Trabajo Fin de Grado

Grado en Ingeniería de Tecnologías de Telecomunicación

Simulador de señalización para redes de conmutación de circuitos

Autor: Ana María Escorza Aguilar

Tutor: Juan Manuel Vozmediano Torres

Departamento de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2014

Trabajo Fin de Grado
Grado en Ingeniería de Tecnologías de Telecomunicación

Simulador de señalización para redes de conmutación de circuitos

Autor:

Ana María Escorza Aguilar

Tutor:

Juan Manuel Vozmediano Torres

Profesor titular

Departamento de Ingeniería Telemática

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2014

Proyecto Fin de Carrera: Simulador de señalización para redes de conmutación de circuitos

Autor: Ana María Escorza Aguilar
Tutor: Juan Manuel Vozmediano Torres

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2014

El Secretario del Tribunal

A mi familia

A mis maestros

AGRADECIMIENTOS

Después de recorrer un duro camino, a lo largo del cual he tenido la oportunidad de madurar y adquirir una formación necesaria, aquí me veo escribiendo los agradecimientos en mi proyecto fin de grado.

En general, agradecer a todos los que directa o indirectamente han contribuido en este trabajo.

En primer lugar, a mi tutor, Juan Manuel Vozmediano Torres, por la oportunidad que me ha brindado para realizar este proyecto y aprender de él. Sus directrices, críticas y sugerencias, que han contribuido a la elaboración y mejora de este trabajo. Siempre tan exigente conmigo y consigo mismo.

En segundo lugar, al departamento de telemática y todo su profesorado, por lo mucho que he aprendido con ellos.

Conjuntamente, a la escuela y todos los que forman parte de ella, que hacen que todo siga adelante.

También, a mis padres que siempre saben orientarme y sus consejos siempre ayudan. Siempre con tanta paciencia. Por supuesto, al resto de mi familia, mi hermana, que la quiero con locura, mi abuela, mis tíos,… por vuestra confianza en mí, siempre apoyándome.

Finalmente, a Tomás porque tu apoyo durante este año ha sido muy importante para mí.

A todos, muchísimas gracias.

RESUMEN

Toda llamada telefónica requiere de un modelo de señalización para que ésta sea establecida y mantenida. El envío del número telefónico, el tono de llamada o de ocupado y la información del número al que se llama son algunos ejemplos. SS7 se ha convertido en el estándar internacional de señalización en telefonía.

En este proyecto hemos diseñado un software didáctico, con interfaz gráfica, que simula el comportamiento de SS7, cuando se realiza una llamada, de cualquier escenario definido por el usuario mediante un archivo de configuración. El cliente elegirá desde donde a donde quiere que se realice la llamada y, seguidamente, se mostrará un diagrama de paso de mensajes de establecimiento de llamadas. De la misma manera, para la liberación.

El objetivo es mejorar la enseñanza-aprendizaje de manera autónoma de este sistema. Para desarrollarlo he utilizado el lenguaje imperativo orientado a objetos C++ y la biblioteca multiplataforma Qt para la interfaz gráfica.

ABSTRACT

Every phone call requires a signaling model for it to be established and maintained. Sending the phone number, ringtone or busy information and the called number are examples. SS7 has become the international standard telephony signaling.

In this project we designed an educational software, with graphic interface, which simulates the behavior of SS7, when a call from any stage by using a user-defined configuration file is performed. The client will choose from where to where you want the call is made and then a diagram of message passing is shown.

The objective is to improve teaching and learning independently of this system. I used to develop it the imperative object-oriented language C + + and Qt library for GUI.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

1.1 Presentado SS7Simulator

SS7Simulator es un software didáctico que simula el paso de los mensajes en una llamada del sistema de señalización por canal común SS7[1] en un escenario. Esta red es definida por el usuario en un archivo de configuración, según unas directivas, y puede cambiarlas en cualquier momento recargando este archivo. Además, nos muestra una representación gráfica del escenario definido, para tener una visión más clara.

Ilustración 1 - Icono del programa

1.2 Objetivos del Proyecto

El objetivo del proyecto es desarrollar un programa educativo con el fin de ayudar en el proceso de enseñanza del estándar de señalización SS7, en la asignatura de "estructuras y protocolos de redes públicas" de 2º de Grado de Telecomunicación, o en cualquier otra materia.

Debe tener una interfaz clara e intuitiva para el usuario, y exista un retroalimentación entre quien lo utiliza y éste.

1.3 Motivación Personal

Al finalizar la universidad los alumnos deben enfrentarse al mundo laboral, por ello, durante su etapa en la titulación escogida se le otorgará una serie de conocimientos y habilidades que le serán útil en el mercado laboral.

Mi motivación es poner en práctica todos estas competencias adquiridas y poder hacerle frente de forma individual en tecnologías no vistas durante el grado, como hacer frente a un proyecto software desde cero, el uso del potente lenguaje C++ o la biblioteca multiplataforma Qt.

[1] Sistema de Señalización por Canal Común nº 7

Además, son tecnologías que se siguen utilizando mucho a día de hoy, sitios como Adobe Photoshop Album, Skype (1), Walt Disney Animation Studios, Samsung (2), Google Earth, Volvo (3),…y muchos más utilizan Qt. Asimismo,en su última versión contiene APIS para plataformas móviles (android, iOS y blackberry) (2) para el diseño de aplicaciones móviles cuyo uso está creciendo tan exponencialmente.

1.4 ESQUEMA DE LA DOCUMENTACIÓN

En esta sección se planificará la estructura de la memoria. Esto simplificará las labores de navegación a través del documento y ayudará a situar los distintos apartados. La memoria está dividida en bloques que se describirán brevemente a continuación:

Bloque 1 Introducción:

Visto hasta ahora, donde hemos presentado el proyecto, sus objetivos y la motivación personal.

Bloque 2 Fundamento teórico:

En esta sección se orientará al lector a los distintos conceptos que se deberán de conocer antes de empezar con el resto del documento. Asimismo, se tratará de describir brevemente la situación del contexto actual.

Bloque 3 Recursos y herramientas:

En esta sección se presentarán los recursos, tanto tecnológicos como de personal, con los que se contará para llevar a cabo la realización del proyecto.

Bloque 4 Requisitos del sistema:

En este apartado se describirán las funcionalidades que deberá presentar el sistema final. Dicha descripción del sistema se dividirá en objetivos, requisitos de información, requisitos funcionales y requisitos no funcionales.

Bloque 5 Análisis del sistema:

En esta sección se profundizará en los detalles más importantes del sistema, tales como los tipos que modelan la situación que se desarrolla y sus interrelaciones, las funcionalidades que el sistema debe ofrecer al usuario final y las operaciones que el sistema debe realizar para poder implementar los casos de uso.

Bloque 6 Diseño del sistema:

En este apartado se trazarán los medios por los cuales se llevarán a cabo las funcionalidades descritas en el apartado de análisis.

Bloque 7 Implementación:

En este bloque se muestran brevemente algunas de las soluciones que se han ideado durante la etapa de implementación del proyecto.

Bloque 8 Pruebas:

A lo largo de este bloque se describirán todas las pruebas que se han realizado al sistema para validar los requisitos del proyecto.

Bloque 9 Planificación temporal.

Donde se expone la estimación temporal del proyecto.

Bloque 10 Conclusiones:

En este apartado se extraerán las conclusiones finales del proyecto, y la experiencia de desarrollo del mismo. Asimismo, se estudiarán las posibles mejoras que pudiesen aplicarse al sistema.

Bloque 11 Bibliografía:

Por último, es en este apartado donde se exponen las fuentes consultadas a lo largo del desarrollo del mismo, necesarias para la documentación.

Anexos:

Definiremos como obtener un ejecutable para los distintos sistemas operativos y, además, un manual de usuario, explicando el uso del programa y la definición del archivo de configuración. Asimismo, se expondrán dos ejemplos prácticos.

2 FUNDAMENTO TEÓRICO

2.1 Señalización

La señalización es el intercambio de información no vocal específicamente dedicada al control de las llamadas (plano C) y gestión de la red (plan C/M) entre abonado y central, central y central o entre central y centros de gestión de red. Su objetivo es controlar el camino ("circuito") dedicado en exclusiva entre los extremos de la conexión.

Existen dos tipos de señalización:

- Asociada al canal: (CAS) Es el método tradicional. Usa el mismo canal para las señales de control y la llamada propiamente dicha. Existen dos formas: intrabanda y fuera de banda.

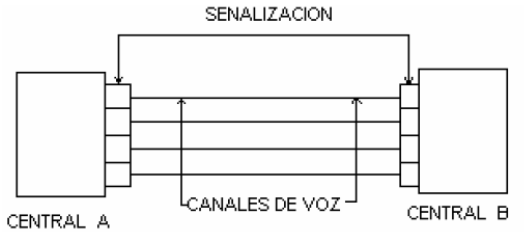


Ilustración 2 -Señalización CSS

- Señalización por canal común: (CSS) las señales de control se transmiten por rutas independientes (enlaces de señalización) de los canales de voz.

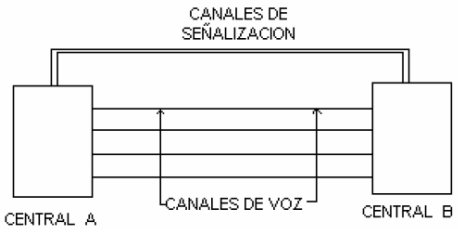


Ilustración 3 - Señalizacion CAS

Antiguamente se utilizaba la señalización asociada al canal, pero tenía un uso ineficiente de los circuitos, ya que, por ejemplo, si el destino estaba ocupado, al usar el mismo canal para las señales de control y llamada, sin importar la distancia de ambos terminales, todas las conexiones circuitales hasta el destino tenían que permanecer hechas sólo para llevar la señal de línea ocupada hasta el terminal llamante.

A causa de este uso ineficaz, las compañías telefónicas no podían afrontar la demanda existente. Sin embargo, por aquel entonces, los conceptos digitales estaban ya lo suficientemente desarrollados para que las compañías telefónicas se plantearan la posibilidad de convertir los datos analógicos en paquetes digitales y enviarlos a través de la red usando el cableado existente para un uso digital.

Un canal determinado o circuito individual se usaba sólo para transportar conversación y señalización perteneciente a una llamada cada vez. Un paquete digital podía compartir un canal común con cientos o miles de otros paquetes digitales. Así pues, miles de señales podían compartir un único canal y solamente se perdía un circuito de voz para transportar la señalización de millones de circuitos de voz. Este aprovechamiento es lo que se conoce como "Señalización por Canal Común".

Así, las ventajas de CSS frente a CAS, entre muchas, son que tenemos un menor retardo en el establecimiento llamada, número de mensajes prácticamente ilimitados, una mayor flexibilidad para nuevos servicios, un encaminamiento alternativo, un sistema de corrección de errores mediante retransmisión de tramas…

Respecto a la señalización por canal común existen tres tipos:

- Modo asociado: es aquel en el que los mensajes relativos a la comunicación siguen un camino paralelo a los de voz. (Figura 5)
- Modo disociado: utiliza un camino diferente que el de la voz
- Modo cuasi-asociado: caso particular de no asociado en el que el trayecto de los mensajes está predeterminado y es fijo en un momento dado. Además, todos los mensajes de señalización correspondientes a la misma conexión de circuito siguen la misma ruta.

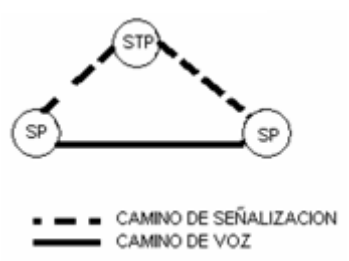


Ilustración 4 - CSS, modo asociado

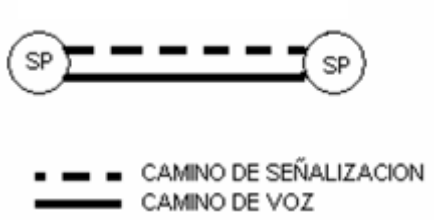


Ilustración 5 - CSS, modo cuasi-asociado

2.1.1 Estándarización.

Después de todos estos avances, todo el mundo en la industria comprendió que este sistema seria casi inútil a menos que una llamada telefónica pudiera conectar dos teléfonos de cualquier parte del mundo. Justamente, era el momento para desarrollar un estándar que estableciera la base para todos los detalles de cómo manejaría cada situación el nuevo sistema que fue conocido como SS7.

Los protocolos de este sistema fueron desarrollados por AT&T a partir de 1975 y definido como un estándar por el UIT-T2 en 1981 en la serie de Recomendaciones Q.7XX (3).

2.2 Sistema de Señalización nº 7 (CSS7)

2.2.1 Red física

Una red SS7 tiene que estar hecha de equipos capaces de soportar SS7 de terminal a terminal para proveer su funcionalidad completa. La red está hecha de muchos tipos de enlace (A, B, C, E, y F) y tres nodos de señalización – Punto de Conmutación de Servicios (SSP), Punto de Transferencia de Señal (STP), y Punto de Control de Servicio (SCP). Cada nodo es identificado en la red por un número llamado código de punto.

2.2.2 Arquitectura

SS7 tiene una parte común y una parte de usuario. La parte Común, MTP, conforma el sistema de transporte para la transferencia fiable de mensajes de señalización.

Por otro lado, la parte de usuario, inicialmente sólo tiene control de telefonía relacionada con circuitos. Pero evolucionó adaptando ciertos elementos de SS7 al modelo OSI (4) . Por ejemplo, protocolos orientados a conexión y no orientados a conexión. En la Figura 6 podemos ver la correspondencia con el modelo, pasaremos a explicar sus capas principales.

[2] Unión Internacional de las telecomunicaciones.

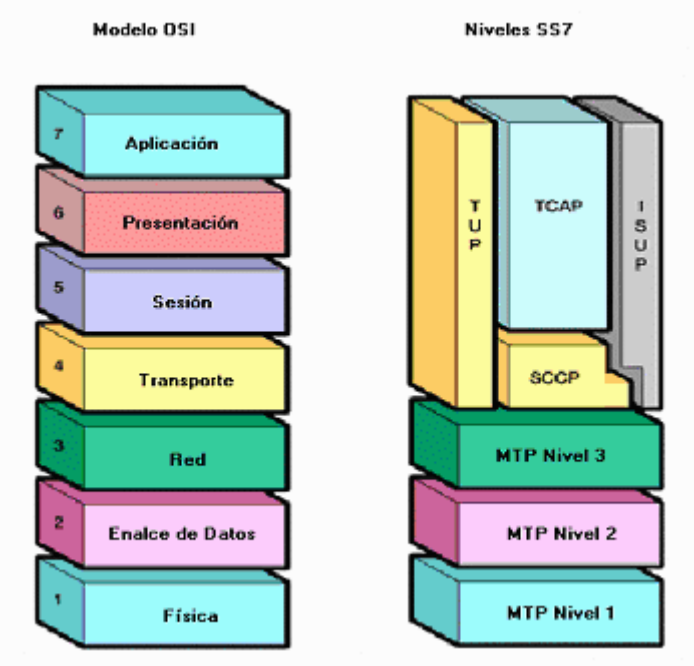


Ilustración 6 - Arquitectura SS7

2.2.2.1 MTP 1 (Enlace de datos de señalización)

Correspondiente a la Recomendación Q.702. Son las características físicas, eléctricas y funcionales del enlace de datos y los medios para acceder al mismo.

2.2.2.2 MTP 2 (Enlace de señalización)

Transferencia fiable de mensajes de señalización entre dos puntos por el enlace de datos de señalización. (Q.703)

2.2.2.3 MTP 3 (Red de señalización)

Es un nivel que ocupa el enrutamiento de los mensajes. Sus funciones son las siguientes (Q.704):

- Transferencia y procedimientos que son comunes e independientes de la operación de los enlaces de comunicación.
- Funciones de tratamiento de los mensajes de señalización.
- Funciones de gestión de la red de señalización.

2.2.2.4 SCCP

Pertenece a la parte de usuario. Hace referencia a las Recomendaciones Q.711-Q.716 de la ITU-T.

Proporciona funciones adicionales a MTP, para soporte de servicios de redes orientadas y no orientadas a conexión, también lo relacionado con la Traslación del Título Global (GTT). De igual manera, el código de punto destino se transmite al MTP efectuando después encaminamiento.

2.2.2.5 ISUP

Define los protocolos y procedimientos usados para iniciar, administrar y finalizar el tráfico de circuitos que llevan llamadas de voz y datos entre SSPs, o sea, proporciona servicios por conmutación de circuitos. (Recomendaciones Q.761-Q.769)

Debe interfuncionar con el protocolo de control de llamadas usuario-red de RDSI (Q.931).

Q.931	Facilidades de señalización por canal común usadas por el usuario de RDSI para el control de llamadas.
ISUP	Facilidades de señalización empleadas por el proveedor de la red en nombre del usuario de RDSI para proporcionarle los requisitos solicitados en el control de llamadas.

Tabla 1- ISUP frente a Q.931

2.2.2.5.1 Estructura del mensaje ISUP

Un mensaje ISUP es transmitido por el canal de señalización en una trama MSU. Esta trama tiene la primera parte de su campo SIO codificada 0101 para indicar que transporta una información que proviene del subsistema usuario ISUP.

Un mensaje ISUP está compuesto por los campos siguientes (Figura 7):

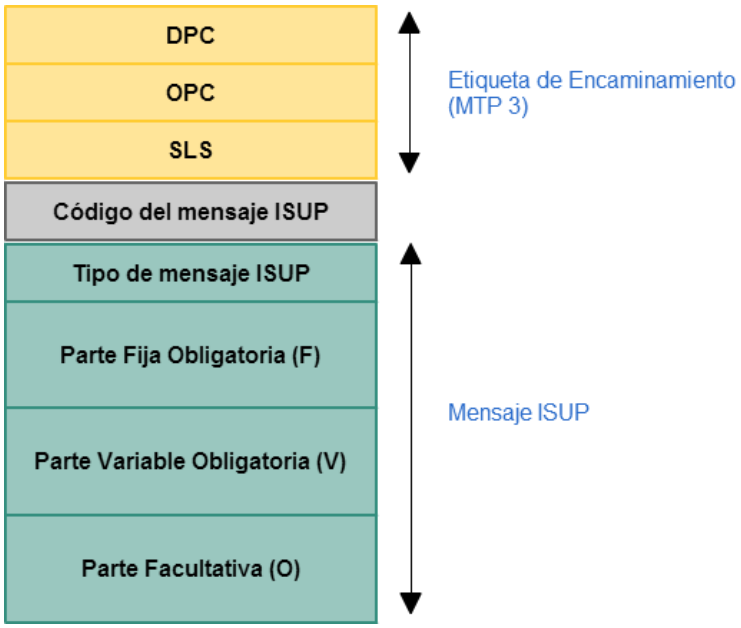


Ilustración 7 - Estructura del mensaje ISUP

- Etiqueta de encaminamiento: forma parte de la cabecera MTP3 e indica los puntos fuentes y destino del mensaje, y un código de señalización para compartir la carga entre múltiple enlaces físicos. Para cada conexión de circuito individual se debe usar la misma etiqueta de encaminamiento para todos los mensajes asociados a esa conexión.

- CIC :(Circuit Identification code) especifica el circuito con el que está relacionado el mensaje. (Conexiones n*64Kbit/s)

- Tipo de mensaje: identifica el mensaje ISUP que se envía, el contenido del resto del mensaje depende del tipo. En la Tabla 2 podemos ver definidos los mensajes esenciales.

- Una parte fija obligatoria, constituida de parámetros de longitud fija que siempre está presentes en un tipo de mensaje concreto. La longitud y el orden de los parámetros están determinados por el tipo de mensaje; por tanto, no es necesario incluir el nombre y la longitud de los parámetros.

- Una parte variable obligatoria constituida de parámetros de longitud variable de presencia obligatoria. Dada una longitud variable, se usa un puntero codificado en un único byte para indicar el principio del parámetro. Un parámetro está compuesto por el indicador de longitud del parámetro, seguido del contenido del parámetro. No es necesario indicar los nombres de los parámetros, ya que los parámetros presentes en la parte variable obligatoria están implícitamente definidos como su puntero para cada tipo de mensaje ISUP.

- Una parte facultativa que está compuesta por parámetros de longitud fija o variable.

IAM	(Initial Address Message) Mensaje hacia adelante para iniciar la toma de un circuito de salida y transmitir el número y otras informaciones relativas al encaminamiento y tratamiento de la llamada.
SAM	(Subsequent Address Message) Transporta las cifras marcadas aún no contenidas en el IAM.
ANM	(Answer Message) Mensaje hacia atrás para indicar que la llamada ha sido respondida. Inicia el cómputo de la tasación a aplicar.
CPG	(Call Progress Resume) Transporta información relativa a un evento.
ACM	(Address Complete Message) Mensaje hacia atrás que indica que se han recibido toda la información necesaria para encaminar la llamada hacia la parte llamada.
REL	(Release Message) Mensaje en uno u otro sentido para indicar que el circuito se libera (según causa) -CAUSE = 16, cuando tanto el que llama como el que es llamado concluyen la llamada. -CAUSE=17, cuando la línea del abonado destino está ocupada. -CAUSE=34, cuando el canal no está habilitado.
RLC	(Release Complete Message) Se indica fin de la interconexión de un canal útil y se confirma rececepción del REL.

Tabla 2- Mensajes esenciales ISUP

3 RECURSOS Y HERRAMIENTAS

3.1 Recursos humanos

Este proyecto está realizado únicamente por una única persona, ya que es un proyecto fin de grado, bajo la supervisión y ayuda de un tutor.

Ana María Escorza Aguilar- Alumno desarrollador del proyecto.

Juan Manuel Vozmediano Torres-Tutor del proyecto.

3.2 Recursos hardware

Entre los recursos hardware podemos encontrar los siguientes: un PC de sobremesa y otro PC portátil.

A continuación, vamos a detallar cada uno de esos recursos hardware:

3.2.1 PC sobremesa

- **Sistema operativo**: Windows 7 Ultimate SP 64 Bits
- **Procesador:** Intel Core i3 540 3.06 GHz
- **Almacenamiento:** 1 TB
- **Memoria RAM:** 8 GB
- **GPU:** NVIDIA GeForce GTX 650

3.2.2 PC portátil HP ProBook 4520s

- **Sistema operativo**: Windows 7 Ultimate SP 1, Ubuntu (32 Bits)
- **Procesador:** Intel Celeron CPU P4500 1.87Ghz
- **Almacenamiento:** 200GB
- **Memoria RAM:** 3 GB
- **GPU:** Intel Hd Graphics

3.3 Recursos software

Para llevar a cabo el proyecto será imprescindible el uso de varios recursos software que nos permitirán realizar cada una de las tareas: elaboración de diagramas, diseño de imágenes y que pueden ser de varios tipos: herramientas de desarrollo, procesador de texto.

3.3.1 Elaboración de diagramas

- **Microsoft Visio Professional 2013**: Desarrollado por Microsoft, esta herramienta nos permite crear una amplia variedad de diagramas con un estilo profesional.
- **Dia**: Herramienta de licencia libre que nos permite crear diagramas de forma sencilla.

3.3.2 Procesador de texto

- **Microsoft Word 2013**: Herramienta también perteneciente a Microsoft, nos permite redactar documentos, aplicarles estilos muchas más funciones que nos permitirá obtener documentos espectaculares

3.3.3 Diseño de imágenes

- **Adobe Ilustrator:** diseño vectorial.

3.3.4 Entorno de desarrollo y lenguaje.

Qt Creator IDE (4), versión 4, es el entorno de desarrollo utilizado para la creación del proyecto, ideado por Trolltech para el desarrollo de aplicaciones creadas a partir de las bibliotecas de Qt. Esta bajo una licencia comercial y de código abierto.

Qt Creator, viene acompañado de un conjunto de herramientas para facilitar su uso. Además, posee un avanzado editor de código C++, lenguaje utilizado en la realización de este proyecto, ya que es el lenguaje que utiliza de forma nativa. Al utilizar el lenguaje de programación C++, he podido hacer uso del potencial de la programación orientada a objetos.

La distribución interna de Qt Creator es la siguiente: (Figura 8)

- QtCore: Contiene el núcleo no gráfico de Qt.
- QtGui: Colección básica de componentes gráficos.
- QtNetwork: Clases para escribir clientes y servidores TCP/IP.
- QtOpenGL: Facilita el uso de openGL.
- QtScript: Expone las aplicaciones a scripting con un lenguaje ECMAscript.
- QtScriptTools: Es un depurador de QtScript.
- QtSQL: Integración de bases de datos.
- QtSVG: Soporte SVG.
- QtWebKit: Popular motor Web, con Qt.
- QtXml: Soporte básico de Xml.
- QtXmlPatterns: Es un motor de XQuery 1.0 y XPath 2.0 y parcialmente Xslt.
- Phonon: Framework multimedia.
- Otros: QtDessigner, QtUiTools, QtHelp, QtAssistant, QtTest, QtDBus (solo
- Unix), y a partir de Qt 4.6 QtOpenVG y QtMultimedia.

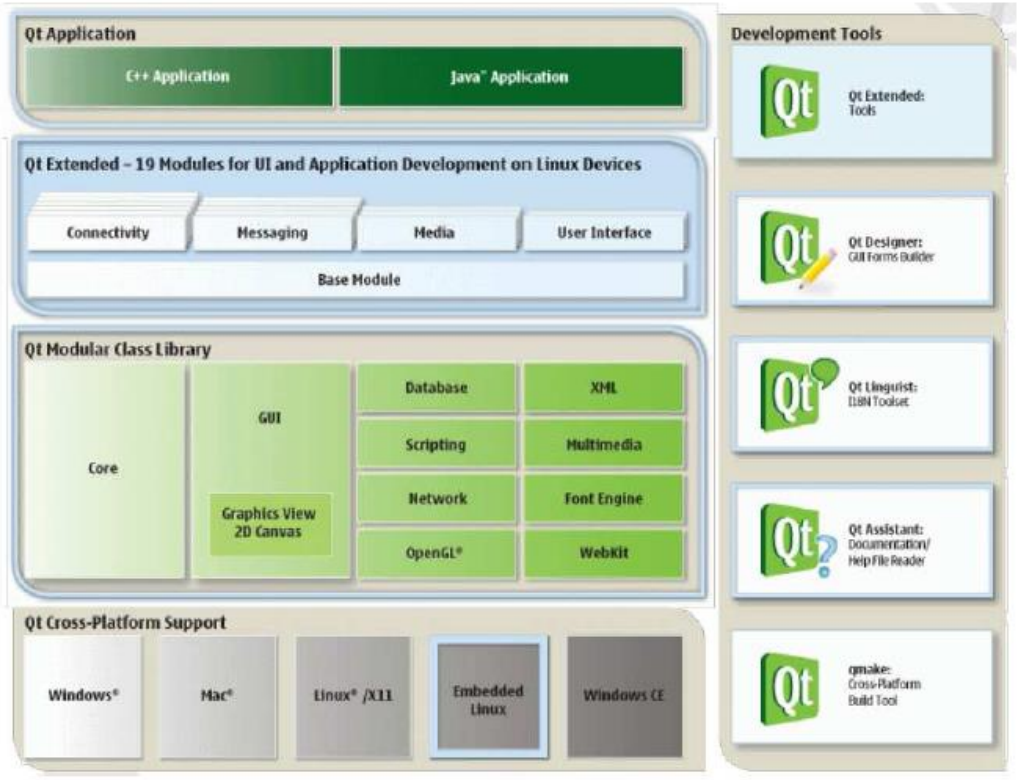


Ilustración 8 - Distribución paquetes Qt Creator

Para este proyecto se han utilizado en su mayoría las clases de las librerías QtCore y QtGUI.

En el entorno de trabajo puedes encontrar numerosos ejemplos de cómo utilizar distintas clases de Qt. Por otro lado, tiene otras muchas funcionalidades, como generar el makefile, depurar paso a paso, poner puntos de ruptura, añadir nuevas ventanas gráficas o clases automáticamente…

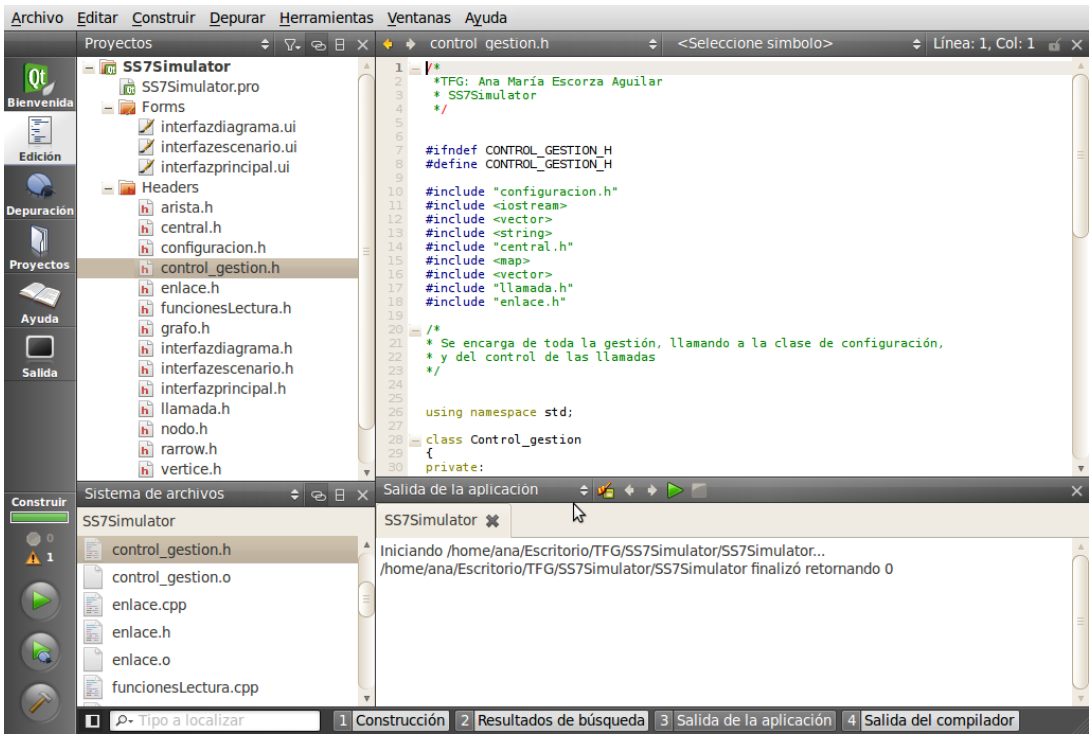


Ilustración 9 - Entorno de trabajo Qt Creator

4 REQUISITOS DEL SISTEMA

En este apartado abordaremos con detalle la idea del proyecto. Lo primero que haremos será presentar a todas las personas que participan en el proyecto y seguidamente definiremos los objetivos del sistema. Posteriormente, continuaremos con el apartado de requisitos de información, donde se detallarán todos los tipos de datos con los que trabajará el sistema. Finalmente, definiremos los requisitos tanto funcionales como los no funcionales.

4.1 OBTENCIÓN DE REQUISITOS

4.1.1 Introducción

En este paso se tratará de abordar el problema desde otro nivel, explicando en primer lugar los objetivos del sistema, necesarios para cubrir todos los requisitos actuales, y posteriormente, comenzando a detallar los primeros aspectos técnicos que surgirán durante el desarrollo.

Cabe destacar la especial importancia de este apartado, pues es cuando se deberán especificar unos objetivos acordes a las necesidades del cliente, así como se establecerán los primeros pilares para un adecuado progreso del proyecto.

4.1.2 Participantes en el proyecto

4.1.2.1 Organizaciones

Organización	Dpto. de Telemática
Dirección	Escuela Técnica Superior de Ingenieros. Camino de los Descubrimientos, s/n. . 41092 Sevilla
Teléfono	+34 954487384
Fax	+34 954487385
Comentarios	Departamento para el que desarrollaremos el proyecto

Tabla 3 - Organizaciones

4.1.2.2 Personas

Participante	Ana María Escorza Aguilar
Organización	Dpto. de Telemática
Rol	Analista – Desarrollador
Es desarrollador	Sí
Es cliente	Sí
Es usuario	No
Comentarios	Alumna de Grado en Ingeniería de Tecnologías de Telecomunicación. Se encargará de desarrollar y documentar el proyecto que estamos tratando con objetivo de superar la asignatura “Proyecto Fin de Grado”

Tabla 4 - Participante

Participante	Juan Manuel Vozmediano Torres
Organización	Dpto. de Telemática
Rol	Tutor del proyecto
Es desarrollador	No
Es cliente	Sí
Es usuario	No
Comentarios	Profesor y director del departamento de telemática. Orienta a los alumnos en el desarrollo del proyecto

Tabla 5- Participante

4.1.3 Descripción del sistema actual

Software didáctico que simula el paso de mensaje del establecimiento de llamada y liberación entre dos líneas telefónicas, del sistema estandarizado de señalización SS7.

El entorno en el que se realiza la llamada es definido por el usuario mediante un archivo de configuración. El programa también es capaz de mostrar este escenario mediante un dibujo.

4.1.4 Objetivos del sistema

Para conseguir nuestro cometido, que es realizar con éxito nuestro proyecto, debemos definir claramente cual son los objetivos que deben cumplir dicho proyecto.

OBJ-0001	Mostrar escenario
Versión	1.0 (15/10/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Descripción	El sistema deberá mostrar una representación gráfica del escenario que el usuario ha definido mediante unas directivas en el archivo de configuración.
Subobjetivos	[OBJ-0007] Leer datos archivo configuración [OBJ-0008] Gestionar los datos.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Habrá dos formas para mostrar el escenario una en el que todos los nodos están superpuestos encima del otro y el usuario los colocará como quiera y otra en el que el escenario se ordenará según un grafo circular. El escenario constará de las centrales, líneas telefónicas, enlaces de datos y enlaces de señalización.

Tabla 6 - OBJ-0001

OBJ-0002	Menú de Llamada
Versión	1.0 (15/10/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Descripción	El sistema deberá mostrar un menú de llamada, donde se puede rellenar el origen y destino de la llamada que queramos realizar.
Subobjetivos	[OBJ-0007] Leer datos archivo configuración

	[OBJ-0008] Gestionar los datos. [OBJ-0003] Listado telefónico.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Al lado se encontrará el listado telefónico para que el usuario tenga presente los teléfonos que puede realizar llamada. Se mostrará un mensaje si el destino o el origen no son correcto, o si falta algún campo.

Tabla 7 - OBJ-0002

OBJ-0003	**Mostrar listado telefónico**
Versión	1.0 (15/10/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Descripción	El sistema deberá mostrar un listado de números a los que el usuario puede llamar.
Subobjetivos	[OBJ-0007] Leer datos archivo configuración [OBJ-0008] Gestionar los datos.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	-

Tabla 8 - OBJ-0003

OBJ-0004	**Llamar**
Versión	1.0 (15/10/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Descripción	El sistema deberá mostrar el paso de mensajes de la llamada, una vez que el usuario le dé a aceptar en el menú y los teléfonos sean correctos.
Subobjetivos	[OBJ-0008] Gestionar los datos. [OBJ-0002] Menú de llamada.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Se deberá mostrar los mensajes ISUP con el cic de los circuitos tomados. Si no hay circuitos libres, una liberación de llamada.

Tabla 9 - OBJ-0004

OBJ-0005	**Colgar**
Versión	1.0 (15/10/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Descripción	El sistema mostrará un paso de mensajes de liberación de llamada, cuando el usuario decida colgarla.
Subobjetivos	[OBJ-0008] Gestionar los datos. [OBJ-0004] Llamada.
Importancia	Vital
Urgencia	Inmediatamente

Estado	Validado
Estabilidad	Alta
Comentarios	Se podrá decidir quién cuelga la llamada, si el origen o destino. Se deberán liberar los circuitos.

Tabla 10 - OBJ-0005

OBJ-0006	**Gestión de enlaces**
Versión	1.0 (15/10/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Descripción	El sistema deberá gestionar los enlaces existentes entre las centrales.
Subobjetivos	[OBJ-0008] Gestionar los datos. [OBJ-0004] Llamar. [OBJ-0005] Colgar.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Tomar o liberar circuitos según se requiera.

Tabla 11 - OBJ-0006

OBJ-0007	**Leer datos archivo de configuración**
Versión	1.0 (15/10/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Descripción	El sistema deberá leer los datos del archivo de configuración y avisar si hay errores en él.
Subobjetivos	Ninguno
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	El mensaje de error deberá mostrar el número de línea en el que se produce el error.

Tabla 12 - OBJ-0007

OBJ-0008	**Gestionar datos**
Versión	1.0 (15/10/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Descripción	El sistema deberá almacenará los datos del archivo de configuración y gestionarlos fácilmente cuando sea necesario.
Subobjetivos	[OBJ-0007] Leer datos archivo configuración
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	-

Tabla 13 - OBJ-0008

OBJ-0009	Ofrecer una interfaz de usuario sencilla y funcional
Versión	1. (15/10/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Descripción	El sistema deberá mostrar una interfaz intuitiva, amigable y sencilla.
Subobjetivos	Ninguno
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Para el diseño de la interfaz utilizaremos clases de la biblioteca multiplataforma Qt.

Tabla 14 - OBJ-0009

4.1.5 Requisitos de información

Los requisitos de información nos serán útiles gracias a que a partir de ellos identificaremos las clases que manejará la información del sistema. A continuación mostraremos los requisitos identificados:

IRQ-0001	Información Centrales (STP)
Versión	1.0 (28/10/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Dependencias	Ninguno
Descripción	El sistema deberá almacenar información de cada central definida.
Datos específicos	• Código de puntos • Prefijo • Teléfonos • Ruta de encaminamiento de enlace de datos • Ruta de encaminamiento de enlaces de señalización
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	El prefijo identificará a una o varias centrales pertenecientes a una zona determinada.

Tabla 15 - IRQ-0001

IRQ-0002	Información enlaces
Versión	1.0 (28/10/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Dependencias	Ninguno
Descripción	El sistema deberá almacenar los enlaces existentes en cada central y el número de circuitos en cada uno de ellos.
Datos específicos	• Código de puntos • Central origen • Central destino • Número de circuitos • Estado de los circuitos
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	-

Tabla 16 - IRQ-0002

IRQ-0003	Información de la llamada
Versión	1.0 (28/10/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Dependencias	Ninguno
Descripción	El sistema deberá almacenar información de cada llamada.
Datos específicos	• Origen • Destino • Ruta Señalización • Ruta de datos • CIC
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	-

Tabla 17 - IRQ-0003

4.2 REQUISITOS FUNCIONALES

Los requisitos funcionales definen el comportamiento del sistema que se está desarrollando, incluyendo los procesos fundamentales que el software llevará a cabo. La mayoría de estos requisitos funcionales provienen directamente de un requisito de usuario.

En este apartado vamos a citar los requisitos funcionales de nuestro proyecto. Estos requisitos fueron identificados en la fase de análisis. Los requisitos serán mostrados en forma de diagramas y casos de usos, ya que de esta forma tenemos unos datos más comprensibles.

Los casos de usos vendrán agrupados en distintos subsistemas según al que pertenezcan. Los subsistemas contienen los casos de usos que están relacionados entre sí, de esta forma nos será más fácil buscarlos. Hay que tener en cuenta que no todos los subsistemas tienen porque tener necesariamente casos de uso.

4.2.1 Diagrama de casos de usos

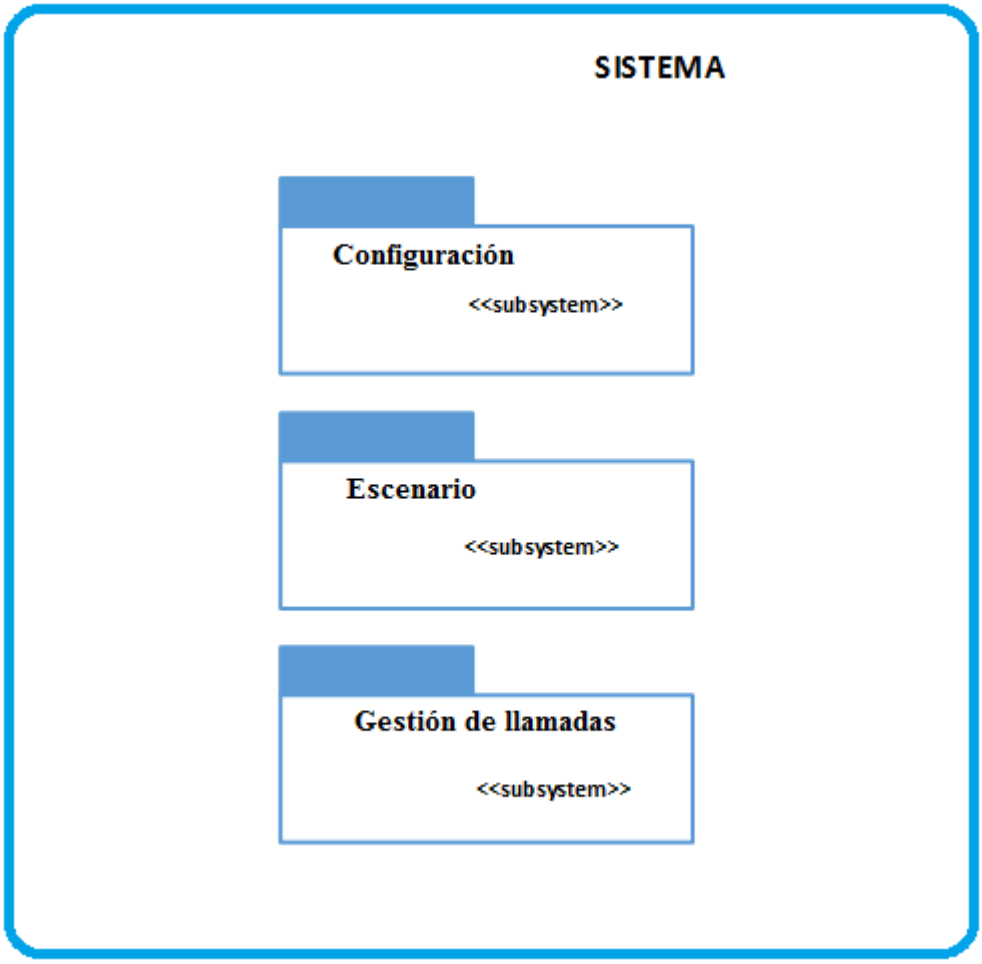


Ilustración 10 - Subsistemas

Ilustración 11 - Subsistema Configuración

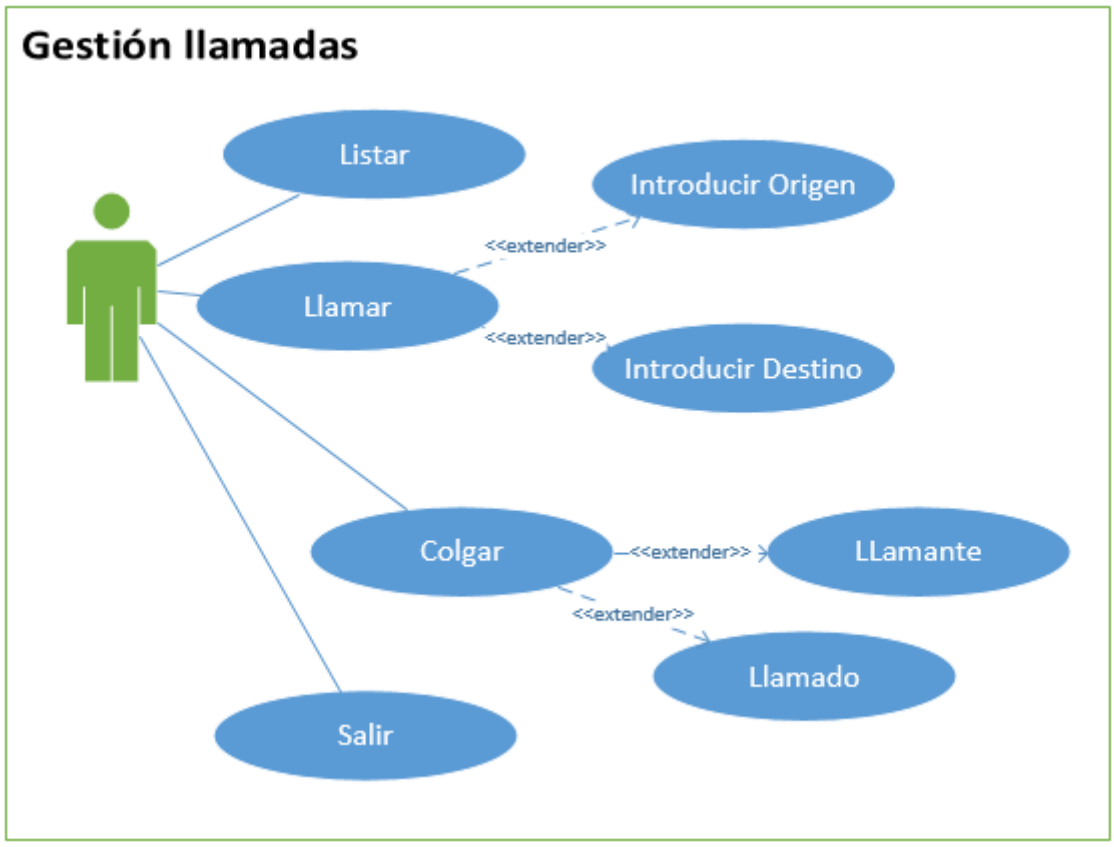


Ilustración 12 - Subsistema Gestión Llamadas

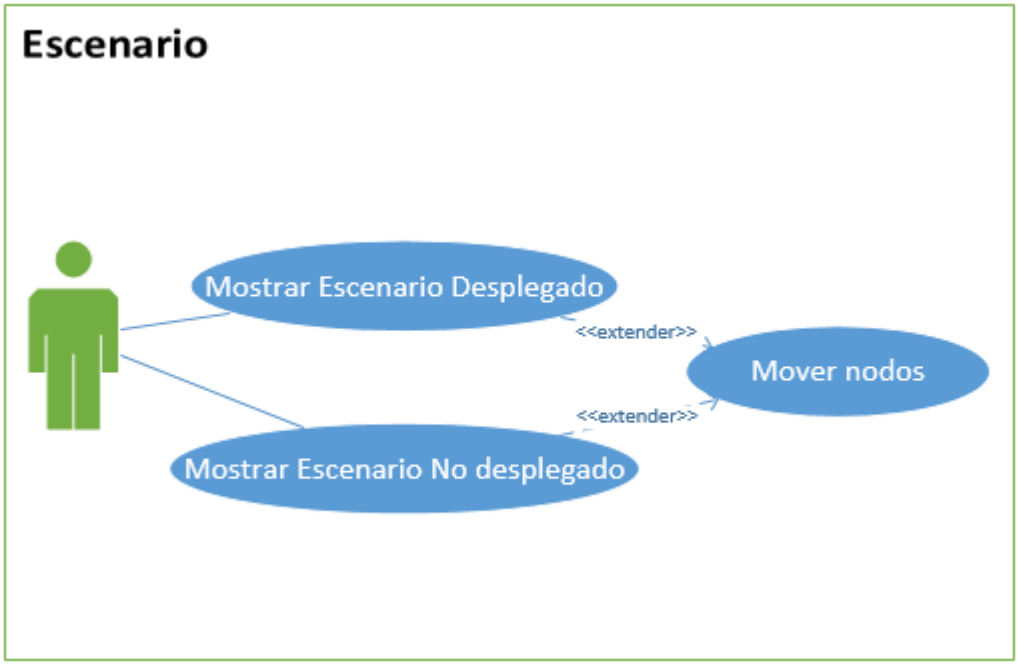


Ilustración 13 – Subsistema escenario

Los casos de uso definen el comportamiento del sistema cuando el usuario interactúa con este. En el caso de nuestro sistema tenemos un único usuario, que es el actor. Posteriormente definiremos este actor y explicaremos cada caso de uso.

4.2.2 Definición de actores

ACT-0001	Usuarios de la aplicación
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar Juan Manuel Vozmediano Torres
Descripción	Este actor representa a cualquier usuario de la aplicación
Comentarios	-

Tabla 18 - ACT-0001

4.2.3 Casos de usos del sistema

UC-0001	Escribir origen llamada, Escribir destino llamada
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Dependencias	• [OBJ-0007] Leer datos archivo de configuración • [OBJ-0008] Gestionar los datos • [OBJ-0002] Menú de llamada • [OBJ-0003] Mostrar listado telefónico. • [IRQ-0001] Información Centrales (STP)
Descripción	El usuario (ACT-0001) introducirá el origen y fin de la llamada

Precondición	-
Secuencia normal	Pasos / Acción
	1. El usuario (ACT-0001) introducirá origen llamada y destino llamada.
Postcondición	-
Excepciones	Pasos / Acción
	Si el destino o el origen no son correctos se avisará al usuario.
Importancia	Importante
Urgencia	Urgente
Estado	Validado
Estabilidad	Alta
Comentarios	-

Tabla 19 - UC-0001

UC-0002	**Llamar**
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Dependencias	• [OBJ-0002] Menú de Llamada • [OBJ-0004] Llamar • [OBJ-0006] Gestión de enlaces • [OBJ-0007] Leer datos archivo de configuración • [OBJ-0008] Gestionar los datos • [IRQ-0001] Información Centrales (STP) • [IRQ-0002] Información de Enlaces • [IRQ-0003] Información de la llamada
Descripción	El sistema mostrará al usuario el paso de mensajes de la llamada realizada.
Precondición	• Introducir el origen y fin de la llamada correcto
Secuencia normal	Pasos / Acción
	1. El usuario (ACT-0001) introducirá origen y destino de la llamada. 2. El usuario (ACT-0001) seleccionará llamar.
Postcondición	El sistema le mostrará el diagrama de paso de mensajes de la llamada.
Excepciones	Pasos / Acción
	Si no hay circuitos disponibles, se liberará la llamada. Si no se puede encaminar correctamente, se avisará al usuario.
Importancia	Importante
Urgencia	Urgente
Estado	Validado
Estabilidad	Alta
Comentarios	-

Tabla 20 - UC-0002

UC-0003	**Listar**
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Dependencias	• [OBJ-0007] Leer datos archivo de configuración • [OBJ-0008] Gestionar los datos • [IRQ-0001] Información Centrales (STP)
Descripción	El sistema permitirá ver un listado con los teléfonos existentes.

Precondición	Cargar archivo configuración.
Secuencia normal	Pasos / Acción 1. El usuario (ACT-0001) pedirá ver el listado telefónico.
Postcondición	Se mostrará el listado telefónico.
Excepciones	Pasos / Acción
Importancia	Importante
Urgencia	Urgente
Estado	Validado
Estabilidad	Alta
Comentarios	Ayudará al usuario para ver al instante los teléfonos con los que puede realizar llamadas.

Tabla 21 - UC-0003

UC-0004	**Ver Escenario Desplegado**
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Dependencias	• [OBJ-0001] Mostrar escenario • [OBJ-0006] Gestión de enlaces • [OBJ-0007] Leer datos archivo de configuración • [OBJ-0008] Gestionar los datos • [IRQ-0001] Información Centrales (STP) • [IRQ-0002] Información de Enlaces
Descripción	El sistema mostrará un esquema del escenario desplegado definido en el archivo de configuración.
Precondición	Cargar archivo configuración.
Secuencia normal	Pasos / Acción 2. El usuario (ACT-0001) pedirá muestre el escenario.
Postcondición	Se mostrará una nueva ventana con el escenario.
Excepciones	Pasos / Acción Si no es un archivo de configuración correcto no se mostrará nada.
Importancia	Importante
Urgencia	Urgente
Estado	Validado
Estabilidad	Alta
Comentarios	-

Tabla 22 - UC-0004

UC-0005	**Ver Escenario no Desplegado**
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Dependencias	• [OBJ-0001] Mostrar escenario • [OBJ-0006] Gestión de enlaces • [OBJ-0007] Leer datos archivo de configuración • [OBJ-0008] Gestionar los datos • [IRQ-0001] Información Centrales (STP) • [IRQ-0002] Información de Enlaces
Descripción	El sistema mostrará un esquema del escenario no desplegado definido en el archivo de configuración.
Precondición	Cargar archivo configuración.

Secuencia normal	Pasos / Acción
	3. El usuario (ACT-0001) pedirá muestre el escenario.
Postcondición	Se mostrará una nueva ventana con el escenario.
Excepciones	Pasos / Acción
	Si no es un archivo de configuración correcto no se mostrará nada.
Importancia	Importante
Urgencia	Urgente
Estado	Validado
Estabilidad	Alta
Comentarios	-

Tabla 23 - UC-0005

UC-0006	**Recargar archivo de configuración**
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Dependencias	• [OBJ-0007] Leer datos archivo de configuración • [OBJ-0008] Gestionar los datos • [IRQ-0001] Información Centrales (STP) • [IRQ-0002] Información de Enlaces
Descripción	El sistema cargará un nuevo archivo de configuración y se almacenaran los datos.
Precondición	Tener un archivo de configuración guardado en la carpeta del ejecutable.
Secuencia normal	Pasos / Acción
	1. El usuario (ACT-0001) introducirá nuevo archivo de configuración. 2. El usuario (ACT-0001) pedirá al sistema recargar el archivo.
Postcondición	Se almacenarán los nuevos datos.
Excepciones	Pasos / Acción
	Si hay errores en las directivas del archivo de configuración, o en los datos, se dará un mensaje de error informando de cuál es el error y el número de línea en el que se encuentra.
Importancia	Importante
Urgencia	Urgente
Estado	Validado
Estabilidad	Alta
Comentarios	-

Tabla 24 - UC-0006

UC-0007	**Colgar Llamada (Llamante, Llamado)**
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Dependencias	• [OBJ-0004] Llamar • [OBJ-0005] Colgar • [OBJ-0006] Gestión de enlaces • [IRQ-0002] Información de Enlaces • [IRQ-0003] Información de la llamada
Descripción	El sistema colgará la llamada, mostrando su respectivo diagrama paso mensajes, además se liberarán los circuitos.

Precondición	Llamar.
Secuencia normal	Pasos / Acción
	1. El usuario (ACT-0001) pedirá al sistema finalizar la llamada.
Postcondición	Se mostrará el paso de mensajes de la liberación de la llamada.
Excepciones	Pasos / Acción
	-
Importancia	Importante
Urgencia	Urgente
Estado	Validado
Estabilidad	Alta
Comentarios	Podrá elegir tanto colgar el llamante como el llamado.

Tabla 25 - UC-0007

UC-0008	**Mover nodos**
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar
Dependencias	• [OBJ-0001] Mostrar escenario • [OBJ-0007] Leer datos archivo de configuración • [OBJ-0008] Gestionar los datos • [IRQ-0001] Información Centrales (STP)
Descripción	El sistema permitirá mover los nodos del escenario.
Precondición	Haber mostrado el escenario.
Secuencia normal	Pasos / Acción
	1. El usuario (ACT-0001) mostrará el escenario. 2. El usuario moverá los nodos.
Postcondición	Grafo cambiado.
Excepciones	Pasos / Acción
	En caso de que el archivo de configuración no sea válido no se mostrará nada.
Importancia	Importante
Urgencia	Urgente
Estado	Validado
Estabilidad	Alta
Comentarios	-

Tabla 26 - UC-0008

4.3 REQUISITOS NO FUNCIONALES

Los requisitos funcionales son aquellos requisitos que no implican funciones a realizar ni describen información a guardar, sino más bien son de carácter general y que se exigen a cualquier sistema final. Ejemplos de requisitos no funcionales: rendimiento, disponibilidad, estabilidad, etc. Ahora vamos a exponer nuestros requisitos no funcionales.

NFR-0001	Soporte
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar Juan Manuel Vozmediano Torres
Dependencias	Ninguno
Descripción	El sistema debe tener facilidad de instalación, facilidad de mantenimiento, lo que requiere código y diseño documentado y facilidad de actualización hacia versiones más moderna.
Importancia	Importante
Urgencia	Hay presión
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 27 - NFR-0001

NFR-0002	Respuesta
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar Juan Manuel Vozmediano Torres
Dependencias	Ninguno
Descripción	El sistema debe tratar de dar un funcionamiento fluido y con respuestas lo más rápidamente posible.
Importancia	Importante
Urgencia	Hay presión
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 28 - NFR-0002

NFR-0003	Fiabilidad
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar Juan Manuel Vozmediano Torres
Dependencias	Ninguno
Descripción	El sistema debe ser fiable en todo momento de cara a las peticiones realizadas por el usuario.
Importancia	Importante
Urgencia	Hay presión
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 29 - NFR-0003

NFR-0004	Escalabilidad
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar Juan Manuel Vozmediano Torres
Dependencias	Ninguno
Descripción	Nuestro sistema debe ser fácilmente escalable.
Importancia	Importante
Urgencia	Hay presión

Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 30 - NFR-0004

NFR-0005	**Usabilidad**
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar Juan Manuel Vozmediano Torres
Dependencias	Ninguno
Descripción	Nuestro sistema debe ser intuitivo, directo y sencillo de utilizar.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 31 - NFR-0005

NFR-0006	**Portabilidad**
Versión	1.0 (10/11/2014)
Autores	Ana María Escorza Aguilar
Fuentes	Ana María Escorza Aguilar Juan Manuel Vozmediano Torres
Dependencias	Ninguno
Descripción	El sistema podrá usarse en otros sistemas operativos.
Importancia	Importante
Urgencia	Hay presión
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 32 - NFR-0006

4.4 MATRIZ DE RASTREABILIDAD

La matriz de rastreabilidad es muy útil, ya que nos permite con un simple vistazo ver las relaciones entre los casos de uso y los objetivos del sistema.

TRM-0001	OBJ-0001	OBJ-0002	OBJ-0003	OBJ-0004	OBJ-0005	OBJ-0006	OBJ-0007	OBJ-0008
UC-0001	-	↑	↑	-	-	-	↑	↑
UC-0002	-	↑	-	↑	-	↑	↑	↑
UC-0003	-	-	-	-	-	-	↑	↑
UC-0004	↑	-	-	-	-	↑	↑	↑
UC-0005	↑	-	-	-	-	↑	↑	↑
UC-0006	-	-	-	-	-	-	↑	↑
UC-0007	-	-	-	-	↑	↑	-	↑
UC-0008	↑	-	-	-	-	-	↑	↑

Tabla 33 - Matriz de rastreabilidad

Todos los requisitos especificados y los casos de uso se ha confirmado que son válidos y consistentes.

5 ANÁLISIS DEL SISTEMA

En el apartado anterior hemos definidos los requisitos del sistema, los cuales describe la idea del proyecto y sus objetivos. Ahora vamos a describir las tareas que deben llevarse a cabo para examinar los requisitos con el objetivo de delimitarlos y definir exactamente cada uno de ellos.

Los objetivos principales del análisis de requisitos consiste en:[3]

- Detectar y resolver los conflictos entre requisitos.
- Delimitar el software y establecer con qué elementos externos interacciona.
- Elaborar los requisitos del sistema para obtener, a partir de ellos, los requisitos del software a desarrollar.

En dicho análisis se definirán los aspectos más internos del sistema. Para ello recurriremos a una serie de diagramas que se reparten en dos apartados, donde se detallarán los flujos de información que se dan durante la ejecución de la aplicación.

5.1 MODELO ESTÁTICO DEL SISTEMA

Este modelo nos permitirá definir los tipos que necesitaremos durante el desarrollo del sistema y sus interrelaciones con el propósito de obtener una visión global del mismo.

Todo elemento de un modelo estático debe estar trazado hacia aquellos requisitos que lo justifican.

[3] Objetivos principales según la guía SWEBOK

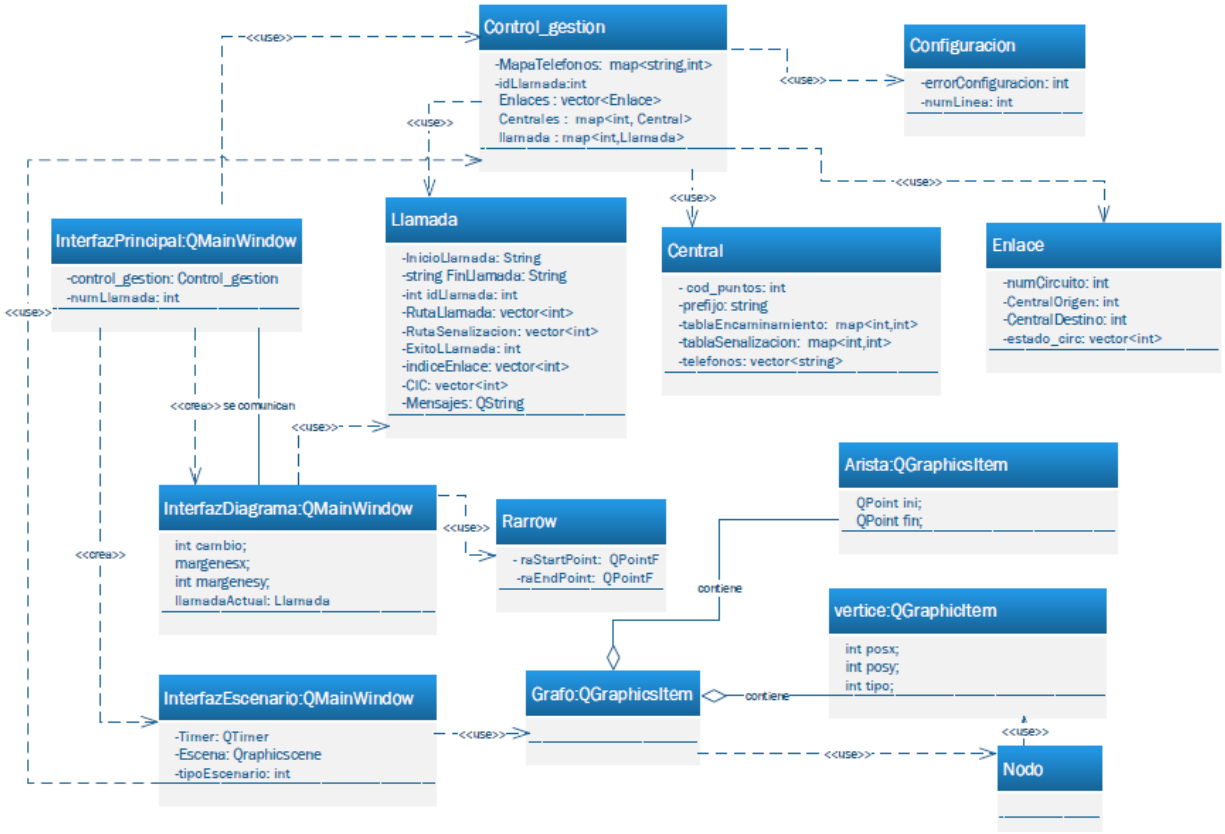


Ilustración 14 - Diagrama de clases

TYP-0001	Control_gestion		
Versión	1.0 (7/4/2013)		
Autores	Ana María Escorza Aguilar		
Dependencias	Ninguno		
Descripción	Objeto principal que se encarga de la gestión de los datos y de la llamada.		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	Ninguno	Tipo	Multiplicidad
	-	-	-
Comentarios	Ninguno		

Ilustración 15 - TYP-0001

TYP-0002	Configuracion		
Versión	1.0 (7/4/2013)		
Autores	Ana María Escorza Aguilar		
Dependencias	Ninguno		
Descripción	Este objeto se encarga de leer etiquetas del archivo de configuración e ir creando las centrales y enlaces definidos.		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	Ninguno	Tipo	Multiplicidad
	-	-	-
Comentarios	Ninguno		

Ilustración 16 - TYP-0002

TYP-0003	Central		
Versión	1.0 (7/4/2013)		
Autores	Ana María Escorza Aguilar		
Dependencias	Ninguno		
Descripción	Este objeto representa una central.		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	Ninguno	Tipo	Multiplicidad
	-	-	-
Comentarios	Ninguno		

Ilustración 17 - TYP-0003

TYP-0004	Enlace		
Versión	1.0 (7/4/2013)		
Autores	Ana María Escorza Aguilar		
Dependencias	Ninguno		
Descripción	Este objeto representa un enlace entre centrales.		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	Ninguno	Tipo	Multiplicidad
	-	-	-
Comentarios	Ninguno		

Ilustración 18 - TYP-0004

TYP-0005	Llamada		
Versión	1.0 (7/4/2013)		
Autores	Ana María Escorza Aguilar		
Dependencias	Ninguno		
Descripción	Este objeto representa la información necesaria de una llamada. Ahí obtenemos los mensajes para el paso de mensajes.		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	Ninguno	Tipo	Multiplicidad
	-	-	-
Comentarios	Ninguno		

Ilustración 19 - TYP-0005

TYP-0006	Grafo		
Versión	1.0 (7/4/2013)		
Autores	Ana María Escorza Aguilar		
Dependencias	Ninguno		
Descripción	Este objeto representa el grafo que forma el escenario.		
Supertipo	QGraphicsItem		
Subtipos	Ninguno		
Componentes	Ninguno	Tipo	Multiplicidad
	-	-	-
Comentarios	Ninguno		

Ilustración 20 - TYP-0006

TYP-0008	Arista		
Versión	1.0 (7/4/2013)		
Autores	Ana María Escorza Aguilar		
Dependencias	Ninguno		
Descripción	Este objeto las uniones entre vértices en el grafo del escenario.		
Supertipo	QGraphicsItem		
Subtipos	Ninguno		
Componentes	Ninguno	Tipo	Multiplicidad
	-	-	-
Comentarios	Ninguno		

Ilustración 21 - TYP-0008

TYP-0009	Vertice		
Versión	1.0 (7/4/2013)		
Autores	Ana María Escorza Aguilar		
Dependencias	Ninguno		
Descripción	Este objeto representa los vértices del grafo		
Supertipo	QGraphicsItem		
Subtipos	Ninguno		
Componentes	Ninguno	Tipo	Multiplicidad
	-	-	-
Comentarios	Ninguno		

Ilustración 22 - TYP-0009

TYP-0010	IntefazEscenario		
Versión	1.0 (7/4/2013)		
Autores	Ana María Escorza Aguilar		
Dependencias	Ninguno		
Descripción	Este objeto representa la ventana que muestra el grafo con el escenario.		
Supertipo	QMainWindow		
Subtipos	Ninguno		
Componentes	Ninguno	Tipo	Multiplicidad
	-	-	-
Comentarios	Ninguno		

Ilustración 23 - TYP-0010

TYP-0011	InterfazDiagrama		
Versión	1.0 (7/4/2013)		
Autores	Ana María Escorza Aguilar		
Dependencias	Ninguno		
Descripción	Este objeto representa la ventana que muestra el paso de mensajes.		
Supertipo	QMainWindow		
Subtipos	Ninguno		
Componentes	Ninguno	Tipo	Multiplicidad
	-	-	-
Comentarios	Ninguno		

Ilustración 24 - TYP-0011

TYP-0012	Rarrow
Versión	1.0 (7/4/2013)
Autores	Ana María Escorza Aguilar
Dependencias	Ninguno
Descripción	Este objeto representa las flechas de los diagramas.
Supertipo	Ninguno

Subtipos	Ninguno		
Componentes	Ninguno	Tipo	Multiplicidad
	-	-	-
Comentarios	Ninguno		

Ilustración 25 - TYP-0012

TYP-0013	**Nodo**		
Versión	1.0 (7/4/2013)		
Autores	Ana María Escorza Aguilar		
Dependencias	Ninguno		
Descripción	Este objeto representa los vértices, para movernos por ellos.		
Supertipo	Ninguno		
Subtipos	Ninguno		
Componentes	Ninguno	Tipo	Multiplicidad
	-	-	-
Comentarios	Ninguno		

Ilustración 26 - TYP-0013

TYP-0014	**InterfazPrincipal**		
Versión	1.0 (7/4/2013)		
Autores	Ana María Escorza Aguilar		
Dependencias	Ninguno		
Descripción	Este objeto representa la interfaz principal del usuario.		
Supertipo	QMainWindow		
Subtipos	Ninguno		
Componentes	Ninguno	Tipo	Multiplicidad
	-	-	-
Comentarios	Ninguno		

Ilustración 27 - TYP-0014

Identificación de Asociaciones y Agregaciones

Control_gestion, clase principal de este proyecto, usa Configuracion para almacenar los datos del archivo de configuración en Centrales y Enlaces. Además, esta clase es la que se encarga de la gestión de las llamadas, almacenando todo lo necesario en la clase Llamada.

La clase InterfazPrincipal utiliza dos clases para crear dos ventanas, una para mostrar el paso de mensajes, InterfazDiagrama, y otra para el escenario, InterfazEscenario. Además, InterfazDiagrama e InterfazPrincipal se comunican entre sí cuando una llamada es liberada.

Por otro lado, InterfazPrincipal usa la clase Rarrow para las flechas del diagrama.

Finalmente, InterfazEscenario, usa Control_gestion para obtener el código de puntos de las Centrales, los números de teléfonos y las relaciones entre éstos. Mediante esta información usa la clase Grafo que contiene Arista y Vertice, y usa Nodo y éste a su vez usa la clase Vertice. Todo esto para obtener el esquema del escenario.

5.2 MODELO DINÁMICO DEL SISTEMA

5.2.1 Operaciones de configuración

El usuario puede cambiar el archivo de configuración tantas veces como desee. Los casos de usos representados en la siguiente figura son:

- [UC-0006] - Recargar archivo de configuración.

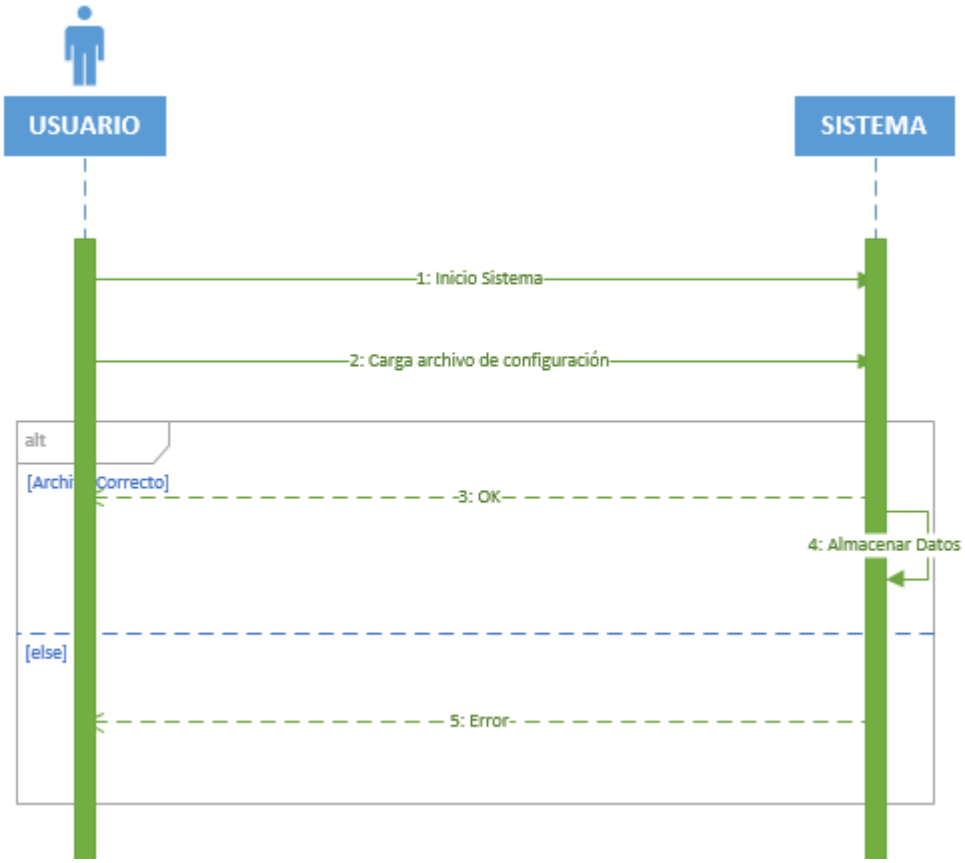


Ilustración 28 - Operaciones Configuración

5.2.2 Operaciones de escenario

El sistema podrá mostrar el escenario que se defina en el archivo de configuración. Este escenario podrá mostrarse desplegado (como un grafo circular) o sin desplegar. Además, los nodos del dibujo representado

Los casos de usos son:

- [UC-0004] - Ver Escenario Desplegado.
- [UC-0005] - Ver Escenario No Desplegado.
- [UC-0008] - Mover Nodos.

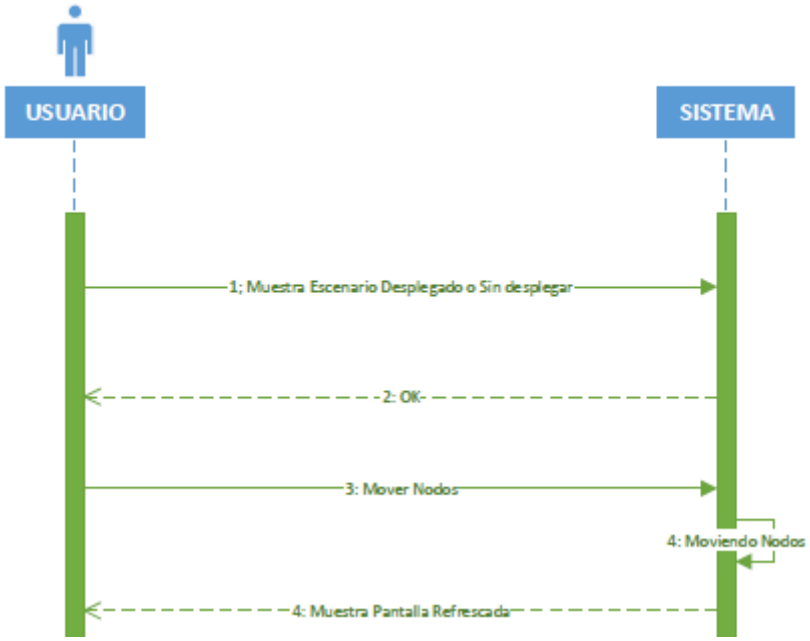


Ilustración 29 - Operaciones Escenario

5.2.3 Operaciones gestión llamadas

Como su propio nombre indica son operaciones relacionadas con el control y gestión de las llamadas, para posteriormente, obtener los datos necesarios, para realizar el paso de mensajes de ésta, y cuándo el usuario decida colgar, igual.

Los casos de usos son los siguientes:

- [UC-0001] - Escribir origen llamada, Escribir destino llamada
- [UC-0002] - Llamar
- [UC-0003] - Listar
- [UC-0007] - Colgar Llamada (Llamante, Llamado)

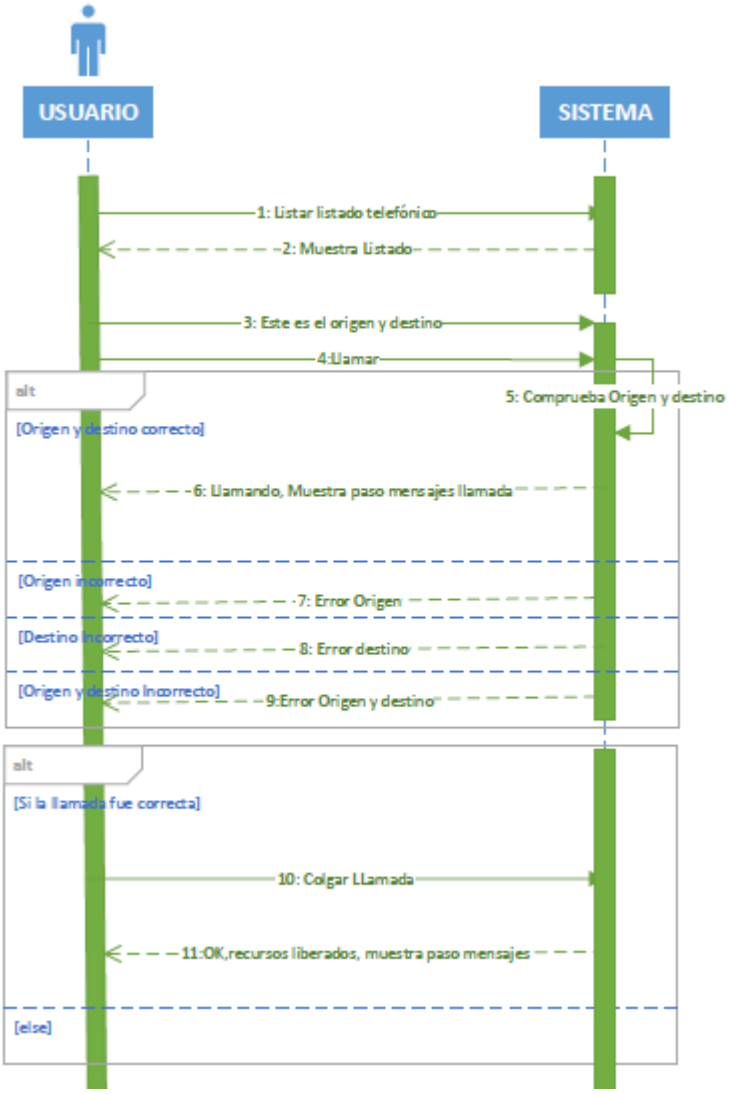


Ilustración 30 - Operaciones Gestión de Llamadas

6 DISEÑO Y ARQUITECTURA DEL SISTEMA

Una vez que hemos acabado con el análisis del sistema continuamos con el diseño del sistema. En esta fase se llevará a cabo dicho diseño que establecerá los pilares de la arquitectura del software a desarrollar así como el diseño de la interfaz de usuario. Está fase es necesario desarrollar un diseño lo más eficiente posible, pues un buen diseño puede marcar la diferencia a la hora de depurar, mantener o ampliar un proyecto software, por lo tanto nos definirá la calidad final del proyecto software.

También se contemplará la parte referente a la interfaz de usuario, ya que es otro de los aspectos más importantes de la aplicación. El diseño de la interfaz de usuario es la encargada de mostrar la información al usuario y que éste interactúe con la aplicación, por lo que es de vital importancia tener una interfaz organizada, sencilla y amigable, con la que el usuario se sienta cómodo trabajando.

6.1 CLASES

6.1.1 Control_Gestion:

Es la clase principal de nuestro programa. Utiliza la clase *Configuracion* para leer del archivo de configuración y almacenar los datos en sus atributos:

- MapaTelefonos: donde almacena los teléfonos de todas las centrales. Es un mapa donde la clave es el teléfono y con ella accedemos al CP de la central donde se encuentra.
- Enlaces: es un vector donde cada elemento es una instancia de la clase *Enlace,* donde se almacenan las características de cada enlace definido (origen, destino, número de circuitos,…)
- Centrales: vector donde cada elemento es una instancia de la clase *Central,* donde almacenamos los atributos de cada una de las centrales leído en el archivo de configuración.

Otra parte de esta clase, es la de gestión de las llamadas, recibe el origen y destino de la llamada, y ella se encarga de gestionarla, almacenando todo lo necesario en el atributo *llamada*. Éste es un mapa donde la clave es un identificador mediante el cual accedemos a la llamada en concreto.

Ilustración 31 - Clase Control_gestion

6.1.2 Configuración:

Se encarga de leer los datos del archivo de configuración. Va creando las centrales y guardándola en el mapa, cada central lee y guarda sus propios datos.

Después lee la parte de Enlaces, guarda la información referente a éste, y almacena cada enlace en un vector.

Ilustración 32 - Clase Configuración

6.1.3 Central y Enlace

Guardamos la información en Central y en Enlace, para utilizarla posteriormente cuando se quiera realizar una llamada y tengamos que gestionarla.

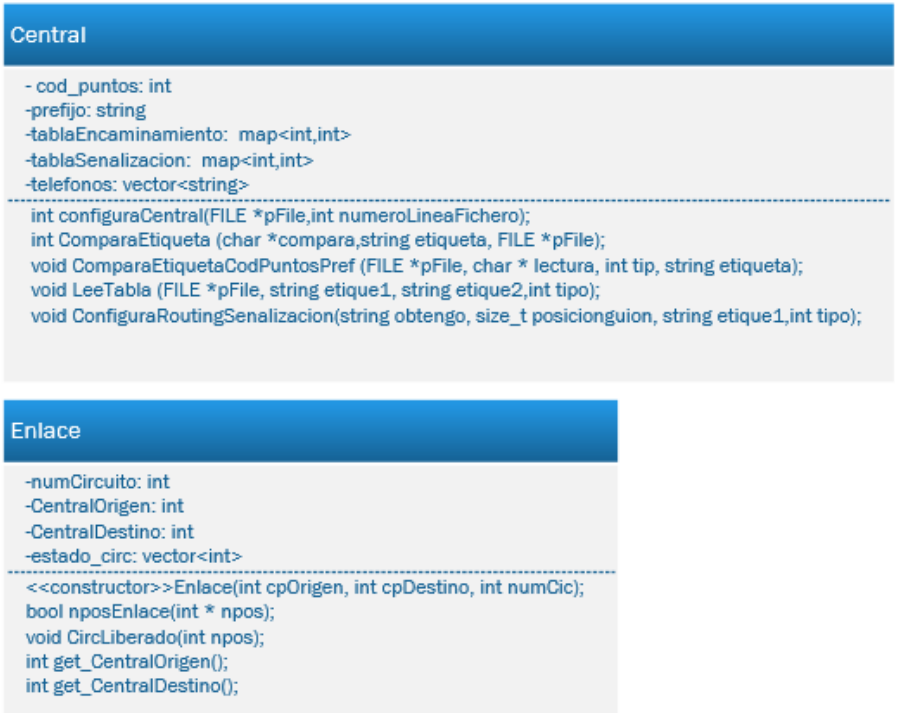


Ilustración 33 - Clase Central y Enlace

Como vemos (Figura 19) para cada central tenemos el código de puntos, un prefijo (identifica la zona de la central), una tabla de encaminamiento y otra para los enlaces de señalización, y por último, los teléfonos pertenecientes a ésta.

Por otro lado, para cada enlace, definimos el número de circuitos en cada enlace, desde que central hasta que central está definido y el estado de cada circuito que posea (ocupado, libre, ocioso). Los métodos utilizados en la clase enlace son para acceder a sus atributos y modificarlos. Por ejemplo, cuando queremos utilizar un circuito, tenemos que buscar cuál está libre, cambiar su estado a ocupado, y restarle uno al número de circuitos. Del mismo modo pero para liberarlo, cambiamos su estado a libre y sumamos uno al número de circuitos.

6.1.4 Llamada

En esta clase se almacena toda la información necesaria para cursar una llamada: origen, destino, identificador, CIC del circuito que ha tomado en cada enlace y el vector de mensajes que le pasamos a la clase InterfazDiagrama para que dibuje el paso de mensajes. Los métodos pertenecientes a esta clase son para rellenar este vector de mensajes.

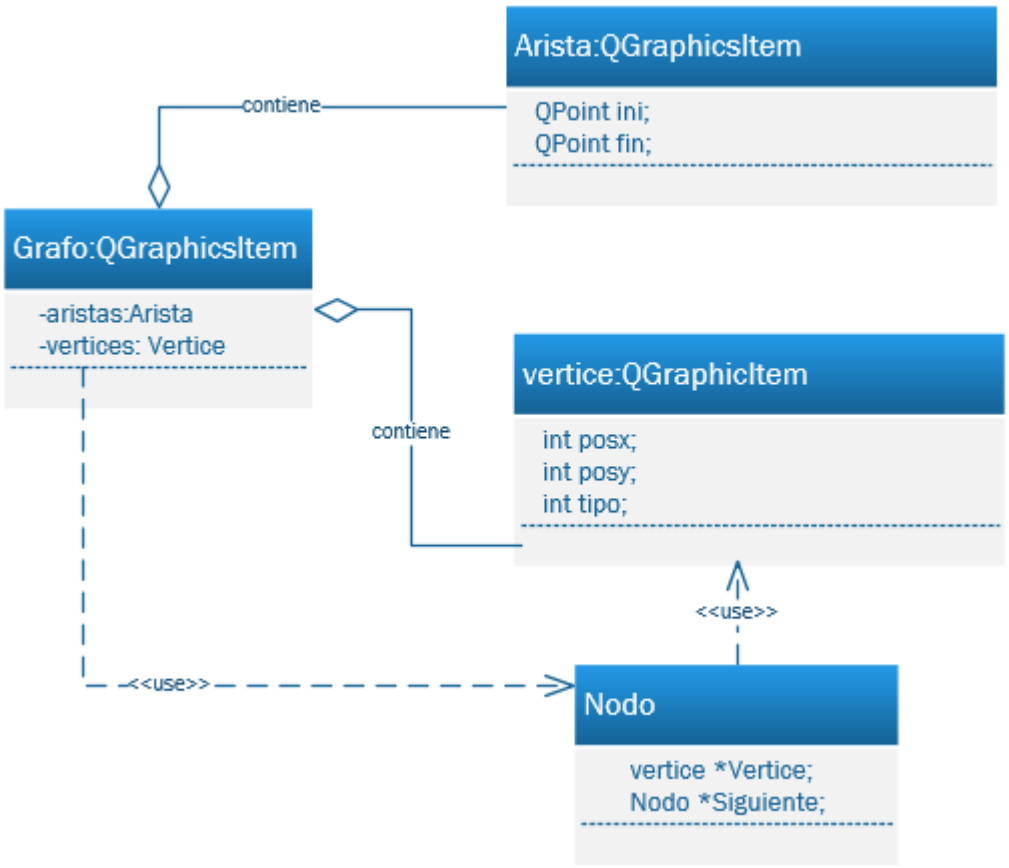
También, posee los atributos RutaLlamada y RutaSenalizacion que hace referencia por las centrales que pasa para llegar al destino por señalización y por voz. Y por último, el atributo, ExitoLlamada, que identifica si la llamada se realizó con éxito, o no. Una llamada puede no realizarse por varios motivos: porque no encuentre el enlace de señalización, porque no encuentre el enlace que le indica la tabla de encaminamiento o porque no existan circuitos libres en un enlace.

Ilustración 34 - Clase Llamada

6.1.5 Grafo, Nodo, Vertice, Arista

Estas clases son para crear el grafo del escenario y representarlo, con las centrales, teléfonos y sus enlaces. Grafo contiene Arista y Vertice. Además, usa la clase Nodo que a su vez ésta usa la clase Vertice, ya que contiene el vértice y el puntero hacia su adyacente.

Las clases Vertice, Grafo y Arista heredan de la clase QGraphicsItem de Qt.



6.1.6 Rarrow

Los mensajes se representan gráficamente mediante flechas. Debido a que Qt no tiene "flechas", pero si otras formas, cómo líneas, polígonos, elipses… He creado una clase que crea gráficamente una flecha mediante cálculos y formas de Qt ya existentes, llamada Rarrow. Podemos cambiar características como grosor línea, grosor de la punta, el ancho y el alto, y el origen y el destino.

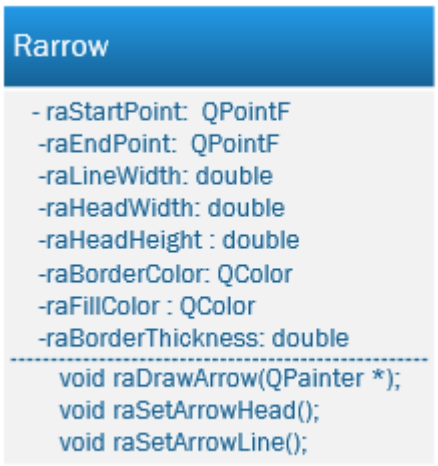


Ilustración 35 - Clase Rarrow

Una vez que le indiquemos el origen y fin de la flecha utilizamos la función "raDrawArrow", que utiliza los otros dos métodos (Figura 21) para dibujarla.

Posteriormente, pasaremos a explicar el diseño de la interfaz gráfica, y las clases utilizadas. Ni que decir tiene que todos los atributos utilizados en todas las clases, son privados y hay que acceder a ellos mediante un setter o un getter.

6.2 DISEÑO DE LA INTERFAZ GRÁFICA

Para el diseño de la interfaz se han desarrollado tres clases. La encargada de la interfaz principal, la que dibuja el paso de mensajes y la que muestra el grafo del escenario.

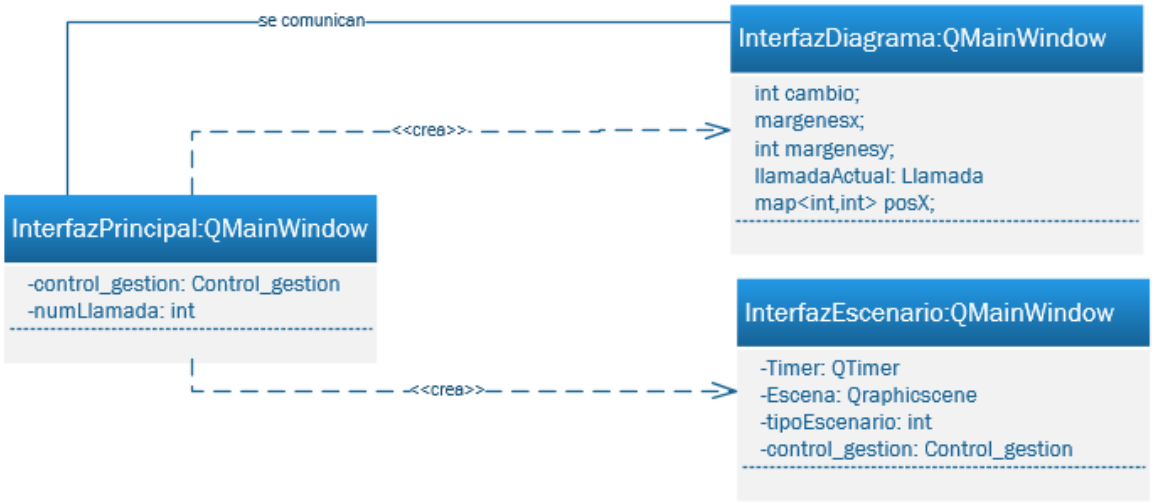


Ilustración 36 - Clases de Interfaz de Usuario

Ambas heredan de la clase QMainWindow de la librería Qt que presenta muchas funcionalidades adicionales.

InterfazPrincipal tiene un atributo perteneciente a la clase Control_gestion que recarga sus datos cada vez que cambiamos el archivo de configuración. Aquí el usuario tiene opción a cambiar este archivo, a ver el listado telefónico, a mostrar el escenario o a realizar una llamada.

Ilustración 37 - Interfaz Principal

Cuando un usuario elige realizar una llamada, InterfazPrincipal utiliza Control_gestion para gestionarla y seguidamente crea una instancia de InterfazDiagrama pasándole en el constructor una instancia de la llamada actual (clase Llamada) que ha tramitado control_gestion. Ahí tenemos almacenado toda la información de la llamada cursada, incluido el vector de mensajes, con todos los mensajes ISUP y Q.931. InterfazDiagrama se encarga de pintar el escenario de la llamada (origen, destino y centrales por las que pasa), y posteriormente recorre el vector dibujando los mensajes.

Además, si la llamada se realizó con éxito tendremos opción a liberarla y con ello los circuitos utilizados. Cuando colgamos la llamada, InterfazDiagrama, envía una señal con el identificador de la llamada a colgar a InterfazPrincipal para que control_gestion (atributo en InterfazPrincipal que se encarga de la gestión de las llamadas) libere los recursos de la llamada que está eligiendo colgar. Si la llamada no se ha realizado con éxito, evidentemente, el usuario no tendrá opción a colgar.

InterfazEscenario, recibe el atributo control_gestion de InterfazPrincipal que contiene las centrales, teléfonos y enlaces de datos y señalización, y se encarga de crear el grafo. Posteriormente mostrarlo al usuario tanto desplegado en forma de grafo circular o plegado, en ambos el usuario puede mover los nodos.

7 IMPLEMENTACIÓN:

Explicaremos brevemente algunas de las soluciones que se han ideado durante la etapa de implementación del proyecto.

7.1 ENCAMINAMIENTO DE LA LLAMADA.

La tabla de encaminamiento de cada central consta de los códigos de puntos de todas las centrales existentes en el escenario y cuál es el siguiente salto. Utilizamos por tanto encaminamiento plano. Esta tabla para acceder mejor al siguiente salto, sin necesidad de recorrerla, utilizamos un mapa en el que la clave es el código de puntos de cada central y con ello accedemos al siguiente salto. Para señalización, lo mismo, pero con los enlaces de señalización.

De modo que para el encaminamiento de la llamada miramos en la central correspondiente su tabla de rutas cuál sería el siguiente salto para llegar a la central en la que se encuentra el destino. Una vez obtenido este código de puntos, lo guardo en un vector de la clase Llamada (RutaLlamada) y compruebo que también puedo acceder a éste por señalización.

Además, en la instancia llamada guardo el identificador del circuito tomado que corresponderá al número de circuitos y el índice del vector de estado. Por otro lado, en el enlace perteneciente al circuito que he tomado (clase Enlace) paso el estado del circuito de libre a ocupado y resto uno al número de circuitos.

Y así hasta que llegue al destino o encuentre un enlace en el que no haya circuitos disponibles. Obteniendo una ruta con todos los códigos de puntos de las centrales por las que pasa el mensaje. También se ha tenido en cuenta que el usuario se equivoque definiendo las tablas de encaminamiento y se produzcan bucles, en ese caso se avisará al usuario.

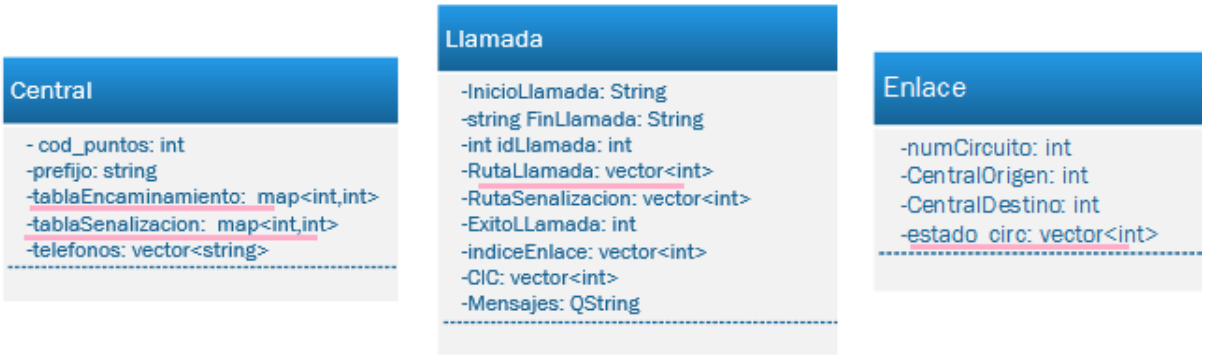


Ilustración 38 - Central, Llamada, Enlace

7.2 MENSAJES (DEFINICIÓN Y REPRESENTACIÓN)

Los que representamos son los mensajes ISUP y Q.931 que vimos en el bloque 2 de este proyecto.

Los mensajes lo definimos mediante una estructura, donde indicamos el tipo mensaje (IAM, ACM, ANM,…) junto con el identificador de circuito (CIC) cuándo procede, el código de punto de la central origen y el código de puntos de la central destino.

```
struct Mensaje{
    QString msg;
    int inicio;
    int fin;
};
```

Ilustración 39 - Estructura del mensaje

MANEJAR LOS MENSAJES:

La clase *Llamada*, además de almacenar los datos necesarios para cursarla, es la que se encarga mediante sus métodos de crear una tabla de estructuras con todos los mensajes del establecimiento de la llamada (mensajes ISUP y Q.931), o de liberación.

REPRESENTAR LOS MENSAJES:

InterfazDiagrama es la que se encarga de la representación gráfica. Recibe una instancia de la clase Llamada con todos los datos de la llamada, y con los métodos para crear la tabla de mensajes. Esta tabla tiene el tamaño del número de mensajes a guardar en ella, que se calcula a priori:

(NºMensaje Q.931)*2+(NºMensajesISUP)*(NºCentrales que pasa -1)

Lo primero que hace InterfazDiagrama es asignarle a cada central y a cada teléfono una posición X en la ventana, dividiendo la ventana en tantas partes como nodos. En la siguiente Figura 25, podemos ver un ejemplo y como calculamos el ancho y largo de la pantalla mediante funciones de QMainWindow, de donde hereda InterfazDiagrama. En la Figura 26, podemos ver como lo dibuja equispaciadamente.

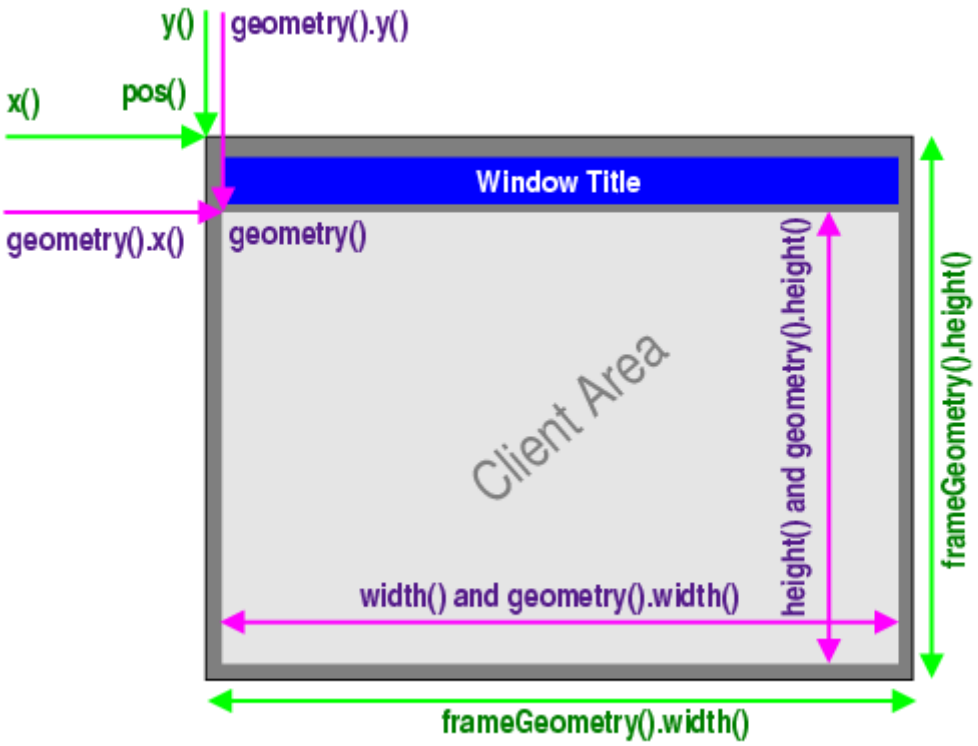


Ilustración 40 - Obtener anchura y altura de ventana

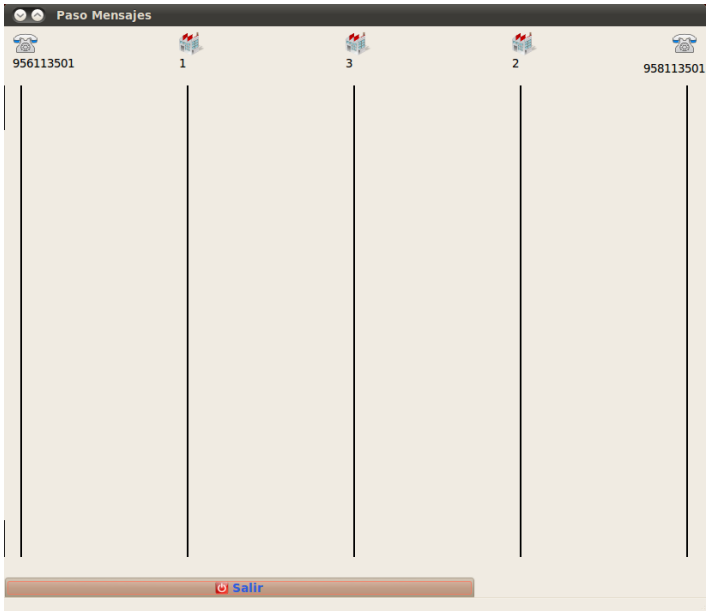


Ilustración 41 - Sin Mensajes

La posición X de dónde está colocado cada nodo lo guardamos en un mapa, cuya clave es el propio nodo.

Una vez actualizado este mapa, recorrerá el vector de mensajes y traducirá del mapa, el origen y destino del mensaje por posiciones en la ventana gráfica de donde se encuentra cada nodo, creando una flecha con esas dos posiciones X de principio y fin.

Ilustración 42 - Función recorre tabla de estructura de mensajes

Y para que se pinten seguidamente y no se solapen vamos aumentando la posición Y. Lo que le sumamos a Y lo calculamos de la siguiente forma, donde height() es un método que calcula la altura de la ventana.

```
distanciaMensajes = height()/numMensajes;
```

Ilustración 43 - Distancia entre mensajes

Obteniéndo un diagrama como éste (Figura 29):

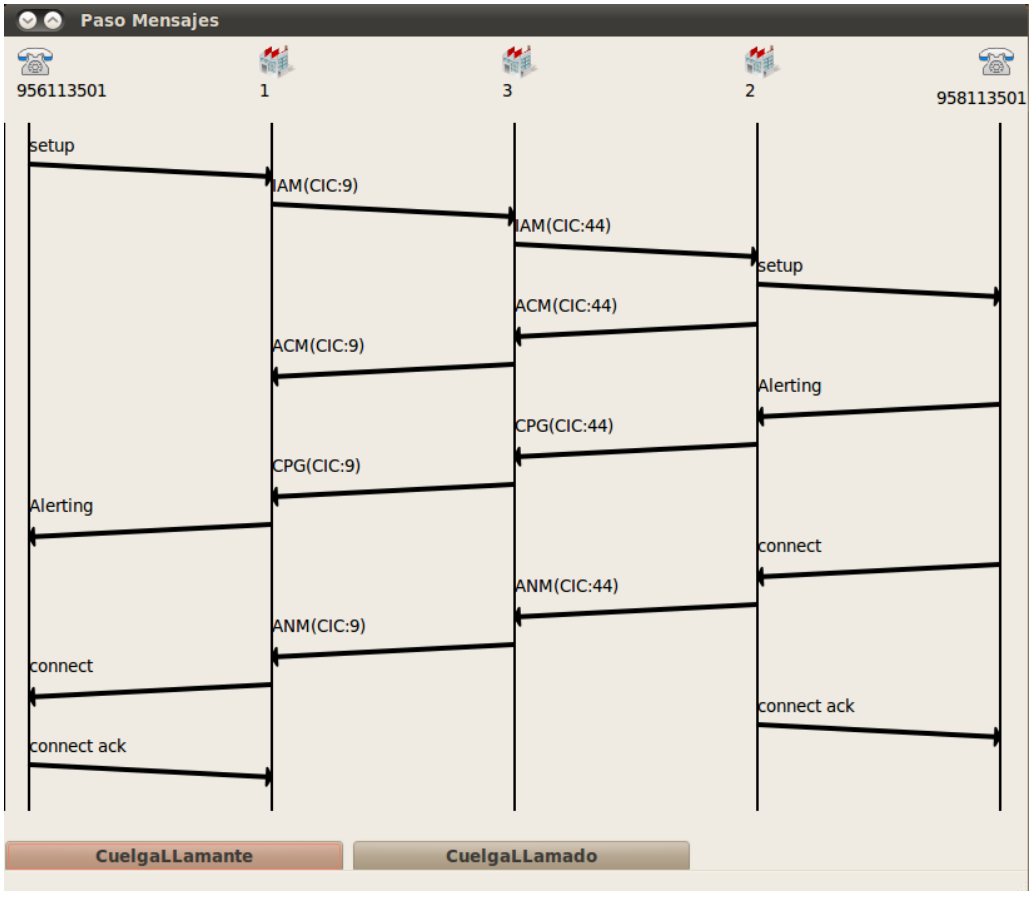


Ilustración 44 - Diagrama Ejemplo, establecimiento llamada.

Cuando se cuelga una llamada, lo que se hace es invocar a un método de la clase Llamada que cambia esta tabla de mensajes de establecimiento de llamadas, por otra con los mensajes de liberación. Y sólo tenemos que llamar al método que recorre los mensajes de InterfazDiagrama para volverlos a representar.

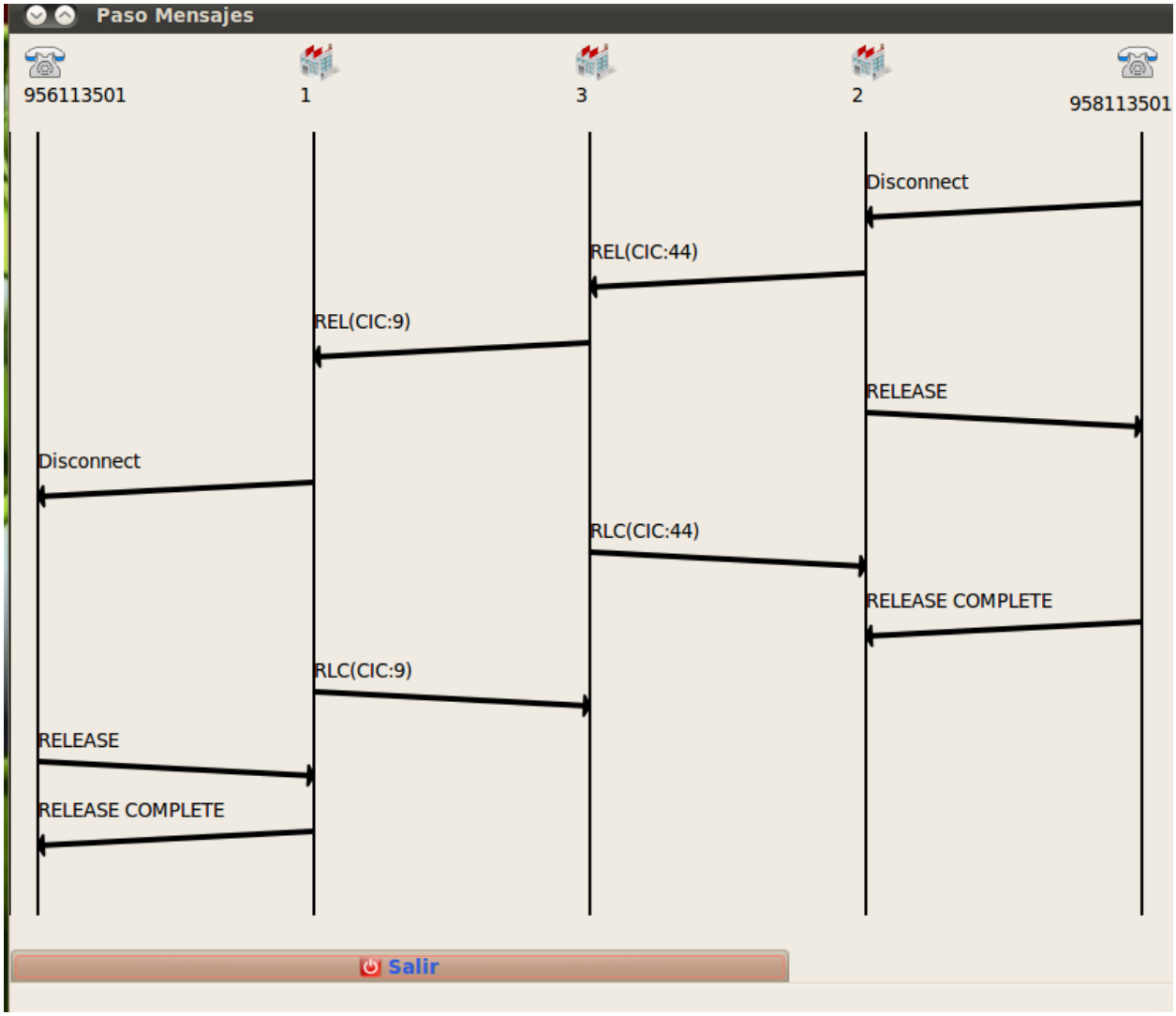


Ilustración 45 - Diagrama ejemplo, cuelga llamante

Por último añadir, que cuando se redimensione la pantalla, cambie su tamaño, hacemos saltar un evento que detecta este cambio y vuelve a hacer este mismo proceso de dibujo, actualizando antes que nada el mapa de posiciones. Así logramos que se redimensione la representación de paso de mensajes cada vez que cambia el tamaño de la pantalla.

```
void InterfazDiagrama::paintEvent(QPaintEvent *event)
```

Ilustración 46 - Evento detecta cambio tamaño pantalla

7.3 COMUNICACIÓN

Mediante los conectores Signal y Slot podemos establecer comunicaciones entre objetos. Es la manera con la que podemos notificar a otro Widget (ventana, en nuestro caso) cuando este ha sido modificado.

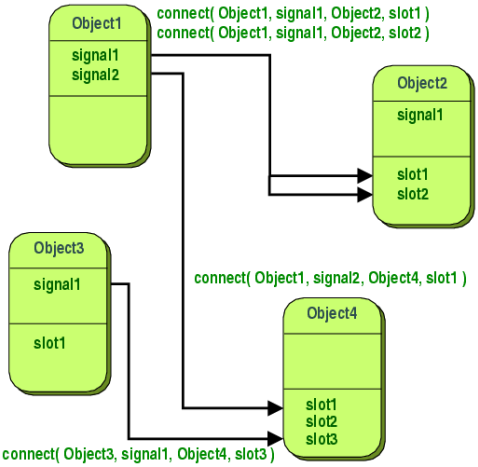


Ilustración 47 - Ejemplo de conectores Qt

InterfazDiagrama tiene que comunicarle a InterfazPrincipal cuando el usuario decide colgar una llamada para que libere los recursos (circuitos) y borre la llamada. Pues si recordamos InterfazDiagrama es el que contiene el atributo control_gestion perteneciente a la clase Control_gestion encargada de la gestión de las llamadas. Para saber la llamada que estamos tratando, la clase Llamada tiene un atributo para identificar cada llamada.

Por lo que mediante los métodos de signals y slots de Qt, cuando el usuario pulsa el botón de colgar InterfazDiagrama envía una señal a InterfazPrincipal y con ella el identificador de la llamada. Una vez recibido la señal, liberamos los circuitos, de ocupado a libre, y sumamos uno al número de circuitos total en el enlace al que pertenece.

Es decir, en InterfazDiagrama emitimos esta señal, cuando el usuario elija colgar la llamada:

```
signals:
    void emitesenal(int id); //emite la señal con el id de la llamada que se cuelga
```

Ilustración 48 - Señal

Y en InterfaPrincipal, definimos este slot, que capta la señal emitida:

```
void InterfazPrincipal::recibesenal(int id)
{
    control_gestion->libera(id);
    control_gestion->descuelgaLlamada(id);
}
```

Ilustración 49 - Slot

Y mediante esta sentencia conectamos la señal con el slot:

```
connect(diagram, SIGNAL(emitesenal(int)), this, SLOT(recibesenal(int)));
```

Ilustración 50 - Conectar señal y slot

7.4 CLASES QT USADAS

A continuación expliclaremos las principales clases de Qt utilizadas y su uso.

QMainWindow:

Esta clase de la que hereda nuestras clases de interfaz, provee el marco de trabajo para construir una aplicación con interfaz de usuario. QMainWindow tiene su propio contenedor de Widget en el que se podrán añadir barra de herramientas, dock widgets, barras de menú y barras de estado. El contenedor dispone de un área central que puede ser ocupado por cualquier tipo de Widget.

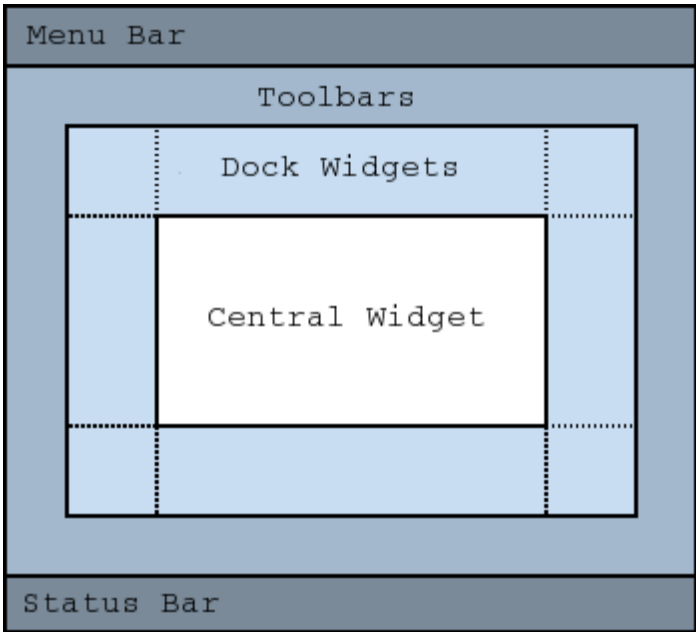


Ilustración 51 - Estructura de las interfaces.

Pushbuttom:

Usando la clase QPushButton hemos podido fabricar botones de comando. Cada botón está asociado a un slot que llama cuando se pulsa y se realiza la acción correspondiente.

Ilustración 52 - Botón listar y llamar

Line Edit:

Con el uso la clase QLineEdit hemos podido fabricar líneas de texto, para introducir el origen y el destino de la llamada.

Ilustración 53 - QLIneEdit

Label:

Nos ha permitido crear etiquetas para insertar tanto texto como imágenes en la interfaz principal.

QListWidget:

Sirve como contenedor para los elementos de tipo QListWidgetItem. La hemos utilizado para listar el listado telefónico.

Ilustración 54 - QListWidget

QMenu:

Mediante la clase QMenu hemos podido incorporar un control de menú para usar en la barras de menús e incluso en los menús contextuales. Los menús desplegables se mostrarán al usuario cuando este haga clic sobre él.

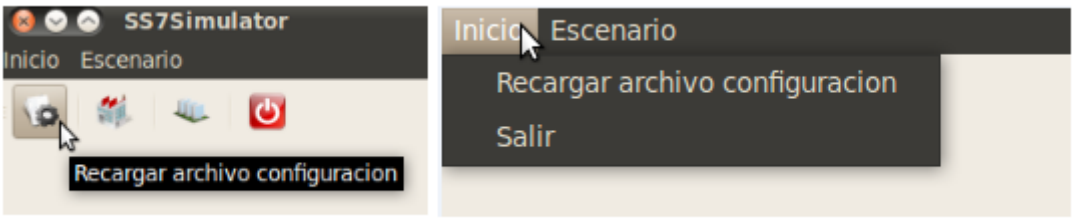


Ilustración 55 - QMenu, QMenuBar

QMenuBar:

Para crear la barra de menús que se muestra en SS7Simulator hemos tenido que hacer uso de la clase QMenuBar la cual implementa la funcionalidad de las barras de herramientas horizontales.

QToolBar:

Creamos la barra de herramientas que se muestra en la Figura 55. Si dejamos el ratón encima, nos sale una descripción de cada icono.

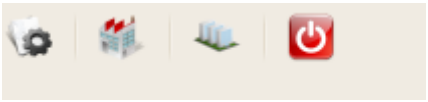


Ilustración 56 - QtoolBar

QGraphicsItem y QGraphicsScene:

Clases para el diseño de gráficos. La clase QGraphicsItem es la clase base para todos los elementos gráficos en un QGraphicsScene.

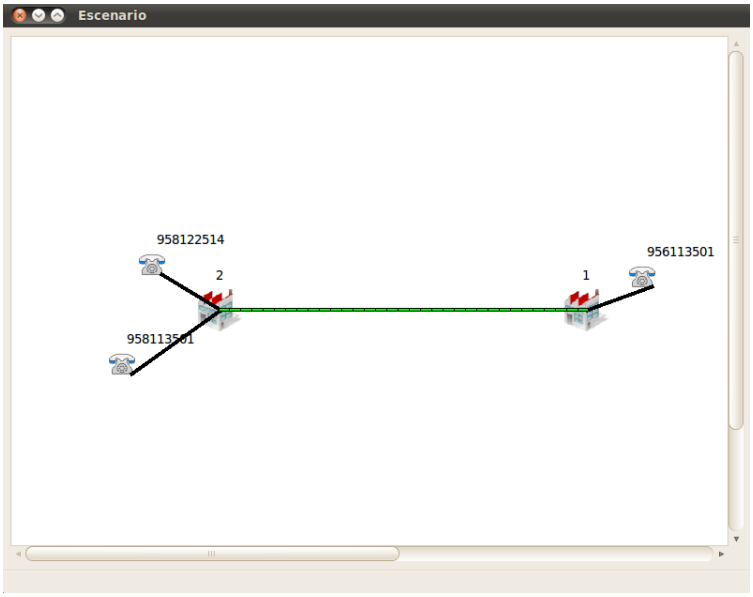


Ilustración 57 - Uso QGraphicsScene, QGraphicsItem

QPainter:

Se usa para llamar a las primitivas de dibujo: puntos, líneas, rectángulos, elipses, arcos, polígonos, pixmaps, imágenes, texto, etc.

QPixmap:

Está diseñada y optimizada para mostrar imágenes en pantalla.

QMessageBox:

Un cuadro de mensaje muestra un texto inicial para alertar al usuario sobre una situación, o un texto informativo para explicar con más detalle la descripción del problema o para mostrar una nueva situación. También suelen incluir iconos para notificar la gravedad del mensaje.

Lo usamos para alertar de los errores del archivo de configuración e introducción de origen y fin de la llamada. En la Figura 57 mostramos algunos ejemplos.

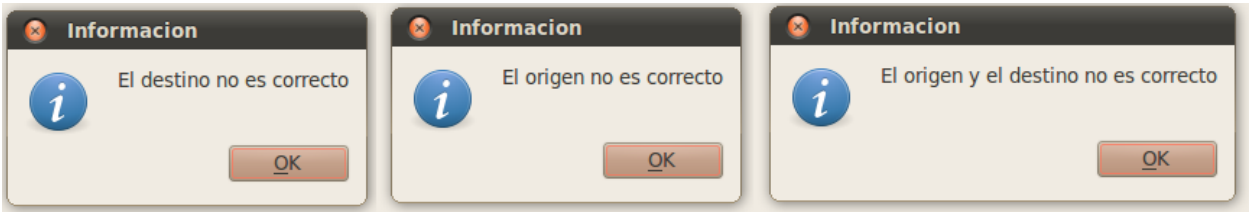


Ilustración 58 - QMessageBox

SplashScreen:

Durante el inicio de la aplicación se le ofrece al usuario una pantalla de presentación.

TFG: Ana María Escorza Aguilar

Ilustración 59 - SplashScreen

8 PRUEBAS

El apartado de pruebas es vital en un proyecto, puesto que es la última oportunidad de comprobar que se cumplen todos los requisitos y exigencias antes de dar el proyecto por finalizado. A continuación, se describirán los procedimientos que se han empleado para probar el sistema y garantizar así la calidad del producto final.

8.1 CONSIDERACIONES DEL PLAN DE PRUEBA:

La prioridad en el plan de prueba es que se compruebe la mayor cantidad de código posible, para evitar de esta forma que se escape cualquier error de programación y ahorrarnos situaciones no esperadas. Para ello intentaremos que sea comprobada la mayor casuística posible haciendo uso de las pruebas de cajas blancas u otras pruebas que se consideren oportunas. Entre dichas pruebas se comprobará tanto los valores límite en las condiciones como los valores exageradamente altos en los tipos numéricos.

8.2 CONDICIONES GENERALES:

Se debe comprobar que la funcionalidad del sistema es completa y exacta para llevar a cabo la validación. Por lo tanto se realizarán pruebas unitarias a cada uno de los módulos, y una vez acabadas se analizará el funcionamiento del conjunto del programa.

Se comprobará que todas las excepciones son manejada de forma correcta según el caso, de esta forma se evitará rupturas de ejecución y otras situaciones no deseadas. Evaluaremos los tiempos de respuesta del sistema a las distintas peticiones del usuario. También se validará que los datos almacenados sean correctos, y que se resuelva en un tiempo razonable aquellas operaciones que bloqueen el sistema o en caso contrario, proporcionar una alternativa al usuario.

8.3 DEFINICIÓN DEL ALCANCE DE LAS PRUEBAS:

Los distintos niveles de pruebas que se van a realizar al sistema durante su desarrollo son los que se citan a continuación:

- Pruebas unitarias, de cada una de las funciones de las clases del sistema. Se realizarán durante las primeras fases de programación del sistema. Deben realizar correctamente su funcionalidad y devolver los valores esperados para ser validadas.
- Pruebas de integración, entre las distintas funciones. Se realizarán junto con las pruebas unitarias. Para ser validadas las funciones de cada una de las clases deben interoperar entre sí de manera satisfactoria.
- Pruebas del sistema, una vez desarrolladas todas las clases del sistema se debe probar en un entorno local el correcto funcionamiento total del sistema con datos reales.

- Pruebas de implantación, una vez superadas las pruebas del sistema de manera local debe probarse en la localización real. Para ser verificadas el sistema debe presentar el mismo resultado que en el entorno local.
- Pruebas de aceptación, el tutor deberá validar el sistema.

8.3.1 Pruebas unitarias:

Estas pruebas consisten en probar los módulos que componen el sistema de forma individual, para una vez garantizado su correcto funcionamiento de forma independiente, realizar las pruebas de integración.

Prueba	**PU-01**
Objetivo	Comprobar el correcto funcionamiento de los métodos de cada una de las clases.
Descripción	Cada uno de los métodos deberá realizar la funcionalidad descrita correctamente. Devolviendo los resultados y tipos esperados.
Resultado	Superada

Tabla 34 - PU-01

Prueba	**PU-02**
Objetivo	Comprobar el correcto almacenamiento de los datos del archivo de configuración en el sistema.
Descripción	La lectura del archivo de configuración tiene que ser correcta y los datos obtenidos almacenado en su lugar.
Resultado	Superada

Tabla 35 - PU-02

Prueba	**PU-03**
Objetivo	Manejar los errores del archivo de configuración.
Descripción	Si algo no está bien definido, o alguna etiqueta es errónea, se debe avisar al usuario.
Resultado	Superada

Tabla 36 - PU-03

Prueba	**PU-04**
Objetivo	Gestión de llamada
Descripción	Comprobar que las llamadas se encaminan correctamente.
Resultado	Superada

Tabla 37 - PU-04

Prueba	PU-05
Objetivo	Mensajes
Descripción	Comprobaremos que tanto los mensajes de establecimiento de llamada, como de liberación, son los correctos y se almacenan correctamente.
Resultado	Superada

Tabla 38 - PU-05

Prueba	PU-06
Objetivo	Gestión de enlaces.
Descripción	Los enlaces se sumarán o restarán según se utilicen o liberen. También cambiará su estado.
Resultado	Superada

Tabla 39 - PU-06

Prueba	PU-07
Objetivo	Interfaz Gráfica
Descripción	Se mostrará correctamente la interfaz gráfica con las opciones correspondientes.
Resultado	Superada

Tabla 40 - PU-09

8.3.2 Pruebas de integración:

Las pruebas de integración se encargan de garantizar que los módulos interaccionen entre sí de forma correcta tal como se espera.

Prueba	PI-01
Objetivo	Recargar archivo de configuración
Descripción	Si el archivo de configuración es correcto, todos los datos se almacenarán en el sistema. En otro caso, avisará del error, en el número de línea, y no almacenará ningún valor.
Resultado	Superada

Tabla 41 - PI-01

Prueba	PI-02
Objetivo	Listar
Descripción	Se mostrará el listado telefónico cargado del archivo de configuración.
Resultado	Superada

Tabla 42 - PI-02

Prueba	PI-03
Objetivo	Detectar origen o destino incorrecto.
Descripción	Cuando el usuario elija el origen y destino de la llamada, el sistema comprobará si es correcto.
Resultado	Superada

Tabla 43 - PI-03

Prueba	PI-04
Objetivo	Ejecución satisfactoria al realizar una llamada.
Descripción	Se mostrará el paso de mensajes correcto, tanto al realizarla como al liberarla.
Resultado	Superada

Tabla 44 - PI-04

Prueba	PU-05
Objetivo	Grafo.
Descripción	La estructura del grafo y las posiciones deben de ser las correctas.
Resultado	Superada

Tabla 45 - PI-05

Prueba	PU-06
Objetivo	Comunicación entre ventanas.
Descripción	Cuando se realiza una acción una ventana avisa a su ventana padre. En nuestro caso, cuando se pulsa colgar llamada.
Resultado	Superada

Tabla 46 - PI-06

8.3.3 Pruebas del sistema:

Prueba	PS-01
Objetivo	Concurrencia.
Descripción	Se podrá realizar más de una llamada y el sistema funcionará correctamente.
Resultado	Superada

Tabla 47 - PS-01

Prueba	PS-02
Objetivo	Cambio.
Descripción	Si se cambia el archivo de configuración, el sistema actualiza los datos y seguirá funcionando correctamente.
Resultado	Superada

Tabla 48 - PS-02

Prueba	PS-03
Objetivo	Ejecución satisfactoria cuando se muestre el escenario.
Descripción	Se mostrará el escenario que hemos guardado en el grafo. Los nodos se pueden mover.
Resultado	Superada

Tabla 49 - PS-03

8.3.4 Pruebas de implantación:

Prueba	PImp-01
Objetivo	Funcionamiento en entorno real
Descripción	El sistema funcionará correctamente en cualquier equipo, sin el entorno utilizado.
Resultado	Superada

Tabla 50 - PImp-01

8.3.5 Pruebas de aceptación:

Prueba	PA-01
Objetivo	Aceptación del tutor.
Descripción	El tutor del proyecto debe validar el sistema.
Resultado	Superada

Tabla 51 - PA-01

9 PLANIFICACIÓN

El proyecto ha sido desarrollado durante un periodo de tiempo, en el cual hemos tenido que organizar las tareas y el tiempo que vamos a dedicarles a ellas, primero hicimos una estimación temporal a priori, y al finalizarlo contrastaremos la estimación temporal calculada a priori con la que obtendremos realmente a posteriori. Además, también mostraremos sus respectivos diagramas de Gantt.

9.1 ESTIMACIÓN TEMPORAL A PRIORI

Documentación	125 horas
Documentación SS7	15 horas
Documentación C++	48 horas
Documentación Qt	62 horas

Especificación y análisis de requisitos	26 horas
Definición de objetivos	4 horas
Establecimiento de los requisitos	10 horas
Descripción de los requisitos funcionales	4 horas
Detallar casos de uso	4 horas
Determinación de los requisitos no funcionales	4 horas

Análisis	9 horas
Identificación de los subsistemas	2 horas
Desarrollo de las operaciones del sistema	7 horas

Diseño	12 horas
Diseño de clases	7 horas
Diseño de interfaz de usuario	5 horas

Implementación	60 horas
Codificación de clases	40 horas
Integración del sistema	20 horas

Post – Implementación	17 horas
Testeo y pruebas	5 horas
Corrección de errores	12 horas

Total horas estimadas	249 horas

Ilustración 60 - Diagrama de Gantt estimado a priori

9.2 ESTIMACIÓN TEMPORAL A POSTERIORI

Documentación	135 horas
Documentación SS7	15 horas
Documentación C++	48 horas
Documentación Qt	72 horas

Especificación y análisis de requisitos	28 horas
Definición de objetivos	4 horas
Establecimiento de los requisitos	10 horas
Descripción de los requisitos funcionales	6 horas
Detallar casos de uso	4 horas
Determinación de los requisitos no funcionales	4 horas

Análisis	13 horas
Identificación de los subsistemas	4 horas
Desarrollo de las operaciones del sistema	7 horas

Diseño	22 horas
Diseño de clases	7 horas
Diseño de interfaz de usuario	11 horas

Implementación	80 horas
Codificación de clases	60 horas
Integración del sistema	20 horas

Post – Implementación	18 horas
Testeo y pruebas	7 horas
Corrección de errores	12 horas

Total horas estimadas	297 horas

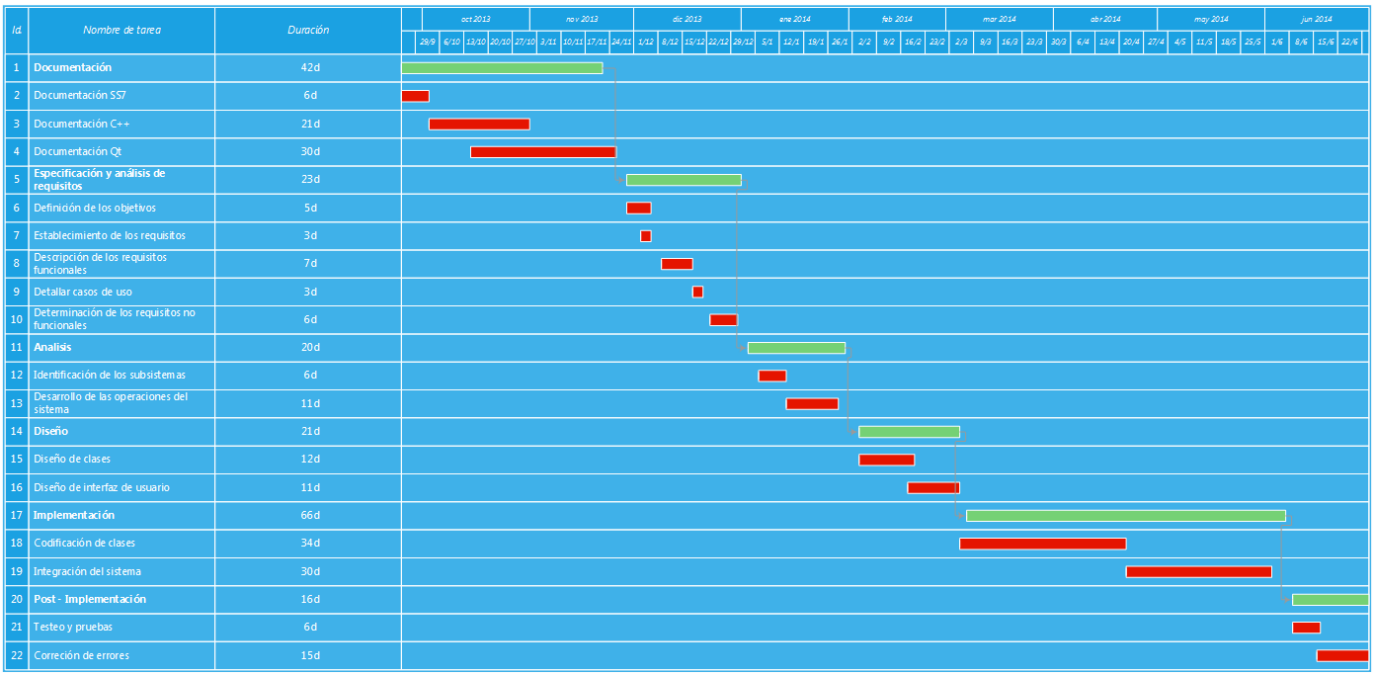


Ilustración 61 - Diagrama de Gantt estimado aposteriori

10 CONCLUSIONES

Una vez acabado el proyecto, llegamos a la hora de sacar las conclusiones y evaluar la experiencia que supone llevar un proyecto de estas características.

Llevar a cabo un proyecto real de esta envergadura es una gran experiencia que expone diversas situaciones que el alumno puede encontrarse en cualquier proyecto software, por lo que aporta experiencia y habilidad para dichos problemas. La realización del proyecto ha ayudado al alumno además de profundizar en los conocimientos en programación, a obtener mayor experiencia en la ingeniería del software y las tareas que acarrea tales como analizar, modelar y desarrollar un proyecto dado los requisitos obtenidos durante la fase de obtención de requisitos.

Cabe destacar que al ser un único alumno, el que desarrolla el proyecto, no ha podido obtener más experiencia a la hora de trabajar en equipo que la que ya poseía previamente, habilidad muy necesaria a la hora de realizar grandes proyectos en equipo.

Este proyecto ha permitido al alumno adentrarse en el estándar de señalización número 7 y en el mundo de la programación C++ junto con la biblioteca multiplataforma Qt para interfaces gráficas.

Ha sido todo un reto y motivación el transcurso del proyecto.

10.1 MEJORAS

Aunque la aplicación cumple los requisitos exigidos, hay muchos aspectos que pueden ser mejorados. No obstante, se ha intentado modelar el proyecto con vistas a que la extensión de funcionalidades sea más sencilla.

Como mejoras se podría añadir más tipos de nodos, como base de datos, más tipos de mensajes y portabilidad.

11 BIBLIOGRAFÍA

1. ***http://web.archive.org/web/20090226053235/http://qtsoftware.com/qt-in-use/story/app/skype.***

2. http://web.archive.org/web/20110723145854/qt.nokia.com/qt-in-use/qt-in-home-media. [En línea]

3. *http://web.archive.org/web/20110714191457/qt.nokia.com/qt-in-use/story/customer/volvo-mobility-systems.*

4. http://qt-project.org/doc/qt-5/mobiledevelopment.html. [En línea]

5. *http://www.itu.int/rec/T-REC-Q.700-199303-I/e.*

6. *http://docente.ucol.mx/al980347/public_html/capas.htm.*

7. http://qt-project.org/. [En línea]

8. http://www.itu.int/rec/T-REC-Q.700-199303-I/e

***9.* http://grokbase.com/t/python/python-es/053p679fsy/pyqt-como-maximizar-un-qwidget**

10. http://stackoverflow.com/

11. http://www.lawebdelprogramador.com/foros/QT/index1.html

12. http://www.cplusplus.com/reference/

13. https://qt-project.org/forums

14. https://www.iconfinder.com/

15. http://www.francescmm.com/category/extras/

16. http://www.codigoqt.com/index.php?topic=70.0

17. www.dependencywalker.com

18. http://programmingexamples.wikidot.com/qt-qpainter-example

19. https://www.cs.princeton.edu/~rs/Algs3.cxx5/code.txt

20. http://code.google.com/p/qgv/

21. http://www.bogotobogo.com/cplusplus/boost.php

22. http://www.richelbilderbeek.nl/CppBoostGraphExample4.htm

23. http://www.algoritmia.net/articles.php?id=18

24. http://blog.ivank.net/force-based-graph-drawing-in-as3.html

25. http://processingjs.nihongoresources.com/graphs/

26. Lee Dryburgh, Jeff Hewett, Signaling System No. 7 (SS7/C7) : protocol, architecture, and services. Indianapolis, IN : Cisco, 2005

27. Jasmin Blanchette, Mark Summerfield . C++ GUI programming with Qt 4. Prentice-Hall, 2006

12 ANEXOS

Finalmente en los anexos, definiré como obtener un ejecutable para los distintos sistemas operativos y, además, un manual de usuario, explicando el uso del programa y la definición del archivo de configuración. Asimismo, se expondrán dos ejemplos prácticos.

12.1 OBTENCIÓN DEL EJECUTABLE

La obtención del ejecutable para los distintos sistemas operativos es muy fácil, ya que Qt es una biblioteca multiplataforma. Sólo tendríamos que instalar el entorno de desarrollo de Qt en el sistema operativo que queramos tener nuestro programa y compilarlo. Para incluir un proyecto solo tenemos que abrir el archivo SS7Simulator.pro y automáticamente nos aparecerán los demás archivos. En este fichero lo que hacemos es definir todos los ficheros que componen el programa inclusive la carpeta de recursos con las imágenes de la aplicación, como vemos en la Figura 61.

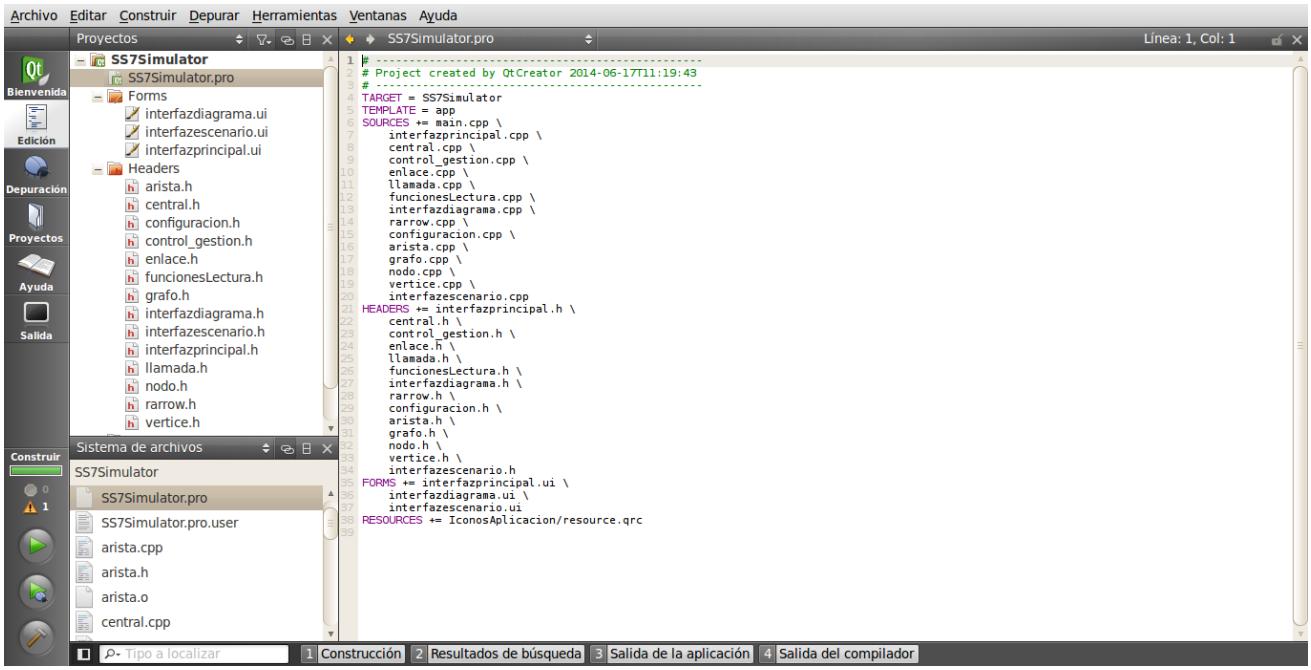


Ilustración 62 - SS7Simulator.pro

Una vez hecho esto, pulsamos el botón verde de la izquierda que compila el código, crea y ejecuta el ejecutable.

12.2 MANUAL DE USUARIO

12.2.1 Introducción

SS7 Simulator es un programa didáctico con el que aprenderás señalización en una red de conmutación de circuitos. Para su uso, primero tienes que definir un archivo de configuración, con el que especificamos el escenario.

Seguidamente, explicaremos la estructura del archivo de configuración, y, tras esto, cómo usamos el programa. Por último, se expondrán dos ejemplos.

12.2.2 Definición del archivo de configuración

El fichero de configuración de SS7Simulator es un fichero de texto que contiene un conjunto de parámetros o directivas que determinan el funcionamiento del programa. Su nombre tiene que ser "conf".

En primer lugar, lo primero que definimos son las centrales con sus teléfonos, y, tras esto, definimos los enlaces entre éstas.

Para definir una central, nos fijaremos en el siguiente ejemplo:

```
<CENTRAL SP>
CODE_POINT 1
PREFIX 956
</PHONE
113501;
>

</ROUTING TABLE
DEST: 2 - NEXTHOP:2;
>
<SIGNALING TABLE
DEST:2 - NEXTHOP: 2;
>
</CENTRAL>
```

Ilustración 63 - Definición de una central

Entre estas dos etiquetas queda definida una central: <CENTRAL SP> y <\CENTRAL>

Los parámetros son:

- Código de puntos: cada nodo está identificado un valor. Se indica mediante la directiva CP_POINT, seguida del código de puntos que queramos darle a esa central. Tiene que ser un número.
- Prefijo: hace referencia a la ubicación. Un prefijo telefónico es una sucesión numérica que se marca delante del número de usuario al realizar una llamada telefónica, con el propósito de seleccionar la demarcación territorial lógica a la que pertenece dicho usuario. Se indica con la etiqueta PREFIX, seguida del prefijo. Tiene que ser un número y tener 3 caracteres.
- Listado de teléfonos: dentro de estas etiquetas "</PHONE y >" definimos los teléfonos pertenecientes a la central. Tienen 6 dígitos, son letras y terminan en ; .
- Tabla de encaminamiento, enlace de datos: la definimos entre las etiquetas <ROUTING TABLE y >. Dentro especificamos el código de puntos de todas las centrales existentes, excepto ella misma, seguida de cuál sería el siguiente salto a ella. Tal como viene en la Figura *.
- Tabla de señalización: igual que la anterior, pero haciendo referencia a los enlaces de señalización.

Y así con todas las centrales que queramos definir en nuestro escenario. Tras hacer esto, definimos los enlaces de datos entre ellas.

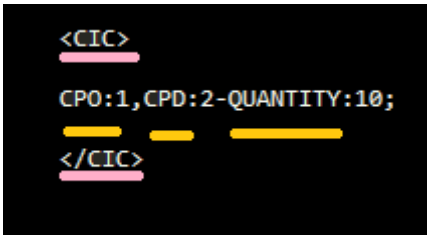


Ilustración 64 - Definición de un enlace

Como podemos ver en la Figura 29, está definido entre las directivas <CIC> y </CIC >

Dentro de ella definimos cada enlace de la siguiente forma:

- Código puntos origen: "CPO:" seguida del código de puntos de la central origen, seguido una coma.
- Código de puntos de la central destino: (CPD:) igual que la anterior y seguido de un guion.
- Número de circuitos: utilizamos la directiva QUANTITY seguida de dos puntos y especificamos el número de circuitos en ese enlace. Por último un ;.

Una vez definido el archivo de configuración, ya podemos pasar a utilizar el programa.

12.2.3 Usando el programa

La pantalla principal del programa es la siguiente:

Ilustración 65 - Interfaz Principal

Donde podemos ver las distintas funcionalidades de éste. Pasaremos a explicar cada una de ellas.

12.2.3.1 Recargar archivo de configuración.

En el menú de desplegable, Inicio->RecargarArchivoDeConfiguracion, podemos cambiar el archivo de configuración cada vez que deseemos. También en el icono justo de debajo representado en la Figura 31.

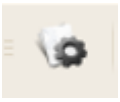


Ilustración 66 - Icono Configuración

Si se ha cargado correctamente nos saldrá un mensaje informándonos. Si por el contrario existiese algún error, nos informaría de error exacto y el número de línea en el que se encuentra.

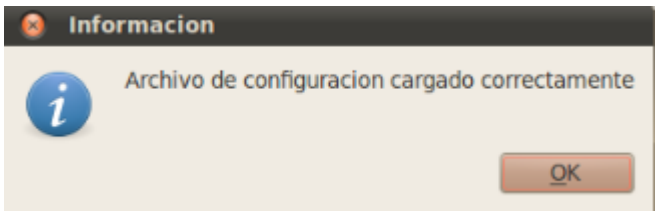


Ilustración 67 - Mensaje información archivo configuración

12.2.3.2 Listado telefónico.

Si queremos ver el listado telefónico sólo tenemos que pulsar el botón listar. Esto son los teléfonos existentes a los que podemos llamar, los cuáles hemos definido en el archivo de configuración.

Ilustración 68 - Listado Telefónico

12.2.3.3 Escenario

Podemos ver el escenario que hemos definido, desplegado o sin desplegar, según la opción que elijamos. Tanto en el menú desplegable como en los iconos.

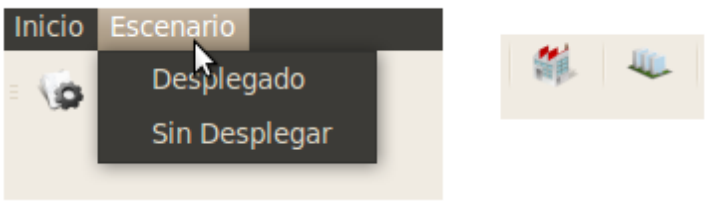


Ilustración 69 - Menú escenario e iconos.

Un ejemplo de escenario que nos muestra es el siguiente, donde podemos mover cualquier nodo. Las líneas negras representan enlaces de datos, y la verdes discontinuas, enlaces de señalización.

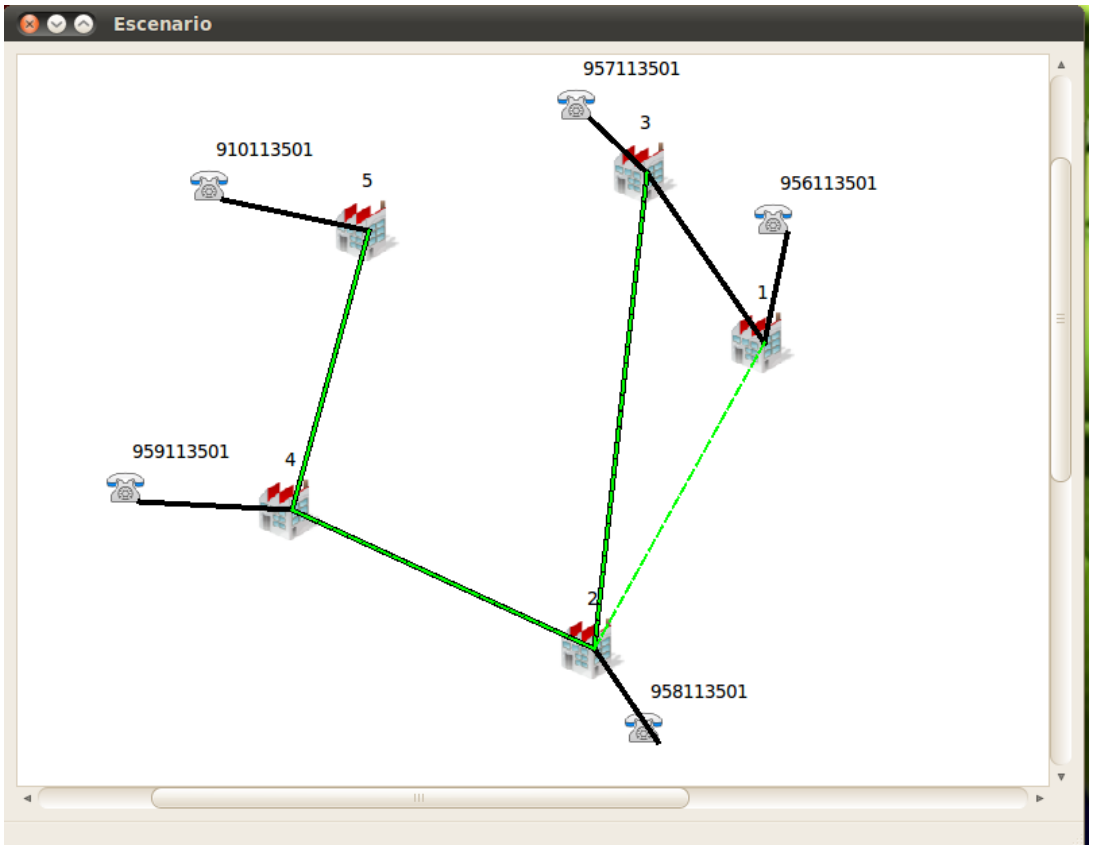


Ilustración 70 - Escenario Desplegado

12.2.3.4 Llamar

Si queremos realizar una llamada solo tenemos que introducir el origen y el destino y pulsar el botón llamar.

Ilustración 71 - Insertar origen y destino

Si alguno no es correcto, nos saldrá un mensaje de información alertándonos.

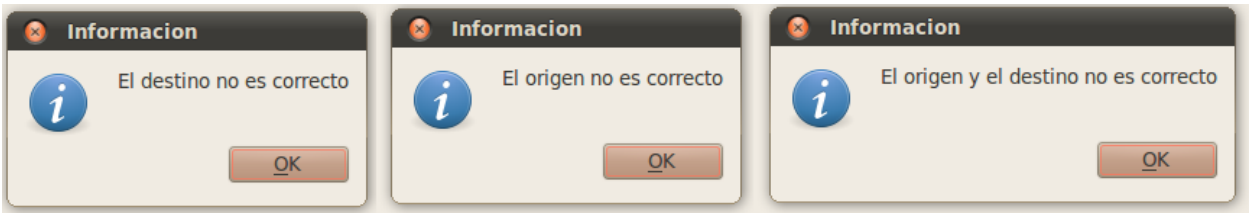


Ilustración 72 - Mensajes error, introducir origen y destino

Si todo es correcto se mostrará el paso de mensaje del establecimiento de llamada. Si hay circuitos suficientes, se podrá realizar la llamada, y tendremos opción a colgar, tanto el llamante como el llamado. De nuevo, mostrándonos un paso de mensajes, pero ahora de la liberación de la llamada. Hasta que no colguemos, no tendremos opción a cerrar esta ventana. Los identificadores de circuitos corresponden al número de circuitos existentes en ese momento, por lo que si volvemos a hacer otra llamada antes de colgar "cic" será un número menor. Podemos verlo en las dos siguientes imágenes.

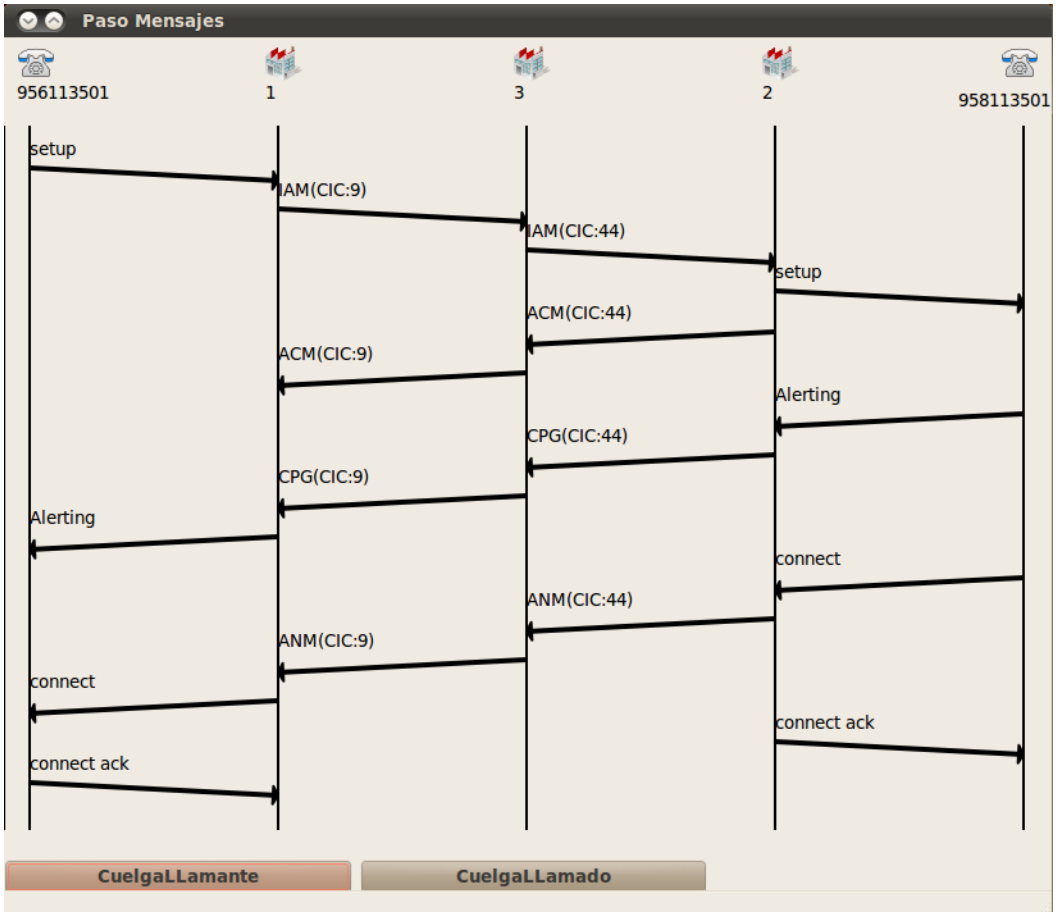


Ilustración 73 - Paso mensajes establecimiento llamada 1

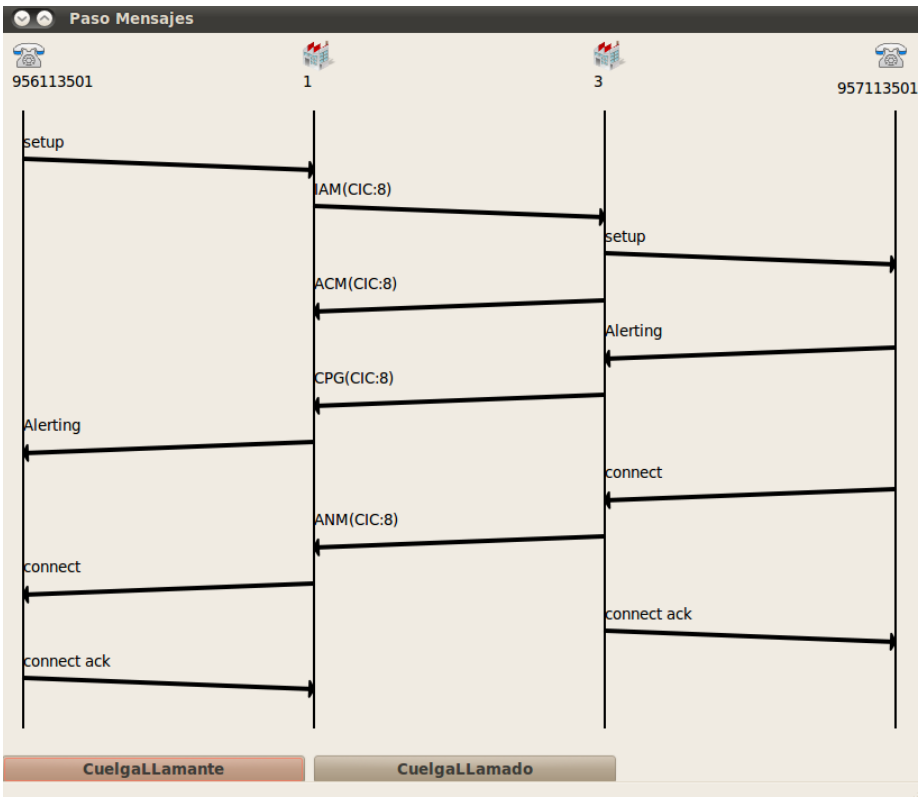


Ilustración 74 - Paso mensajes establecimiento llamada 2

Por el contrario, si la llamada no se estableció con éxito, obviamente no tendremos opción a colgar. Y un ejemplo sería el siguiente, donde no hay circuitos existentes en el enlace entre las centrales con código de puntos 3 y 2.

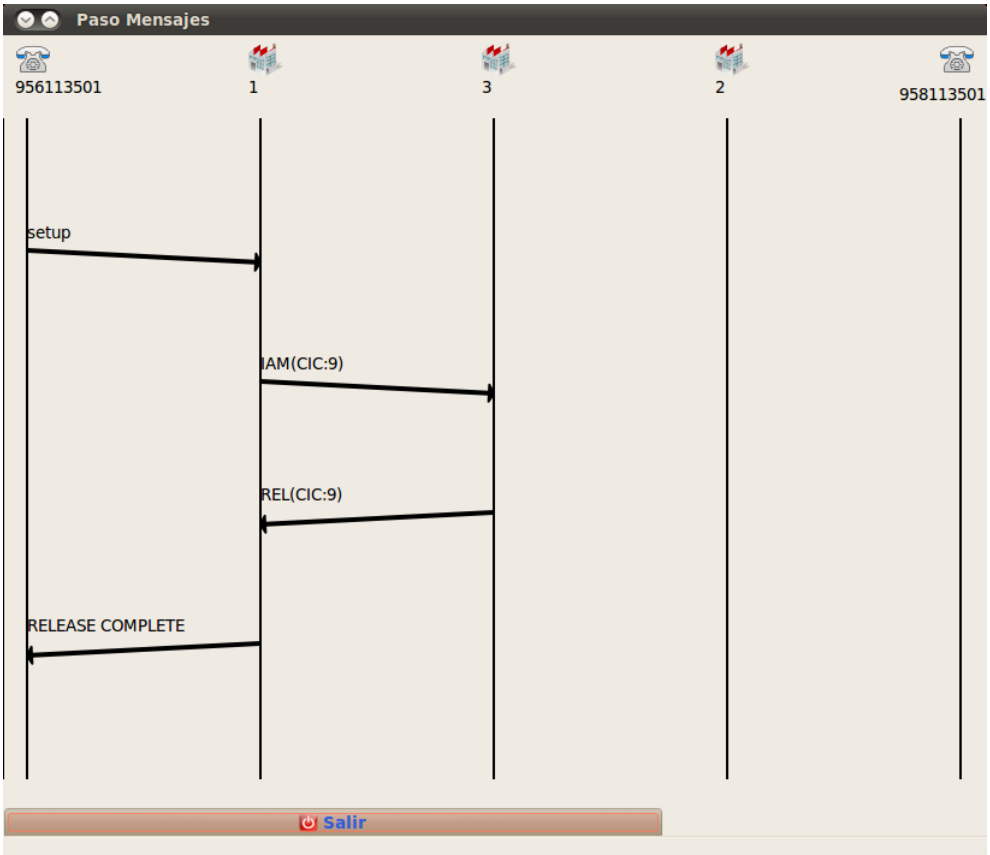
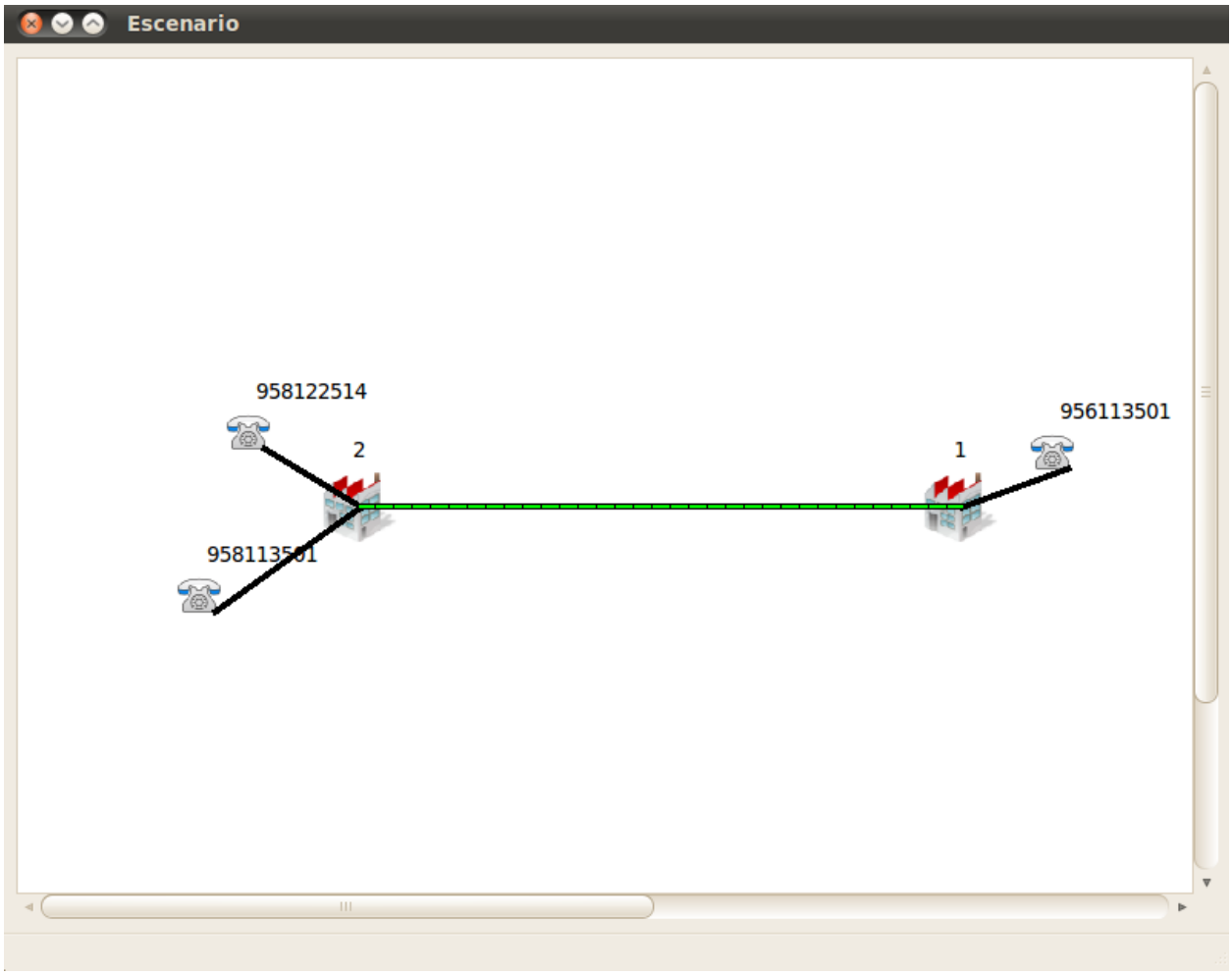


Ilustración 75 - Diagrama llamada fallida

12.2.4 Ejemplos prácticos

En este apartado dejamos dos archivos de configuración, con los que probar, junto con el esquema del escenario que definen.

12.2.4.1 Ejemplo 1




```
<CENTRAL SP>
CODE_POINT 1
PREFIX 956
</PHONE
113501;
>
```

```
</ROUTING TABLE
DEST: 2 - NEXTHOP:2;
>
```

```
<SIGNALING TABLE
DEST:2 - NEXTHOP: 2;
>
</CENTRAL>

<CENTRAL SP>
CODE_POINT 2
PREFIX 958
</PHONE
113501;
122514;
>
</ROUTING TABLE
DEST: 1 - NEXTHOP:1;
>
<SIGNALING TABLE
DEST: 1 - NEXTHOP:1;
>
</CENTRAL>


<CIC>
CPO:1,CPD:2-QUANTITY:10;
</CIC>
```

12.2.4.2 Ejemplo 2

Escenario definido:

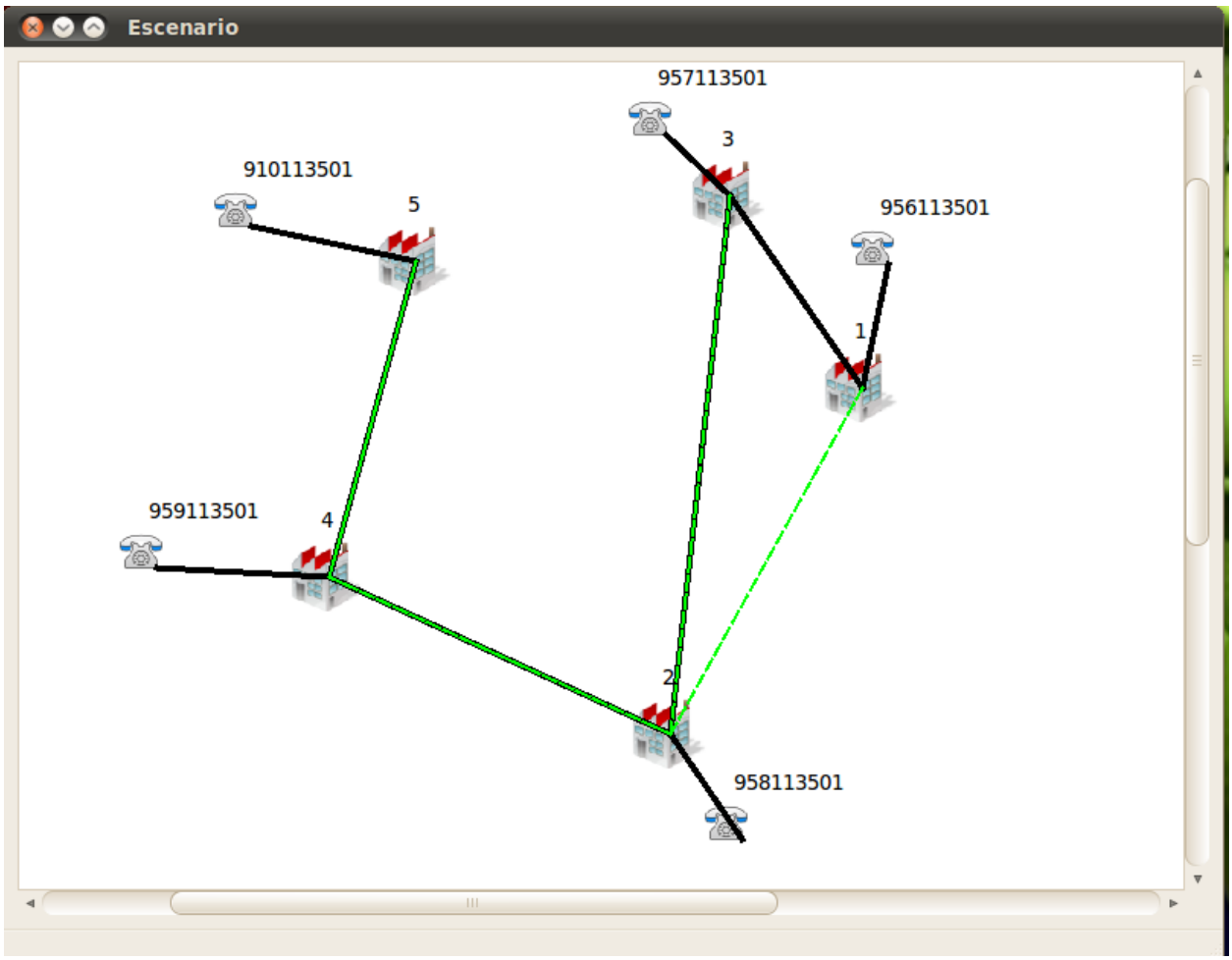


Ilustración 76 - Escenario Ejemplo

Archivo de configuración:

```
<CENTRAL SP>
CODE_POINT 1
PREFIX 956
</PHONE
113501;
>
</ROUTING TABLE
DEST: 2 - NEXTHOP:3;
DEST: 3 - NEXTHOP:3;
DEST: 4 - NEXTHOP:3;
DEST: 5 - NEXTHOP:3;
```

```
>
<SIGNALING TABLE
DEST:2 - NEXTHOP: 2;
DEST: 3 - NEXTHOP:2;
DEST: 4 - NEXTHOP:2;
DEST: 5 - NEXTHOP:2;
>
</CENTRAL>

<CENTRAL SP>
CODE_POINT 2
PREFIX 958
</PHONE
113501;
>
</ROUTING TABLE
DEST: 1 - NEXTHOP:3;
DEST: 3 - NEXTHOP:3;
DEST: 4 - NEXTHOP:4;
DEST: 5 - NEXTHOP:4;
>
<SIGNALING TABLE
DEST: 1 - NEXTHOP:1;
DEST: 3 - NEXTHOP:3;
DEST: 4 - NEXTHOP:4;
DEST: 5 - NEXTHOP:4;
>
</CENTRAL>

<CENTRAL SP  >
CODE_POINT 3
PREFIX 957
</PHONE
113501;
>
</ROUTING TABLE
DEST: 1 - NEXTHOP:1;
DEST: 2 - NEXTHOP:2;
DEST: 4 - NEXTHOP:2;
```

```
DEST: 5 - NEXTHOP:2;
>

<SIGNALING TABLE
DEST: 1 - NEXTHOP:2;
DEST: 2 - NEXTHOP:2;
DEST: 4 - NEXTHOP:2;
DEST: 5 - NEXTHOP:2;
>
</CENTRAL>

<CENTRAL SP  >
CODE_POINT 4
PREFIX 959
</PHONE
113501;
>
</ROUTING TABLE
DEST: 1 - NEXTHOP:2;
DEST: 2 - NEXTHOP:2;
DEST: 3 - NEXTHOP:2;
DEST: 5 - NEXTHOP:5;
>
<SIGNALING TABLE
DEST: 1 - NEXTHOP:2;
DEST: 2 - NEXTHOP:2;
DEST: 3 - NEXTHOP:2;
DEST: 5 - NEXTHOP:5;
>
</CENTRAL>

<CENTRAL SP  >
CODE_POINT 5
PREFIX 910
</PHONE
113501;
>
</ROUTING TABLE
DEST: 1 - NEXTHOP:4;
```

```
DEST: 2 - NEXTHOP:4;
DEST: 3 - NEXTHOP:4;
DEST: 4 - NEXTHOP:4;
>
<SIGNALING TABLE
DEST: 1 - NEXTHOP:4;
DEST: 2 - NEXTHOP:4;
DEST: 3 - NEXTHOP:4;
DEST: 4 - NEXTHOP:4;
>
</CENTRAL>

<CIC>
CPO:1,CPD:3-QUANTITY:10;
CPO:3,CPD:2-QUANTITY:45;
CPO:2,CPD:4-QUANTITY:21;
CPO:4,CPD:5-QUANTITY:20;
</CIC>
```