6];
    ValorDesvEstLlamadas.Visible:=true; // Calcula y muestra
desviación estándar
end;



//
//                          Calcula timbre medio entrantes
//
procedure CalculaTimbreMedio (var TimbreMedio:real);
var
```

```
  contador:integer;
begin
  TimbreMedio:=0;//                          Calcula timbre medio
entrantes
  for contador:=1 to NumEntrantes do
      TimbreMedio:=TimbreMedio+Linea[contador].TimbreLlamada;
  TimbreMedio:=TimbreMedio/NumEntrantes;
end;

//
//                                  Gráfica completa de timbre
//
procedure TFormPrincipal.BotonTodoTimbreClick(Sender: TObject);
var
  contador:integer;
  TiempoMedioTimbre:real;
  Desviacion:real;
  Sumatoria:real;
begin
  ImagenHagaZoom.Visible:=true;
  GraficaTimbreLineSeries1.Clear;
  GraficaMediaTimbreLineSeries2.Clear;
  GraficaDesvTimbreLineSeries2.Clear;
  contador:=1;             // Inicializo variables
  TiempoMedioTimbre:=0;
  CalculaTimbreMedio(TiempoMedioTimbre);
  Sumatoria:=0;
  for contador:=1 to NumEntrantes do
      begin            // Simulatanea las 2 gráficas (Timbre y media)
      GraficaTimbreLineSeries1.AddXY(contador,
Linea[contador].TimbreLlamada,
         Linea[contador].HoraLlamada, clGreen);
      GraficaMediaTimbreLineSeries2.AddXY(contador, TiempoMedioTimbre,
         Linea[contador].HoraLlamada, clRed);
      Sumatoria:=Sumatoria+sqr(Linea[contador].TimbreLlamada-
TiempoMedioTimbre)
      end;
  // Mostrar desviación estándar
  Desviacion:=sqrt(Sumatoria/(NumEntrantes-1));
  ValorDesviacionEstandar.Caption:=floattostr(Desviacion)[1..6];
  ValorDesviacionEstandar.Visible:=true; // Calcula y muestra
desviación estándar
  for contador:=1 to NumEntrantes do

GraficaDesvTimbreLineSeries2.AddXY(contador,Desviacion,Linea[contador]
.HoraLlamada);
  CalendarioInicioT.Date:=strtodate(Linea[1].FechaLlamada);
  CalendarioInicioT.Time:=strtotime(Linea[1].HoraLlamada);
  CalendarioFinalT.Date:=strtodate(Linea[NumEntrantes].FechaLlamada);
  CalendarioFinalT.Time:=strtotime(Linea[NumEntrantes].HoraLlamada);
  CalendarioInicioT.MinDate:=strtodate(Linea[1].FechaLlamada);

CalendarioInicioT.MaxDate:=strtodate(Linea[NumEntrantes].FechaLlamada)
;
  CalendarioFinalT.MinDate:=strtodate(Linea[1].FechaLlamada);

CalendarioFinalT.MaxDate:=strtodate(Linea[NumEntrantes].FechaLlamada);
end;
```

```
//
//            Redibujar gráfica timbre
//
procedure TFormPrincipal.BotonRedibujaTimbreClick(Sender: TObject);
var
  contador,ContadorI,ContadorF:integer;
  TiempoMedioTimbre:real;
  Sumatoria:real;
  Desviacion:real;
  Titulo:string;
begin
  ContadorI:=1;
  ContadorF:=1;
  Contador:=0;
  sumatoria:=0;
  TiempoMedioTimbre:=0;
  CalculaTimbreMedio(TiempoMedioTimbre);
  repeat
    inc(contador);   // Buscamos fecha de inicio buscada
  until
(Linea[contador].FechaLlamada=datetostr(CalendarioInicioT.Date)) or
(contador=NumEntrantes);
  ContadorI:=contador;
  contador:=NumEntrantes+1;
  repeat
    dec(contador);    // Buscamos fecha final buscada de forma
decreciente
  until
(Linea[contador].FechaLlamada=datetostr(CalendarioFinalT.Date)) or
(contador=1);
  ContadorF:=contador;
  GraficaTimbreLineSeries1.Clear;
  GraficaMediaTimbreLineSeries2.Clear;
  GraficaDesvTimbreLineSeries2.Clear;
  Titulo:='De '+Linea[ContadorI].FechaLlamada+' a
'+Linea[ContadorF].FechaLlamada;
  GraficaTimbre.BottomAxis.Title.Caption:=Titulo;
  GraficaTimbre.BottomAxis.Title.Visible:=true;
  for contador:=ContadorI to ContadorF do
      begin
        GraficaTimbreLineSeries1.addxy(contador,
Linea[contador].TimbreLlamada,
          Linea[contador].HoraLlamada, clGreen);
        GraficaMediaTimbreLineSeries2.AddXY(contador,
TiempoMedioTimbre,
          Linea[contador].HoraLlamada, clRed);
        Sumatoria:=Sumatoria+sqr(Linea[contador].TimbreLlamada-
TiempoMedioTimbre);
      end;
  Desviacion:=sqrt(Sumatoria/(NumEntrantes-1));
  ValorDesviacionEstandar.Caption:=floattostr(Desviacion)[1..6];
  ValorDesviacionEstandar.Visible:=true; // Calcula y muestra
desviación estándar
  for contador:=ContadorI to ContadorF do
    GraficaDesvTimbreLineSeries2.AddXY(contador,Desviacion,
      Linea[contador].HoraLlamada);
  BotonGraficacompleta.Enabled:=true;
end;
```

```
//
//                          Calcula timbre medio atendidas
//
procedure CalculaTimbreMedioA (var TimbreMedioA:real);
var
  contador:integer;
  NumAtendidas:integer;
begin
  TimbreMedioA:=0; //                          Calcula timbre medio
atendidas
  NumAtendidas:=0;
  for contador:=1 to NumEntrantes do
      if Linea[contador].DuracionLlamada<>'00:00:00' then
         begin
         TimbreMedioA:=TimbreMedioA+Linea[contador].TimbreLlamada;
         inc(NumAtendidas);
         end;
  if NumAtendidas=0 then NumAtendidas:=1;
  TimbreMedioA:=TimbreMedioA/NumAtendidas;
end;

//
//                          Calcula timbre medio perdidas
//
procedure CalculaTimbreMedioP (var TimbreMedioP:real);
var
  contador:integer;
  NumPerdidas:integer;
begin
  TimbreMedioP:=0; //                          Calcula timbre medio
atendidas
  NumPerdidas:=0;
  for contador:=1 to NumEntrantes do
      if Linea[contador].DuracionLlamada='00:00:00' then
         begin
         TimbreMedioP:=TimbreMedioP+Linea[contador].TimbreLlamada;
         inc(NumPerdidas);
         end;
  if NumPerdidas=0 then NumPerdidas:=1;
  TimbreMedioP:=TimbreMedioP/NumPerdidas;
end;

procedure TFormPrincipal.Button2Click(Sender: TObject);
begin

end;

//
//                          cALCULA NÚMERO DE PERDIDAS entrantes
//
Procedure CalculaPerdidas (var NumPerdidas:integer; var
VPorcentajePerdidas:real);
var
  Cadena:string;
  contador:integer;
begin
  NumPerdidas:=0;
  for contador:=1 to NumEntrantes do
```

```
      begin   //                          cALCULA NÚMERO DE PERDIDAS
entrantes
      Cadena:=Linea[contador].DuracionLlamada;
      if Cadena='00:00:00' then NumPerdidas:=NumPerdidas+1;
      end;
  VPorcentajePerdidas:=NumPerdidas*100/NumEntrantes;
end;

procedure TFormPrincipal.HojaCalidadShow(Sender: TObject);
var
  VPorcentajePerdidas:real;
  SL:real;
  AWT:integer;
  NumPerdidas:integer;
  CursadasAWT:integer;
  contador:integer;
  CursadasNoAnsiosos:integer;
  VPerdidasNoAnsiosos:real;
begin
  // Gráfica AWT
  GraficaGradoServicioLineSeries1.Clear;
  GraficaGradoServicioLineSeries2.Clear;
  ValorAWT.Value:=0;
  for AWT:=1 to 20 do
      begin
      NumPerdidas:=0;       // Inicializado de variables
      CursadasAWT:=0;
      VPorcentajePerdidas:=0;
      CalculaPerdidas(NumPerdidas,VPorcentajePerdidas);
      contador:=1;        //          Cálculo del Grado de Servicio
      for contador:=1 to NumEntrantes do
          if (Linea[contador].TimbreLlamada<AWT) and
             (Linea[contador].DuracionLlamada<>'00:00:00') then
inc(CursadasAWT);
      SL:=100*CursadasAWT/(NumEntrantes-NumPerdidas);
      GraficaGradoServicioLineSeries1.AddXY(AWT,SL,'Grado de
Servicio');
      CursadasNoAnsiosos:=0;//          Cálculo de las pérdidas por no
ansiosos
      for contador:=1 to NumEntrantes do
          if (Linea[contador].TimbreLlamada>AWT) and      // Calculado
para margen dado
          (Linea[contador].DuracionLlamada='00:00:00') then
      inc (CursadasNoAnsiosos);
      if NumPerdidas=0 then
         VPerdidasNoAnsiosos:=100*CursadasNoAnsiosos
         else
         VPerdidasNoAnsiosos:=100*CursadasNoAnsiosos/NumPerdidas;

GraficaGradoServicioLineSeries2.AddXY(AWT,VPerdidasNoAnsiosos,'Perdida
s no ansiosos');
      end;
end;


//
//                                    Calcula potencia para Erlang
C
//
```

```
Function Potencia(NumOper:integer;IntensTrafico:real):real;
var
  acumulador:real;
  contador:integer;
begin
  acumulador:=0;
  for contador:=1 to NumOper do
      if IntensTrafico<>0 then
acumulador:=acumulador+log10(IntensTrafico/contador)
        else acumulador:=0;
  Potencia:=exp(acumulador);
end;

//
//                                                  Calculadora Erlang
C
//
Function ErlangC(NumOper:integer;IntensTrafico:real):real;
var
  acumulador:real;
  contador:integer;
  ValorPotencia:real;
begin
  ValorPotencia:=Potencia(NumOper,IntensTrafico);
  acumulador:=1;
  for contador:=1 to NumOper-1 do
      acumulador:=acumulador+Potencia(contador,IntensTrafico);
  ErlangC:=ValorPotencia/(ValorPotencia+(1-
IntensTrafico/NumOper)*acumulador);
end;

//
//                       Calculador de tiempo total en segundos
//
Procedure CalculaTiempoTotal(FechaI, HoraI, FechaF, HoraF: string; var
TotalSegundos:integer);
var
   PrimeraFecha, UltimaFecha:TDate;
   PrimeraHora, UltimaHora:TTime;
begin
  ShortDateFormat:='yyyy/mm/dd';      // Formateo de fechas
  LongDateFormat:='yyyy/mm/dd';
  PrimeraFecha:=strtodate(FechaI);
  UltimaFecha:=strtodate(FechaF);
  TotalSegundos:=SecondsBetween(PrimeraFecha,UltimaFecha); // Cálculo
de segundos entre fechas
  PrimeraHora:=strtotime(HoraI);
  UltimaHora:=strtotime(HoraF);
  if PrimeraHora>UltimaHora then
     TotalSegundos:=TotalSegundos-
SecondsBetween(UltimaHora,PrimeraHora);
  if PrimeraHora<UltimaHora then

TotalSegundos:=TotalSegundos+SecondsBetween(UltimaHora,PrimeraHora);
end;

//
//        Valor de tiempo de espera aceptable (AWT) en pestaña de
calidad
```

```
//
procedure TFormPrincipal.ValorAWTChange(Sender: TObject);
var
  CursadasNoAnsiosos,contador,CursadasAWT,NumPerdidas,AWT:integer; //
AWT = Tiempo de Espera Aceptable
  VPorcentajePerdidas,SL:real;  // Grado de Servicio o GoS
begin
  AWT:=valorAWT.Value;
  NumPerdidas:=0;       // Inicializado de variables
  CursadasAWT:=0;
  VPorcentajePerdidas:=0;
  CalculaPerdidas(NumPerdidas,VPorcentajePerdidas);
  contador:=1;        //          Cálculo del Grado de Servicio
  for contador:=1 to NumEntrantes do
      if (Linea[contador].TimbreLlamada<AWT) and
      (Linea[contador].DuracionLlamada<>'00:00:00') then
inc(CursadasAWT);
  SL:=100*CursadasAWT/(NumEntrantes-NumPerdidas);
  ValorGradoServicio.Caption:=floattostr(SL)[1..6];
  ValorGradoServicio.Visible:=true;
  CursadasNoAnsiosos:=0;//          Cálculo de las pérdidas por no
ansiosos
  for contador:=1 to NumEntrantes do
      if (Linea[contador].TimbreLlamada>AWT) and      // Calculado
para margen dado
      (Linea[contador].DuracionLlamada='00:00:00') then
         inc (CursadasNoAnsiosos);
  if NumPerdidas=0 then

ValorPerdidasNoAnsiosos.Caption:=floattostr(100*CursadasNoAnsiosos)[1.
.6]
     else

ValorPerdidasNoAnsiosos.Caption:=floattostr(100*CursadasNoAnsiosos/Num
Perdidas)[1..6];
  ValorPerdidasNoAnsiosos.Visible:=true;
end;

//
//                      Calcula PEMAWT
//
procedure TFormPrincipal.ValorAWTCCChange(Sender: TObject);
begin
end;

//
//                              Cambio en el número de operadores en
pestaña calidad
//
procedure TFormPrincipal.ValorNumOperadoresChange(Sender: TObject);
var
  NumOper,AWT:integer; // AWT = Tiempo de Espera Aceptable

ValorProbEspera,DuracionMediaEnSegundos,PEMAWT,ASA,IntensTrafico,Lambd
a:real;  // Grado de Servicio o GoS
  TotalSegundos:integer;
  OcupacionOperador:real;
begin
  GraficaBurbujasBubbleSeries1.Clear;
```

```
  ValorNumOperadores.MinValue:=1;
  ValorNumOperadores.MaxValue:=100;
  CalculaTiempoTotal(Linea[1].FechaLlamada,Linea[1].HoraLlamada,
    Linea[UltimoRegistroEntrante].FechaLlamada,
    Linea[UltimoRegistroEntrante].HoraLlamada, TotalSegundos);
  Lambda:=NumEntrantes/TotalSegundos;// Cálculo de Lambda
  ValorLambda.Caption:=formatfloat('0.###',lambda);
  ValorLambda.Visible:=true;    // u= Lambda * Duracion media
  CalculaDuracionMedia(DuracionMediaEnSegundos);
  IntensTrafico:=Lambda*DuracionMediaEnSegundos;
  ValorIntensidadTrafico.Caption:=floattostr(IntensTrafico)[1..6];
  ValorIntensidadTrafico.Visible:=true;    // Ratio ocupación agentes
  OcupacionOperador:=100*IntensTrafico/ValorNumOperadores.value;

ValorRatioOcupacionOperador.Caption:=floattostr(OcupacionOperador)[1..
6];
  ValorRatioOcupacionOperador.Visible:=true;    // Valor Probabilidad
Espera
  NumOper:=strtoint(ValorNumOperadores.caption);
  if IntensTrafico=0 then ValorProbEspera:=0
     else ValorProbEspera:=100*ErlangC(NumOper,IntensTrafico);
  if ValorProbEspera>=0 then

ValorProbabilidadEspera.Caption:=floattostr(ValorProbEspera)[1..6]
     else
     ValorProbabilidadEspera.Caption:='Inconsistente';
  ValorProbabilidadEspera.Visible:=true;    // Valor ASA (Tiempo medio
de respuesta
  DuracionMediaEnSegundos:=0;
  CalculaDuracionMedia (DuracionMediaEnSegundos);

ASA:=(ErlangC(NumOper,IntensTrafico)*DuracionMediaEnSegundos/(NumOper-
IntensTrafico));
  if ASA>=0 then ValorASA.Caption:=floattostr(ASA)[1..6]
     else ValorASA.Caption:='Inconsistente';    // Probabilidad de
espera menor que el AWT (PEMAWT)
  ValorASA.Visible:=true;
  PEMAWT:=1-ErlangC(NumOper,IntensTrafico)*exp(-(NumOper-
IntensTrafico)*(ASA/DuracionMediaEnSegundos));
  if (PEMAWT>=0) and (ASA>=0) and (ValorProbEspera>=0) then
     ValorProbMenorAWT.Caption:=floattostr(100*PEMAWT)[1..6]
     else ValorProbMenorAWT.Caption:='Inconsistente';
  ValorProbMenorAWT.Visible:=true;
  if (PEMAWT>=0) and (ASA>=0) and (ValorProbEspera>=0) then
     begin
     if (OcupacionOperador<50) then

GraficaBurbujasBubbleSeries1.AddXY(300,300,OcupacionOperador,'Ocupació
n operador',clGreen)
        else

GraficaBurbujasBubbleSeries1.AddXY(300,300,OcupacionOperador,'Ocupació
n operador',clMoneyGreen);
     if ValorProbEspera>15 then

GraficaBurbujasBubbleSeries1.AddXY(50,300,ValorProbEspera*10,'Probabil
idad de espera',clMaroon)
        else
```

```
GraficaBurbujasBubbleSeries1.AddXY(50,300,ValorProbEspera*10,'Probabil
idad de espera',$009595FF);
     if ASA>15 then

GraficaBurbujasBubbleSeries1.AddXY(300,50,ValorProbEspera*10,'ASA',clB
lue)
        else

GraficaBurbujasBubbleSeries1.AddXY(300,50,ValorProbEspera*10,'ASA',clS
kyBlue);
     if 100*PEMAWT<17 then

GraficaBurbujasBubbleSeries1.AddXY(50,50,100*PEMAWT,'PEMAWT',clBlack)
        else

GraficaBurbujasBubbleSeries1.AddXY(50,50,100*PEMAWT,'PEMAWT',clScrollB
ar);
     end;
end;

//
//                        Pestaña parámetros de Call Center
//
procedure TFormPrincipal.HojaCallCenterShow(Sender: TObject);
var
  Lambda:real;
  NumOper:integer;
  ProbEspera:real;
  IntTrafico:real;
  RatioOcupOperador:real;
  DuracionMediaEnSegundos:real;
  TotalSegundos:integer;
begin
  // Gráfica Call Center
  ValorNumOperadores.Value:=0;
  GraficaCallCenterLineSeries1.Clear;
  GraficaCallCenterLineSeries2.Clear;
  CalculaTiempoTotal(Linea[1].FechaLlamada,Linea[1].HoraLlamada,
    Linea[UltimoRegistroEntrante].FechaLlamada,
    Linea[UltimoRegistroEntrante].HoraLlamada, TotalSegundos);
  Lambda:=NumEntrantes/TotalSegundos;// Cálculo de Lambda
  DuracionMediaEnSegundos:=0;
  CalculaDuracionMedia (DuracionMediaEnSegundos);
  IntTrafico:=Lambda*DuracionMediaEnSegundos;
  for NumOper:=1 to 50 do
      begin
      RatioOcupOperador:=100*IntTrafico/NumOper;
      ProbEspera:=100*ErlangC(NumOper,IntTrafico);
      GraficaCallCenterLineSeries1.AddXY(NumOper,RatioOcupOperador);
      GraficaCallCenterLineSeries2.AddXY(NumOper,ProbEspera);
      end;
end;

//
//                        Gráfica de Linea de Tiempos
//
procedure TFormPrincipal.HojaLineaTiemposShow(Sender: TObject);
var
```

```
  contador:integer;
  Duracion:integer;
  Sumatoria:real;
  TiempoMedioLlamadas:real;
  Desviacion:real;
begin
  ImagenHagaZoom.Visible:=true;
  contador:=1;
  Sumatoria:=0;
  CalculaDuracionMedia(TiempoMedioLlamadas);
  GraficaLineaTiempos.BottomAxis.Title.Visible:=false;
  GraficaLineaTiemposBarSeries1.Clear;
  GraficaDuracionMedia1.Clear;
  GraficaDesviacionEstandar2.Clear;
  for contador:=1 to NumEntrantes do
      begin // Calcula duración
      Duracion:=strtoint(Linea[contador].DuracionLlamada[7..8])+
      strtoint(Linea[contador].DuracionLlamada[4..5])*60+
      strtoint(Linea[contador].DuracionLlamada[1..2])*3600;
      If (Duracion<>0) then   // Representa llamadas
         GraficaLineaTiemposBarSeries1.addxy(contador, Duracion,
         Linea[contador].FechaLlamada, clGreen)
      else                     // Representa perdidas
        GraficaLineaTiemposBarSeries1.addxy(contador, Duracion,
        Linea[contador].FechaLlamada, clRed);
      Sumatoria:=Sumatoria+sqr(Duracion-TiempoMedioLlamadas);
      end;
  Desviacion:=sqrt(Sumatoria/(NumEntrantes-1));
  ValorDesvEstLlamadas.Caption:=floattostr(Desviacion)[1..6];
  ValorDesvEstLlamadas.Visible:=true; // Calcula y muestra desviación
estándar
  for contador:=1 to NumEntrantes do
      begin

GraficaDuracionMedia1.AddXY(contador,TiempoMedioLlamadas,Linea[contado
r].HoraLlamada);

GraficaDesviacionEstandar2.AddXY(contador,Desviacion,Linea[contador].H
oraLlamada);
      end;
  CalendarioInicio.Date:=strtodate(Linea[1].FechaLlamada);
  CalendarioInicio.Time:=strtotime(Linea[1].HoraLlamada);
  CalendarioFinal.Date:=strtodate(Linea[NumEntrantes].FechaLlamada);
  CalendarioFinal.Time:=strtotime(Linea[NumEntrantes].HoraLlamada);
  CalendarioInicio.MinDate:=strtodate(Linea[1].FechaLlamada);

CalendarioInicio.MaxDate:=strtodate(Linea[NumEntrantes].FechaLlamada);
  CalendarioFinal.MinDate:=strtodate(Linea[1].FechaLlamada);

CalendarioFinal.MaxDate:=strtodate(Linea[NumEntrantes].FechaLlamada);
end;

//
//                      Gráfica de Timbre
//
procedure TFormPrincipal.HojaTimbreShow(Sender: TObject);
var
  contador:integer;
  TiempoMedioTimbre:real;
```

```
  Desviacion:real;
  Sumatoria:real;
begin
  ImagenHagaZoom.Visible:=true;
  GraficaTimbreLineSeries1.Clear;
  GraficaMediaTimbreLineSeries2.Clear;
  GraficaDesvTimbreLineSeries2.Clear;
  contador:=1;              // Inicializo variables
  TiempoMedioTimbre:=0;
  CalculaTimbreMedio(TiempoMedioTimbre);
  Sumatoria:=0;
  for contador:=1 to NumEntrantes do
      begin            // Simulatanea las 2 gráficas (Timbre y media)
      GraficaTimbreLineSeries1.AddXY(contador,
Linea[contador].TimbreLlamada,
         Linea[contador].HoraLlamada, clGreen);
      GraficaMediaTimbreLineSeries2.AddXY(contador, TiempoMedioTimbre,
         Linea[contador].HoraLlamada, clRed);
      Sumatoria:=Sumatoria+sqr(Linea[contador].TimbreLlamada-
TiempoMedioTimbre)
      end;
  // Mostrar desviación estándar
  Desviacion:=sqrt(Sumatoria/(NumEntrantes-1));
  ValorDesviacionEstandar.Caption:=floattostr(Desviacion)[1..6];
  ValorDesviacionEstandar.Visible:=true; // Calcula y muestra
desviación estándar
  for contador:=1 to NumEntrantes do

GraficaDesvTimbreLineSeries2.AddXY(contador,Desviacion,Linea[contador]
.HoraLlamada);
  CalendarioInicioT.Date:=strtodate(Linea[1].FechaLlamada);
  CalendarioInicioT.Time:=strtotime(Linea[1].HoraLlamada);
  CalendarioFinalT.Date:=strtodate(Linea[NumEntrantes].FechaLlamada);
  CalendarioFinalT.Time:=strtotime(Linea[NumEntrantes].HoraLlamada);
  CalendarioInicioT.MinDate:=strtodate(Linea[1].FechaLlamada);

CalendarioInicioT.MaxDate:=strtodate(Linea[NumEntrantes].FechaLlamada)
;
  CalendarioFinalT.MinDate:=strtodate(Linea[1].FechaLlamada);

CalendarioFinalT.MaxDate:=strtodate(Linea[NumEntrantes].FechaLlamada);
end;

//
//                      Pestaña Tramos horarios
//
procedure TFormPrincipal.HojaTramosHorariosShow(Sender: TObject);
var
  ContTramos:integer;
  ContRegistros:integer;
  Tramo:array [1..25] of integer;
  contador:integer;
begin
    ImagenHagaZoom.Visible:=false;
    TramosHorariosBarSeries1.Clear;
    ContRegistros:=1;
    ContTramos:=1;
    for contador:=1 to 25 do Tramo[contador]:=0;  // Inicializa
contadores
```

```
    for ContRegistros:=1 to NumEntrantes do       // Rastrea matriz de
entrantes
        begin
        ContTramos:=strtoint(Linea[ContRegistros].HoraLlamada[1..2]);
        Tramo[ContTramos+1]:=Tramo[ContTramos+1]+1;
        end;
    for contador:=1 to 24 do                       // Muestra gráfica
        TramosHorariosBarSeries1.AddXY(contador,
Tramo[contador],'',clMoneyGreen);
end;

//
//                                Timbre por Tramos Horarios
//
procedure TFormPrincipal.HojaTimbrePorTramosShow(Sender: TObject);
var
  ContTramos:integer;
  ContRegistros:integer;
  Tramo,Valores:array [1..25] of integer;
  Media:array [1..25] of real;
  contador:integer;
begin
  ImagenHagaZoom.Visible:=false;
  TimbrePorTramosBarSeries1.Clear;
  ContRegistros:=1;
  ContTramos:=1;
  for contador:=1 to 25 do Tramo[contador]:=0;  // Inicializa
contadores
  for contador:=1 to 25 do Valores[contador]:=0;  // Inicializa
contadores
  for ContRegistros:=1 to NumEntrantes do       // Rastrea matriz de
entrantes
      begin
      ContTramos:=strtoint(Linea[ContRegistros].HoraLlamada[1..2]);

Tramo[ContTramos+1]:=Tramo[ContTramos+1]+Linea[ContRegistros].TimbreLl
amada;
      Valores[ContTramos+1]:=Valores[ContTramos+1]+1;
      end;
  for contador:=1 to 24 do                       // Muestra gráfica
      begin
      if Valores[contador]=0 then Valores[contador]:=1;
      Media[contador]:=Tramo[contador]/Valores[contador];
      TimbrePorTramosBarSeries1.AddXY(contador,
Media[contador],'',$009191FF);
      end;
end;

//
//                          Pestaña de valores medios
//
procedure TFormPrincipal.HojaValoresMediosShow(Sender: TObject);
var
  DuracionMediaEnSegundos:real;
  TiempoMedioTimbre:real;
  TiempoMedioTimbreA:real;
  TiempoMedioTimbreP:real;
  NumPerdidas:integer;
  VPorcentajePerdidas:real;
```

```
  TotalSegundos:integer;
  NumRegistros:integer;
  NumAtendidasPorM:real;
  PorcentajeUso:real;
begin
  DuracionMediaEnSegundos:=0;// Duración media
  CalculaDuracionMedia (DuracionMediaEnSegundos);

ValorDuracionMedia.Caption:=floattostr(DuracionMediaEnSegundos)[1..6];
  ValorDuracionMedia.Visible:=true;
  TiempoMedioTimbre:=0;            // Timbres medios
  CalculaTimbreMedio (TiempoMedioTimbre);
  ValorTimbreMedio.Caption:=floattostr(TiempoMedioTimbre)[1..6];
  ValorTimbreMedio.Visible:=true;
  TiempoMedioTimbreA:=0;
  CalculaTimbreMedioA (TiempoMedioTimbreA);
  ValorTimbreMAtendidas.Caption:=floattostr(TiempoMedioTimbreA)[1..6];
  ValorTimbreMAtendidas.Visible:=true;
  TiempoMedioTimbreP:=0;
  CalculaTimbreMedioP (TiempoMedioTimbreP);
  ValorTimbreMPerdidas.Caption:=floattostr(TiempoMedioTimbreP)[1..6];
  ValorTimbreMPerdidas.Visible:=true;
//  GraficaTimbresMediosBarSeries1.BarWidthPercent:=80;
  GraficaTimbresMediosBarSeries1.Clear;

GraficaTimbresMediosBarSeries1.AddXY(1,TiempoMedioTimbreA,'Atendidas',
clMaroon);

GraficaTimbresMediosBarSeries1.AddXY(2,TiempoMedioTimbreP,'Perdidas',c
lMaroon);

GraficaTimbresMediosBarSeries1.AddXY(3,TiempoMedioTimbre,'Total',clMar
oon);
  VPorcentajePerdidas:=0;        // Perdidas
  NumPerdidas:=0;
  CalculaPerdidas(NumPerdidas, VPorcentajePerdidas);

ValorPorcentajePerdidas.Caption:=floattostr(VPorcentajePerdidas)[1..6]
;
  ValorPorcentajePerdidas.Visible:=true;
  TotalSegundos:=0;                    // Valor de uso

CalculaTiempoTotal(Linea[1].FechaLlamada,Linea[1].HoraLlamada,Linea[Ul
timoRegistroEntrante].FechaLlamada,
Linea[UltimoRegistroEntrante].HoraLlamada, TotalSegundos);
  if TotalSegundos=0 then TotalSegundos:=1;
  PorcentajeUso:=(DuracionMediaEnSegundos*NumEntrantes/TotalSegundos);
  ValorPorcentajeUso.Caption:=floattostr(PorcentajeUso)[1..6];
  ValorPorcentajeUso.Visible:=true;
  NumRegistros:=0;                 // % llamadas entrantes
  CalculaRegistros(NumRegistros);

ValorPorcentajeLlamadasEntrantes.Caption:=floattostr(100*NumEntrantes/
NumRegistros)[1..6];
  ValorPorcentajeLlamadasEntrantes.Visible:=true;// % llamadas
atendidas/cursadas

ValorPorcentajeLlamadasCursadas.Caption:=floattostr(100*NumEntrantes/N
umRegistros-VPorcentajePerdidas)[1..6];
```

```
  ValorPorcentajeLlamadasCursadas.Visible:=true;

ValorLlamadasPorHora.Caption:=floattostr(NumEntrantes*3600/TotalSegund
os)[1..6];
  ValorLlamadasPorHora.Visible:=true;// Llamadas recibidas por hora

ValorLlamadasPorMinuto.Caption:=floattostr(NumEntrantes*60/TotalSegund
os)[1..6];
  ValorLlamadasPorMinuto.Visible:=true;// Llamadas recibidas por
minuto

ValorLlamadasPorSegundo.Caption:=floattostr(NumEntrantes/TotalSegundos
)[1..6];
  ValorLlamadasPorSegundo.Visible:=true;// Llamadas recibidas por
segundo
  NumAtendidasPorM:=((NumEntrantes-Numperdidas)/(TotalSegundos/60));
  ValorAtendidasPorMinuto.Caption:=floattostr(NumAtendidasPorM)[1..6];
  ValorAtendidasPorMinuto.Visible:=true;// Atendidas por minuto
  VPorcentajePerdidas:=0;
  NumPerdidas:=0;
  GraficadeUsoPieSeries.clear;  // Gráfica de uso
  GraficadeUsoPieSeries.AddXY (0,PorcentajeUso, 'Ratio de uso: ',
clGreen);
  GraficadeUsoPieSeries.AddXY (1,100-PorcentajeUso, 'Sin tráfico: ',
$009191FF);
end;

//
//                        Opción "Analizar" del menú. Muestra de datos
Generales
//
procedure TFormPrincipal.MenuAnalisisClick(Sender: TObject);
var
   NumRegistros, NumPerdidas:integer;
   TotalSegundos:integer;
   TotalMinutos:real;
   RatioPerdidas:real;
   DuracionMediaEnSegundos:real;
   VPorcentajePerdidas:real;
begin
  PanelGeneral.ActivePageIndex:=0;
  ImagenAnalizar.Visible:=false;
  MenuAnalisis.Visible:=false;
  MenuGuardarInforme.Visible:=true;
  NumRegistros:=0;
  NumPerdidas:=0;
  TotalSegundos:=0;
  PanelGeneral.Visible:=true;                    // Visualizar ventana
de etiquetas
  ValorFechaPrimeraEntrante.caption:=Linea[1].FechaLlamada;
  ValorFechaPrimeraEntrante.Visible:=true;          // Visualizar fecha
inicio
  ValorHoraPrimeraEntrante.caption:=Linea[1].HoraLlamada;
  ValorHoraPrimeraEntrante.Visible:=true;           // Visualizar hora
inicio
  CalculaRegistros(NumRegistros);

ValorFechaUltimaEntrante.caption:=Linea[UltimoRegistroEntrante].FechaL
lamada;
```

```
  ValorFechaUltimaEntrante.Visible:=true;          // Visualizar hora
final

ValorHoraUltimaEntrante.caption:=Linea[UltimoRegistroEntrante].HoraLla
mada;
  ValorHoraUltimaEntrante.Visible:=true;          // Visualizar hora
final
  CalculaTiempoTotal(Linea[1].FechaLlamada,Linea[1].HoraLlamada,
    Linea[UltimoRegistroEntrante].FechaLlamada,
    Linea[UltimoRegistroEntrante].HoraLlamada, TotalSegundos);
  ValorTiempoTotal.Caption:=inttostr(TotalSegundos);
  ValorTiempoTotal.Visible:=true;                 // Visualizar tiempo
total
  TotalMinutos:=TotalSegundos/60;
  ValorTiempoTotalM.Caption:=floattostr(TotalMinutos)[1..6];
  ValorTiempoTotalM.Visible:=true;                 // Visualizar
tiempo total
  ValorTiempoTotalH.Caption:=floattostr(TotalMinutos/60)[1..6];
  ValorTiempoTotalH.Visible:=true;                 // Visualizar
tiempo total
  ValorNumRegistros.Caption:=inttostr(NumRegistros);
  ValorNumRegistros.Visible:=true;                // Muestra entrantes
  ValorNumLlamadasEntrantes.Caption:=inttostr(NumEntrantes);
  ValorNumLlamadasEntrantes.Visible:=true;
  RatioPerdidas:=0;
  CalculaPerdidas(NumPerdidas, RatioPerdidas);
  ValorNumLlamadasPerdidas.Caption:=inttostr(NumPerdidas);
  ValorNumLlamadasPerdidas.Visible:=true;       // Muestra perdidas
  ValorNumLlamadasAtendidas.Caption:=inttostr(NumEntrantes-
NumPerdidas);
  ValorNumLlamadasAtendidas.Visible:=true;
  NumRegistros:=0;
  GraficaEntrantesPieSeries2.Clear;  // Limpia imagen anterior, de
haberla
  GraficaEntrantesPieSeries2.AddXY(0,NumEntrantes,'Entrantes:
',clGreen);
  CalculaRegistros (NumRegistros);// Gráfico de tarta de entrantes
  GraficaEntrantesPieSeries2.AddXY(1,NumRegistros-
NumEntrantes,'Salientes e internas: ',$009191FF);
  TotalSegundos:=0;
  DuracionMediaEnSegundos:=0;

CalculaTiempoTotal(Linea[1].FechaLlamada,Linea[1].HoraLlamada,Linea[Ul
timoRegistroEntrante].FechaLlamada,
Linea[UltimoRegistroEntrante].HoraLlamada, TotalSegundos);
  if TotalSegundos=0 then TotalSegundos:=1;
  CalculaPerdidas(NumPerdidas, VPorcentajePerdidas);
  GraficaPerdidasPieSeries1.clear; //                        Gráfica de
perdidas
  GraficaPerdidasPieSeries1.AddXY (0,VPorcentajePerdidas,'Perdidas: ',
clMaroon);
  GraficaPerdidasPieSeries1.AddXY (1, 100-
VPorcentajePerdidas,'Atendidas: ',clMoneyGreen);
end;

//
//                                      Guardar fichero con informe
//
procedure TFormPrincipal.MenuGuardarInformeClick(Sender: TObject);
```

```
  Var
  FicheroReporte:textfile;
  NombreInforme:string;
  TotalSegundos:integer;
  NumRegistros:integer;
  NumPerdidas:integer;
  RatioPerdidas:real;
  DuracionMediaEnSegundos:real;
  VPorcentajePerdidas:real;
  TiempoMedioTimbre:real;
  NumEspacios:integer;
begin
   if DialogoGuardarInforme.Execute then
      begin
      NombreInforme:= DialogoGuardarInforme.FileName;
      NombreInforme:=NombreInforme+'.txt';
      Assignfile (FicheroReporte, NombreInforme);
      try
         Rewrite (FicheroReporte);
         NumEspacios:=length(NombreArchivo);
         writeln (FicheroReporte,' INFORME de datos de fichero
SMDR_____________________________________');
         writeln (FicheroReporte,'');
         writeln (FicheroReporte,' Nombre del fichero analizado:
'+NombreArchivo);
         writeln (FicheroReporte,'');
         writeln
(FicheroReporte,'╔═══════════════════════════════════════════════════
═══════════════════╗');
         writeln (FicheroReporte,'║
                                                                  ');
         writeln (FicheroReporte,'║Fecha del informe:
'+datetostr(date)+'                                          ║');
         writeln (FicheroReporte,'║
                                                                  ');
         writeln (FicheroReporte,'║Hora del informe:
'+timetostr(time)+'                                            ║');
         writeln (FicheroReporte,'║
                                                                  ');
         writeln
(FicheroReporte,'╚═══════════════════════════════════════════════════
═══════════════════╝');
         writeln (FicheroReporte,'');
         writeln (FicheroReporte,'DATOS
GENERALES:_____________________________________________________');
         writeln (FicheroReporte,'');
         writeln (FicheroReporte,'Fecha primera llamada entrante:
'+Linea[1].FechaLlamada);
         writeln (FicheroReporte,'');
         writeln (FicheroReporte,'Hora primera llamada entrante:
'+Linea[1].HoraLlamada);
         writeln (FicheroReporte,'');
         writeln (FicheroReporte,'Fecha última llamada entrante:
'+Linea[UltimoRegistroEntrante].FechaLlamada);
         writeln (FicheroReporte,'');
         writeln (FicheroReporte,'Hora última llamada entrante:
'+Linea[UltimoRegistroEntrante].HoraLlamada);
         writeln (FicheroReporte,'');
         TotalSegundos:=0;
```

```
CalculaTiempoTotal(Linea[1].FechaLlamada,Linea[1].HoraLlamada,Linea[Ul
timoRegistroEntrante].FechaLlamada,
Linea[UltimoRegistroEntrante].HoraLlamada, TotalSegundos);
         writeln (FicheroReporte,'Tiempo total (s):
'+inttostr(TotalSegundos));
         writeln (FicheroReporte,'');
         writeln (FicheroReporte,'Tiempo total (m):
'+floattostr(TotalSegundos/60)[1..6]);
         writeln (FicheroReporte,'');
         writeln (FicheroReporte,'Tiempo total (h):
'+floattostr(TotalSegundos/3600)[1..6]);
         writeln (FicheroReporte,'');
         NumRegistros:=0;
         CalculaRegistros(NumRegistros);
         writeln (FicheroReporte,'Número total de registros:
'+inttostr(NumRegistros));
         writeln (FicheroReporte,'');
         writeln (FicheroReporte,'Número de llamadas entrantes:
'+inttostr(NumEntrantes));
         writeln (FicheroReporte,'');
         NumPerdidas:=0;
         RatioPerdidas:=0;
         CalculaPerdidas(NumPerdidas, RatioPerdidas);
         writeln (FicheroReporte,'Número de llamadas perdidas:
'+inttostr(NumPerdidas));
         writeln (FicheroReporte,'');
         writeln (FicheroReporte,'Número de llamadas atendidas:
'+inttostr(NumEntrantes-NumPerdidas));
         writeln (FicheroReporte,'');
         writeln (FicheroReporte,'VALORES
MEDIOS:_____________________________________________________');
         writeln (FicheroReporte,'');
         DuracionMediaEnSegundos:=0;
         CalculaDuracionMedia (DuracionMediaEnSegundos);
         writeln (FicheroReporte,'Duración media entrantes:
'+floattostr(DuracionMediaEnSegundos)[1..5]);
         writeln (FicheroReporte,'');
         TiempoMedioTimbre:=0;
         CalculaTimbreMedio (TiempoMedioTimbre);
         writeln (FicheroReporte,'Duración timbre medio:
'+floattostr(TiempoMedioTimbre)[1..6]);
         writeln (FicheroReporte,'');
         VPorcentajePerdidas:=0;
         NumPerdidas:=0;
         CalculaPerdidas(NumPerdidas, VPorcentajePerdidas);
         writeln (FicheroReporte,'Llamadas perdidas (%):
'+floattostr(VPorcentajePerdidas)[1..6]);
         writeln (FicheroReporte,'');
         if TotalSegundos=0 then TotalSegundos:=1;
         writeln (FicheroReporte,'Valor de uso (%):
'+floattostr((DuracionMediaEnSegundos*NumEntrantes)/TotalSegundos)[1..
6]);
         writeln (FicheroReporte,'');
         writeln (FicheroReporte,'Llamadas entrantes (%):
'+floattostr(100*NumEntrantes/NumRegistros)[1..6]);
         writeln (FicheroReporte,'');
```

```
        writeln (FicheroReporte,'Llamadas atendidas (%):
'+floattostr(100*NumEntrantes/NumRegistros-
VPorcentajePerdidas)[1..6]);
        writeln (FicheroReporte,'');
        writeln (FicheroReporte,'Entrantes por hora:
'+floattostr(NumEntrantes*3600/TotalSegundos)[1..6]);
        writeln (FicheroReporte,'');
        writeln (FicheroReporte,'Entrantes por minuto:
'+floattostr(NumEntrantes*60/TotalSegundos)[1..6]);
        writeln (FicheroReporte,'');
        writeln (FicheroReporte,'Entrantes por segundo:
'+floattostr(NumEntrantes/TotalSegundos)[1..6]);
        writeln (FicheroReporte,'');
        writeln (FicheroReporte,'Atendidas por minuto:
'+floattostr((NumEntrantes-Numperdidas)/(TotalSegundos/60))[1..6]);
        writeln
(FicheroReporte,'_____________________________________________________
_________________');
        writeln (FicheroReporte,'FIN DEL INFORME.');
        finally
              CloseFile(FicheroReporte);
        end
     end;   // Fin del open dialog
end;


end.
```

Anexo 5. Aplicaciones *software* SMDR y call center.

SMDR.

AGG *Software*.

http://www.aggsoft.com/

Producto.

Esta herramienta permite la supervisión y presentación de informes proporcionando información útil para la contabilidad de llamadas. Las opciones incluyen informes de análisis de tráfico, informes de centros de llamadas, visualización de informes, departamentos, divisiones y ampliaciones.

Esta herramienta promete a las organizaciones reducir costos, aumentar la productividad, mejorar el servicio al cliente, informar sobre la productividad de los empleados y mejorar la seguridad.

Los Informes PBX pueden ser enviados de forma programada a los administradores a través de correo electrónico o almacenarlo en una unidad de red.

Empresa.

AGG *Software* está especializada en la adquisición y registro de datos y *software* de monitoreo para varias interfaces de *hardware* (RS232, RS485, USB, Ethernet). La compañía ofrece *software* profesional y herramientas que soportan estándares, protocolos, software y *hardware* para ingeniería de sistemas, fabricantes de sistemas, integradores de sistemas y desarrolladores de *hardware*.

TIM4biz.

https://www.tim4biz.com/Default.aspx

Producto.

Este *Software* SMDR recibe información sobre las llamadas telefónicas de equipos PBX (o PABX) y se permite descubrir información sobre el sistema telefónico y en última instancia, reducir los gastos de su negocio. La información generada puede indicar hay demasiadas o demasiado pocas líneas

telefónicas, si se están perdiendo llamadas antes o después de las horas regulares de oficina o si se están haciendo un número apropiado, la duración o el destino de las llamadas.

Software de *call centers*.

Luxor Technologies.

http://www.luxortec.com/

Producto

Luxor Back Office. *Software* especializado en tareas de back office, diseñado específicamente para el trabajo diario en el sector de los centros de llamadas.

Para los directivos y supervisores resulta posible definir en detalle las tareas de todo el personal, gestionar los diferentes grupos de trabajo y usar los recursos disponibles. Esto es posible porque el programa brinda información en tiempo real y de manera diaria: ofrece el nivel actual de cada Key Performance Indicators, el progreso exacto por tarea, la eficiencia de cada agente o el progreso respecto del Service Level Agreement.

Luxor Back Office reparte el trabajo entre los agentes de forma equilibrada y reorganiza continuamente las tareas a llevar a cabo teniendo en cuenta las prioridades operacionales, las habilidades del personal, los datos obtenidos en resoluciones de situaciones similares y los recursos de los que se disponga en ese preciso momento.

Luxor Back Office utiliza los sistemas BPM (Business Process Management), un requisito fundamental para permanecer en el negocio.

Evolution *software*.

http://www.evolutioncallcenter.com/

Producto.

Evolution es una solución diseñada para la automatización y gestión de los Contact Centers. La conectividad de Evolution a las centralitas telefónicas más utilizadas en la industria, y su capacidad de integración con aplicaciones y bases de datos corporativas existentes, permite proteger las inversiones

efectuadas en estas infraestructuras. Las interfaces de usuario y tecnología utilizada, basados en Microsoft Windows, aprovecha el conocimiento ya existente en la mayoría de las empresas.

Avaya.

Producto

Avaya Aura® Call Center Elite. La solución más usada para contact center a nivel mundial[46]. Aura permite manejar todo tipo de interacciones con el cliente de manera eficiente con características de direccionamiento inteligente y selección de recursos. Permite determinar si los clientes deben ser atendidos por el agente menos ocupado, el primero en estar disponible o el que tenga las habilidades que mejor se adaptan a las necesidades del mismo. Aura permite la reducción del volumen de llamadas al mudar las interacciones a canales de bajo costo como correo, chat, SMS y redes sociales. Aura permite analizar los datos históricos y de tiempo real a fin de que se pueda adaptar rápidamente el contact center a las necesidades empresariales.

Aura puede personalizar las experiencias de clientes al compartir detalles, como historial del cliente y ventana emergente de datos, en todos los canales de contacto.

Call Management System ayuda a los gerentes, supervisores y agentes analizar las tendencias del cliente, establecer comparaciones de rendimiento y planear campañas de marketing y atención al cliente que se alineen con los objetivos de la empresa.

Go autodial

http://goautodial.org/

Licencia libre.

GOautodial es un sistema gestor de centro de llamadas basado en Internet con código abierto basado en CentOS. Permite instalar automáticamente las aplicaciones GOautodial (GOadmin, GOreports y GOagent) así como

[46] Según el propio fabricante.

VICIDIAL, MySQL, PHP, Asterisk, limesurvey y otra de *software* de código abierto para tener un sistema de call center completamente funcional.
Sus características principales son Inbound, Outbound y gestión de llamadas Blended, Broadcast y encuesta de marcación, Aplicaciones basadas en Web, Configuraciones Asistente basado, Grabación de llamadas, Escalable a cientos de puestos, Troncales VoIP y líneas Telco estándar, Soporte para Sangoma y tarjetas Digium. Es licencia Open-Source GPLv2, sin costos de licencias de *software*.

Novasim.

http://www.novasim.com/CCProphet/
Capítulo aparte merece este fabricante de *software*, responsable de una excelente herramienta para la implementación mediante simulación de un *call center*.
ccProphet es una aplicación para la simulación y análisis de centros de contacto. El diseño de ccProphet permite crear en un PC un centro de contacto virtual que incorpore todas las dinámicas del sistema precisas y las complejidades de las operaciones reales de la empresa. Este *software* permite experimentar un número ilimitado de escenarios "qué pasaría si" y ver cómo las estrategias alternativas podrían afectar a las operaciones antes de que los cambios se realicen en el mundo real. Esto se traduce en la capacidad de identificar oportunidades específicas para mejorar las operaciones del centro de contacto sin tener que gastar una fortuna en el proceso de examen, y sin que finalmente se den errores costosos.
Esta herramienta integra una avanzada simulación de eventos discretos dentro de una interfaz fácil de usar, minimizando la curva de aprendizaje habitual en el uso de *software* de simulación, con la necesidad de contratar expertos en simulación caros con el fin de obtener los beneficios notables de esta tecnología.

Anexo 6. Instalación y configuración básica de IP Office Manager.

Instalación.

El proceso de instalación del gestor de la centralita IP Office de Avaya es un proceso bastante intuitivo que se realiza en buena parte seleccionando opciones por defecto para obtener una versión bastante completa para un funcionamiento básico.
El proceso empieza por la elección del idioma de la versión (fig. 6.1).

Figura 6.1. Pantalla de inicio.

En el siguiente paso se empiezan a descomprimir los primeros ficheros para un proceso guiado (fig. 6.2).

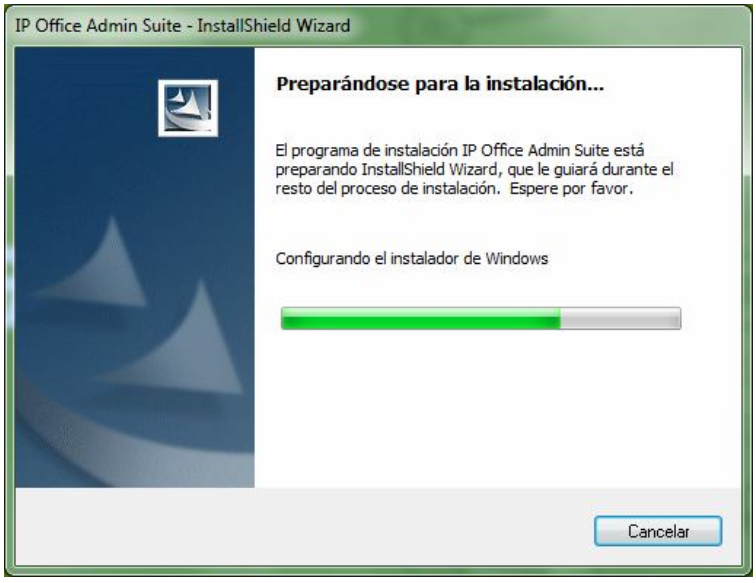


Figura 6.2. Descompresión de ficheros.

La advertencia avisa de la protección legal del software (fig. 6.3).

Figura 6.3. Advertencia de protección legal del software.

Como es habitual en cualquier instalación bajo el sistema operativo Windows, se nos pregunta por la carpeta. En este caso elegimos la definida por defecto (fig. 6.4).

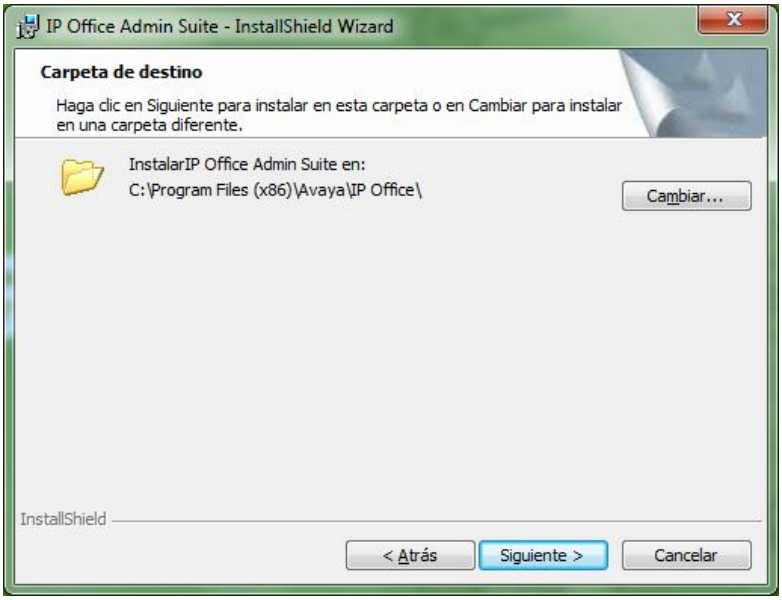


Figura 6.4. Ubicación de ficheros.

El siguiente paso nos permite seleccionar los componentes a instalar (fig. 6.5). A saber: *System Monitor, Manager, System Status Application* y *Call Status.*

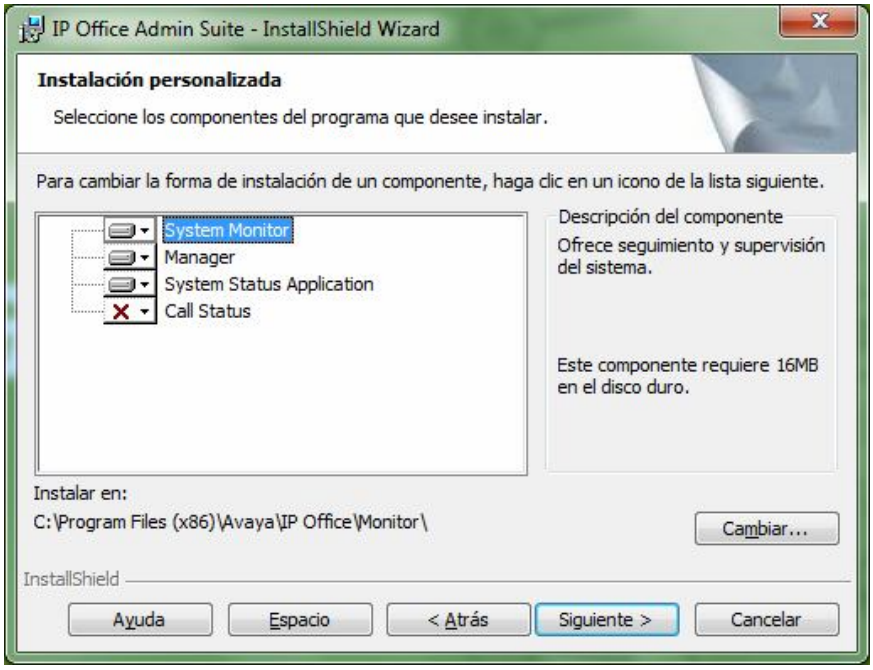


Figura 6.5. Selección de componentes.

Seguidamente pasa a instalar los componentes seleccionados (fig. 6.6). Este proceso toma un tiempo considerable (fig. 6.7).

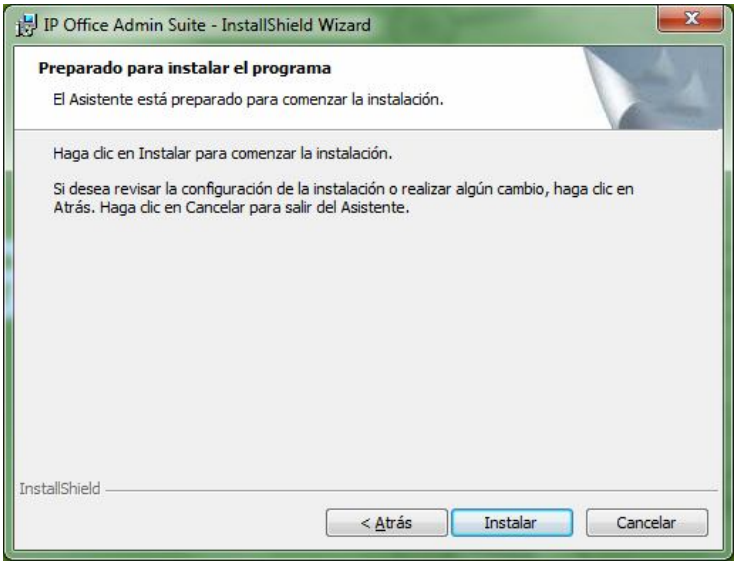


Figura 6.6. Instalación de componentes.

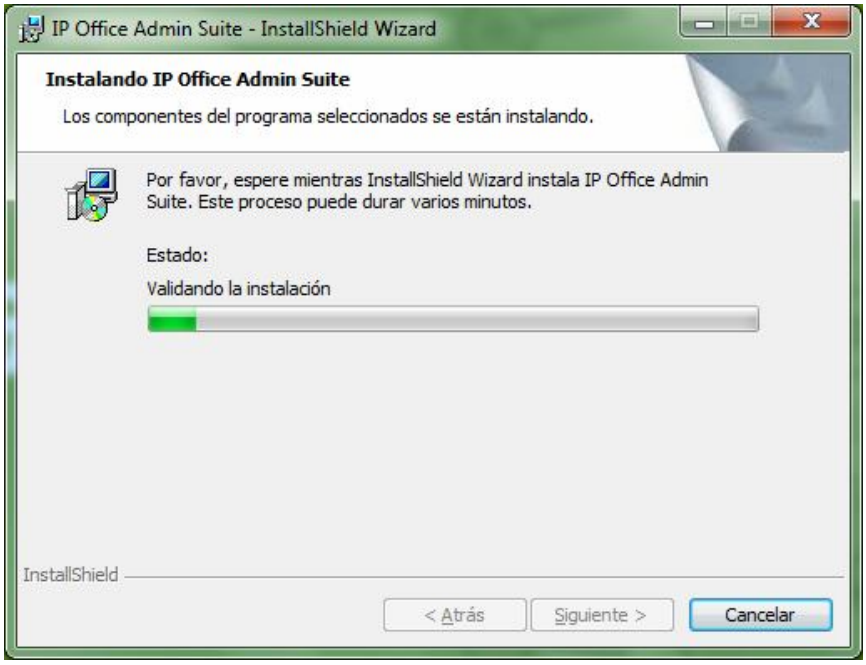


Figura 6.7. Proceso de instalación.

Una ventana de DOS permite hacer un seguimiento a los paquetes instalados. Una vez llegados a este punto, la aplicación está instalada, permitiéndonos lanzarla para su ejecución (fig. 6.8).

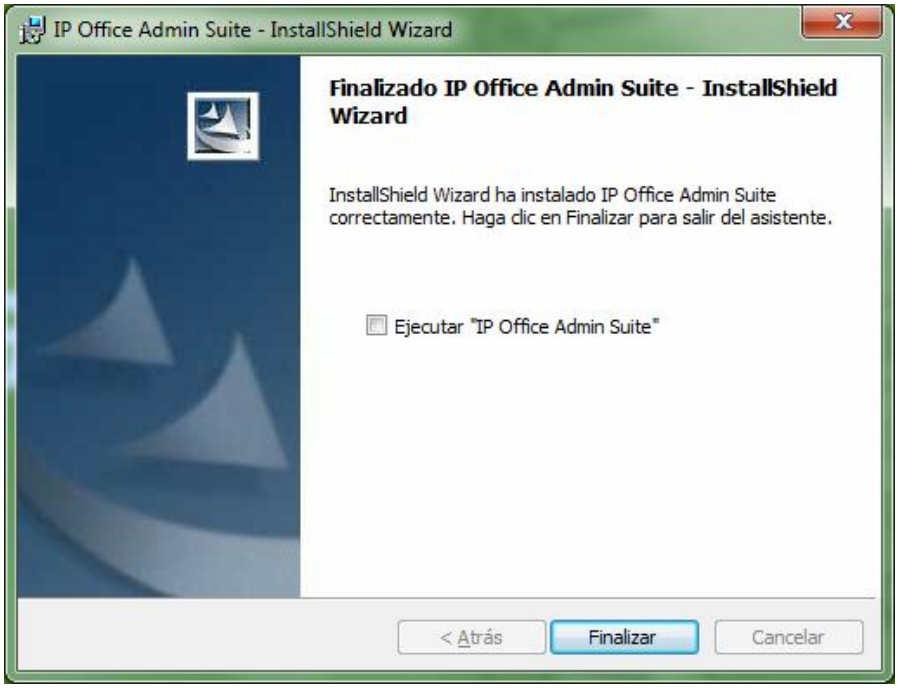


Figura 6.8. Finalización.

De esta forma queda instalada en un PC la aplicación para gestión y administración de la centralita Avaya IP Office.

Configuración básica.

Para disponer de las características necesarias para la gestión de nuestra centralita, debemos disponer la configuración básica de nuestro equipo.
Al lanzar por primera vez el gestor nos aparece la pantalla según muestra la figura 6.9.

Figura 6.9. Inicio del gestor.

A la que sigue una pantalla de configuración de nuestro sistema (fig. 6.10).

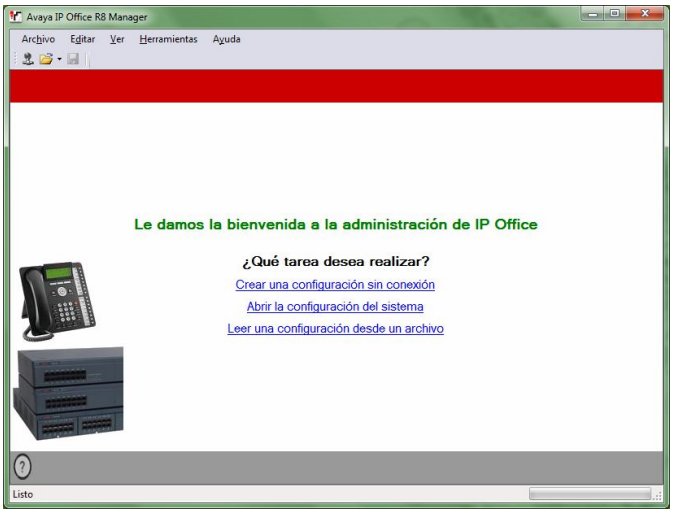


Figura 6.10. Configuración.

En nuestro caso, aprovecharemos la configuración básica del laboratorio de Transmisión por Línea (fig. 6.11) seleccionando la tercera opción.

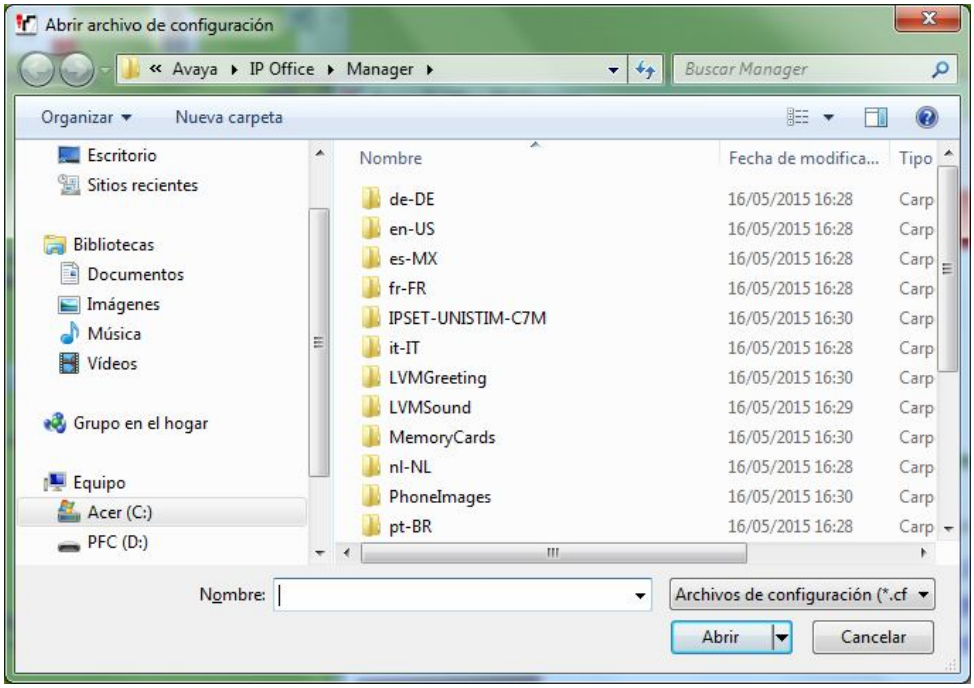


Figura 6.11. Selección de carpeta con fichero de configuración.

Llegados a este punto disponemos del panel de la figura 6.12.

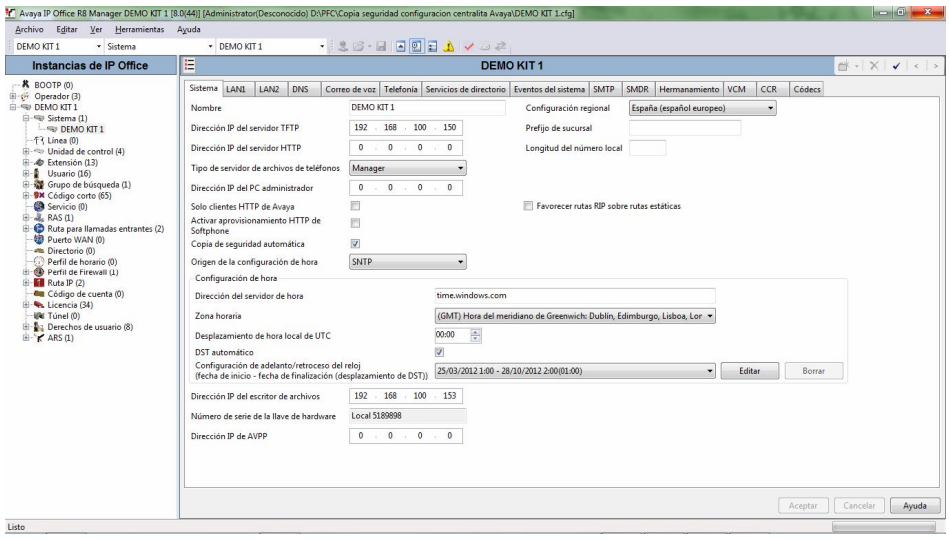


Figura 6.12. Panel del gestor.

Seleccionamos en el panel de la izquierda nuestra central, denominada Demo Kit 1 (fig. 6.13). Hay que recordar que esta centralita es un kit de demostración que trae incluidas todas las funcionalidades que, de otra forma, deben pagarse por separado.

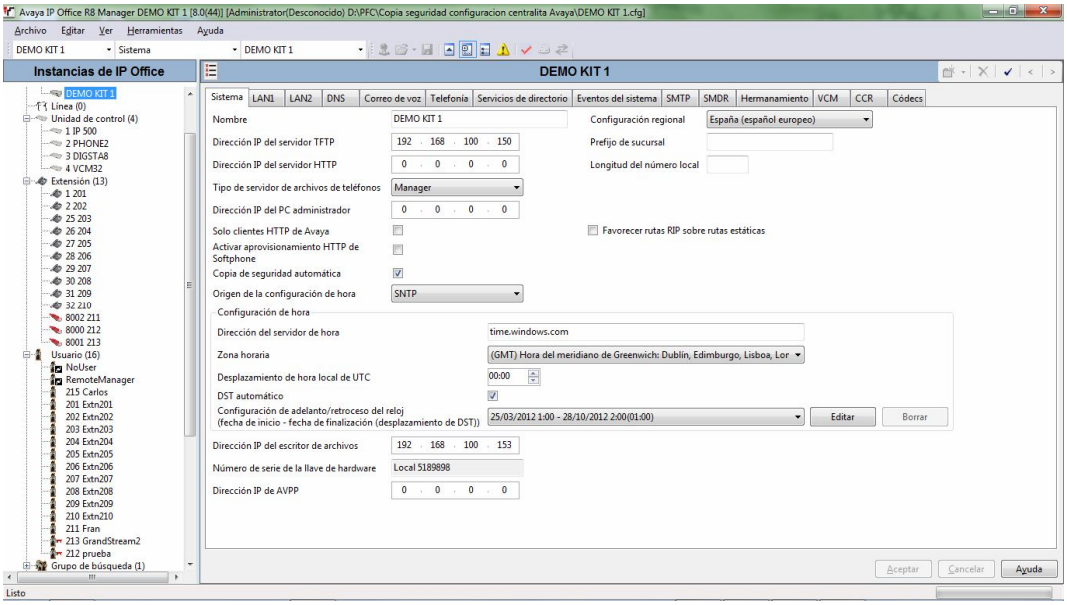


Figura 6.13. Selección de centralita.

A obtención de ficheros SMDR se hace seleccionando la pestaña correspondiente del panel de la derecha (fig. 6.14). Seleccionaremos en el combo la opción “Solo SMDR”.

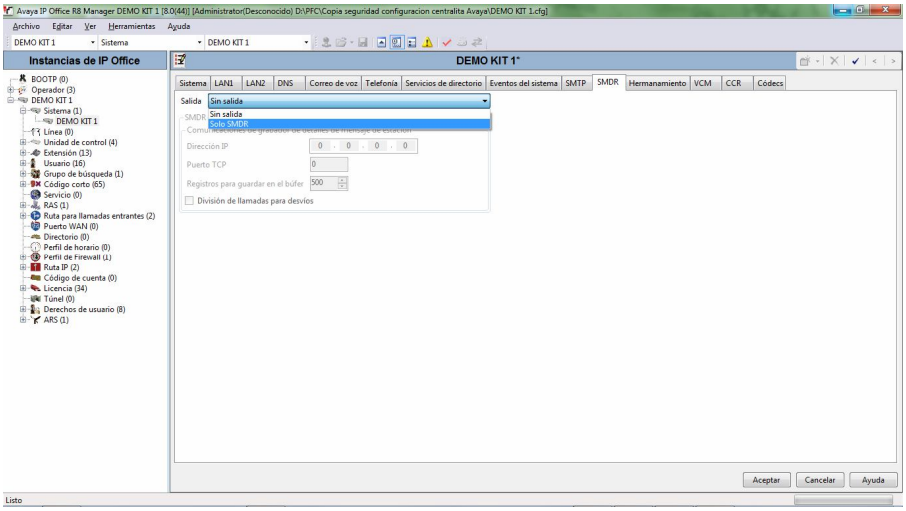


Figura 6.14. Selección de salida SMDR.

De esta forma nos permitirá grabar los datos SMDR en un dispositivo (normalmente un PC) del que debemos indicar su dirección IP (fig. 6.15), el puerto de comunicaciones y el número de registros que puede almacenar el buffer. También podemos seleccionar que se dividan las llamadas desviadas (algo que a efectos de nuestra aplicación no tiene consecuencias prácticas).

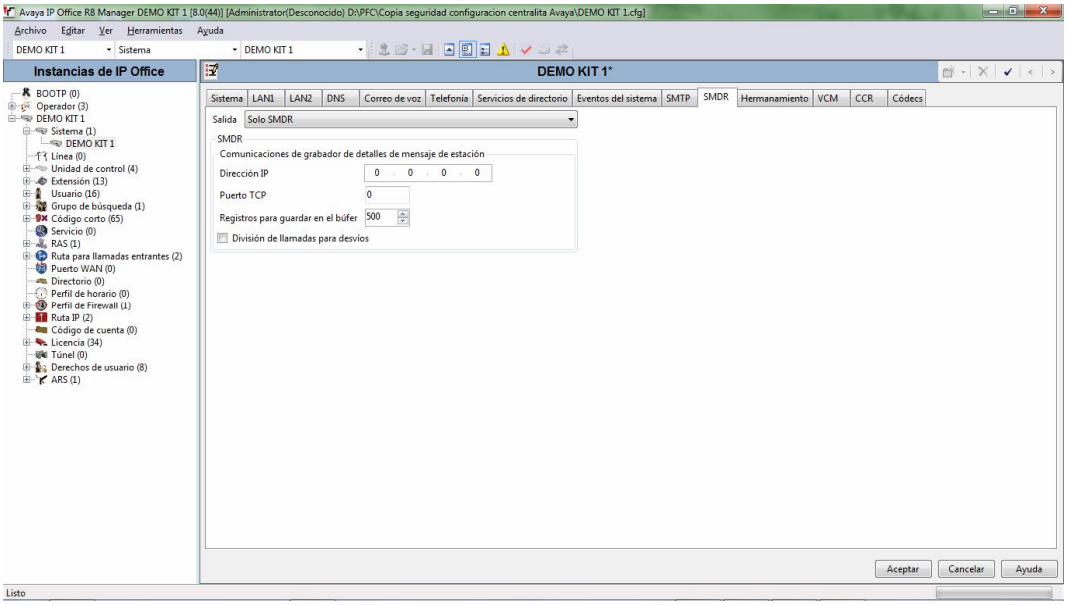


Figura 6.15. Estado de selección.

De esta forma disponemos de un conjunto de funcionalidades bastante amplio en nuestra Avaya IP Office 500.