ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE GRADO

TÍTULO: Generación automática de ficheros de configuración para encaminadores**.**

AUTOR: Alberto Sarró Mora**.**

TITULACIÓN: Grado en Ingeniería Telemática**.**

TUTOR (o Director en su caso): Javier Martín Rueda.

DEPARTAMENTO: Departamento de Ingeniería Telemática y Electrónica (DTE)

VºBº

Miembros del Tribunal Calificador:

PRESIDENTE: Ignacio Álvarez Rocha…….

VOCAL: Javier Martín Rueda…….

SECRETARIO: Rubén de Diego Martínez…….

Fecha de lectura:

Calificación:

El Secretario,

Agradecimientos

Me gustaría dar las gracias a todas aquellas personas que han aportado su granito de arena para hacerme llegar hasta aquí.

En primer lugar, me gustaría agradecer a mis padres, José Luis e Inma, los infinitos esfuerzos que han hecho por darme la mejor educación posible, tanto a nivel personal como a nivel intelectual. Gracias por el apoyo y el cariño, por estar dispuestos siempre a ofrecerme vuestra mano, por no faltarme nunca, vosotros sois el pilar fundamental en el que se cimienta mi vida. Estoy muy orgulloso de vosotros.

A mi hermano, David, referente desde que tengo uso de razón, mi mejor ejemplo, mi mejor amigo. Gracias por todas esas tardes en las que dejabas de hacer tus cosas por ayudarme con los problemas de matemáticas o física, por todo lo que me has enseñado y aportado, por todos los caminos que has abierto por mí sin pedir nada a cambio.

A mi pareja, Jenni, por ser desde que la conocí mi mejor apoyo, mi luz. Gracias por aportarme todas las virtudes que tienes, por tener la palabra exacta para cada momento, por completarme, por hacerme mejor, por apoyarme sin importar nada más, por la confianza y por aguantarme. Y en especial, muchas gracias por todo lo que me has apoyado tanto durante la carrera como durante el desarrollo de este proyecto, sin ti todo hubiera sido muchísimo más difícil. Mil gracias.

A mi cuñada, Susana, por estar siempre que la he necesitado, por el cariño que me ha dado desde el día que la conocí.

A mi sobrina, Naira, por regalarme esa sonrisa que me ha ayudado a seguir en los días más complicados. Por no enfadarse cuando en lugar de jugar y bailar me ponía a hacer fichas.

A mis abuelos, Paulino y Pepe y, en especial, quiero agradecer a mi abuela Flora todos y cada uno de los minutos de los que pude disfrutar de ella. Gracias por haber sido la mejor abuela, por haberme cuidado junto al abuelo como sólo vosotros sabíais, por los veranos en la playa. Gracias por haberme enseñado una manera de querer totalmente diferente a lo que conocía, por todo lo que, sin que nadie lo supiera, he aprendido durante estos años que no han sido tan fáciles, por lo que me has hecho crecer como persona, por lo que soy hoy y seré mañana. Gracias por ser parte de mí.

A mi amigo Dani, por compartir conmigo todas las vivencias, buenas y malas, en la universidad. Por todas las veces que me ha ayudado o lo ha intentado. Por ser una persona excepcional y por ser el mejor compañero de viaje que he podido tener durante la carrera.

A mi tutor, Javier, por los consejos que me ha ofrecido durante todo el desarrollo de este trabajo los cuales han sido de gran ayuda.

A mi compañera Pilar, por ser mi guía durante este proyecto. Por enseñarme y explicarme con paciencia, por sacar un hueco incluso en los días de más trabajo para ayudarme.

Muchas gracias a todos porque, sin vosotros, no hubiera sido capaz.

Resumen

Este Proyecto Fin de Grado (PFG) tiene como objetivo diseñar e implementar un sistema que genere un fichero de texto que contenga la configuración básica de un encaminador.

De esta manera se desea mejorar la eficiencia del personal del departamento donde se va a implantar dicho sistema, liberando a los miembros del mismo de un trabajo repetitivo que se hace varias veces al día. Hasta ahora, esta configuración la realiza cada instalador. Para ello, una vez que se ha cargado y probado la configuración en distintos equipos de manera satisfactoria, se generan un conjunto de plantillas que sirven de modelo para las siguientes configuraciones. Aunque el instalador toma estas plantillas como punto de partida, tiene que modificar manualmente todas las variables que dependen de cada configuración particular. Por tanto, aunque no ha de ejecutar todos los comandos paso a paso, sí debe hacer una revisión total de cada plantilla para generar la configuración adecuada y después cargarla en el encaminador.

Para cada configuración se consultan un total de entre tres y siete plantillas. Si a esto se añade que en el departamento se configuran encaminadores de la marca Cisco y Teldat, que de cada marca se utilizan distintos modelos y que la empresa ofrece cuatro tipos de servicio, cada uno con sus particularidades, la tarea de configurar un equipo es costosa.

El sistema estará constituido por un servidor web que alojará una base de datos y un programa que permite realizar operaciones de consulta sobre la misma, un sitio web sencillo que hará las funciones de interfaz de usuario y una aplicación que permite generar el fichero de texto que contiene la configuración del encaminador en base a una serie de condicionantes.

La base de datos desarrollada es una representación de la utilizada en el entorno real que tiene como objetivo realizar simulaciones del funcionamiento que tendrá el sistema. Por su parte, la funcionalidad del sitio web debe ser la de ofrecer al usuario una interfaz sencilla de utilizar y de interpretar, a través de la cual se puedan realizar consultas a la base de datos así como presentar los resultados de dichas consultas de forma ordenada. La aplicación se encargará de validar los datos a partir de los que se va a generar la configuración, determinar qué plantillas se deben consultar en función a aspectos como el servicio a configurar o la marca del encaminador y finalmente generar el fichero de texto resultado. De este modo, el instalador simplemente tendrá que volcar la información de dicho fichero sobre el encaminador.

El sistema se ha diseñado de manera que sea lo más flexible a cambios, puesto que la idea de los miembros del departamento es ampliar la funcionalidad de esta herramienta.

Abstract

This Final Degree Project is focused on the design and implementation of a system which is able to generate a text file that contains the basic configuration of a router. With this system we want to improve the efficiency of the department members where this system is going to be introduced, releasing them from repetitive work which is done several times per day. Up to now, each installer has to configure the router manually. After checking the configuration of several devices successfully, they create a set of templates which work as models. Although the installers use those templates, they have to modify the variables that depend on the specific features of each kind of configuration. Thus, even though they don´t have to execute the commands step by step, they have to do a review of each used template in order to generate the right configuration. For each configuration, three to seven templates have to be checked. In addition, if the configured routers are both Cisco and Teldat, there are several models per brand and the company offers four types of services to be installed, so the configuration becomes a hard task to do.

The system is comprised of a web server in which both the database and the program responsible for doing queries are hosted, a simple web site that will be the graphic user interface, and an application focused on generating the text file which contains the router configuration based on a set of conditions.

The developed database is the representation of the real one and its aim is to simulate the way the system will work. The function of the web site is to offer an easy interface whereby you can submit a query or you can see the obtained results as a data table. Furthermore, the application has to validate the data in which the text file with the router configuration is based on. Then, it has to decide which templates it is going to use according to different aspects, such as the brand of the router or the type of service we want to configure. Finally, the application generates a text file with the necessary commands. As a result of this, the user of the system only has to copy the contents of this file to the router.

The system has been designed to be flexible to changes because the members of the department want to increase the utility of this tool in the future.

Índice de contenidos

Índice de figuras

Lista de Siglas y Acrónimos

ACK: Acknowledgement

ADSL: Asymmetric Digital Subscriber Line

AES: Advanced Encryption Standard

AON: Active Optical Networks

AP: Access Point

API: Application Programming Interface

APN: Access Point Name

ARP: Address Resolution Protocol

AS: Autonomous System

ASP: Active Server Pages

ATM: Asynchronous Transfer Mode

BGP: Border Gateway Protocol

BSS: Basic Service Set

BSSID: Basic Service Set Identifier

BT: British Telecom

CIDR: Classless Inter-Domain Routing

CNMC: Comisión Nacional de los Mercados y la Competencia

CPE: Customer Premises Equipment

CSED: Customer Solutions and Engineering Delivery

CSS: Cascading Style Sheets

CSV: Comma-Separated Values

CT: Componente Técnico

DES: Data Encryption Standard

DNS: Domain Name System

DOCSIS: Data Over Cable Service Interface Specification

DOM: Document Object Model

DS: Distribution System

DSLAM: Digital Subscriber Line Access Multiplexer

DVB-IP: Digital Video Broadcasting- Internet Protocol

DVB-RCS: Digital Video Broadcasting- Return Channel Satellite

EDGE: Enhanced Data rates for GSM of Evolution

EIGRP: Enhanced Interior Gateway Routing Protocol

ESS: Extended Service Set

ESSID: Extended Service Set Identifier

FTP: File Transfer Protocol

FTTB: Fiber To The Building

FTTH: Fiber To The Home

FTTN: Fiber To The Node

GAFC: Generador Automático de Ficheros de Configuración

Gbps: Giga bits por segundo

GMSK: Gaussian Minimum Shift Keying

GPON: Gigabit-capable Passive Optical Networks

GPRS: General Packet Radio Service

GSM: Global System for Mobile communications

HDSL: High bit-rate Digital Subscriber Line

HFC: Hybrid Fiber Coaxial

HSDPA: High Speed Downlink Packet Access

HSUPA: High-Speed Uplink Packet Access

HTML: HyperText Markup Language

HTTP: HyperText Transfer Protocol

ICMP: Internet Control Message Protocol

IEEE: Institute of Electrical and Electronics Engineers

IGMP: Internet Group Management Protocol

IGRP: Interior Gateway Routing Protocol

IIS: Internet Information Services

INE: Instituto Nacional de Estadística

IP: Internet Protocol

ISO: International Organization for Standardization

ISP: Internet Service Provider

ITU-T: International Telecommunications Union- Telecommunication Standardization

JSON: JavaScript Object Notation

Kbps: Kilobits por segundo

LAN: Local Area Network

LLC: Logical Link Control

LOS: Line Of Sight

LTE: Long Term Evolution

MAC: Media Access Control

MAN: Metropolitan Area Network

Mbps: Megabits por segundo

MDU: Multi Dwelling Unit

MMF: Multimode Fiber

NEBA: Nuevo servicio Ethernet de Banda Ancha

NFC: Near Field Communication

NLOS: Non Line Of Sight

ODN: Optical Distribution Network

OFDM: Orthogonal Frequency Division Multiplexing

OFDMA: Orthogonal Frequency Division Multiple Access

OLT: Optical Line Termination

ONT: Optical Network Termination

ONU: Optical Network Unit

OSI: Open Systems Interconnection

OSPF: Open Shortest Path First

PAN: Personal Area Network

PCMCIA: Personal Computer Memory Card International Association

PFG: Proyecto Fin de Grado

PHP: Hypertext Preprocessor

PLC: Power Line Communications

PON: Passive Optical Network

PPP: Point-to-Point Protocol

PPPoA: Point-to-Point Protocol Over ATM

PPPoE: Point-to-Point Protocol Over Ethernet

PSK: Phase Shift Keying

RADIUS: Remote Authentication Dial-In User Service

RDSL: Rate Adaptative Digital Subscriber Line

RIP: Routing Information Protocol

RJ: Registered Jack

RTC: Red Telefónica Conmutada

SAP: Service Access Point

SC-FDMA: Single Frequency Division Multiple Access

SDSL: Symetric Digital Subscriber Line

SIM: Subscriber Identity Module

SMF: Single Mode Fiber

SMTP: Simple Mail Transfer Protocol

SNMP: Simple Network Management Protocol

SQL: Structured Query Language

SSID: Service Set Identifier

TCP: Transmission Control Protocol

TCP-IP: Transmission Control Protocol-Internet Protocol

UDP: User Datagram Protocol**UML**: Unified Modeling Language

URL: Uniform Resource Locator

USB: Universal Serial Bus

VCI: Virtual Channel Identifier

VDSL: Very High Bit Rate Digital Subscriber Line

VLAN: Virtual Local Area Network

VP: Virtual Path

VPI: Virtual Path Identifier

VPN: Virtual Private Network

WAN: Wide Area Network

WCDMA: Wideband Code Division Multiple Access

WiFi: Wireless Fidelity

XAMPP: Apache MySQL PHP Perl

XML: EXtensible Markup Language

XSD: XML Schema Definition

Glosario de términos

.class: archivo cuyo contenido es el código compilado que entiende la máquina virtual Java.

.error: archivo de texto plano que contiene los errores producidos durante la ejecución de la aplicación Java.

.java: archivo que contiene el código fuente de los programas escritos en lenguaje Java.

.txt: archivos de texto en los que se almacena información en texto plano.

Acknowledgement o acuse de recibo (ACK): en redes de comunicaciones, son los mensajes de acuse de recibo que el receptor envía al emisor para notificarle que ha recibido los datos.

ActiveX: se define como "*pequeños programas que se pueden incluir dentro de páginas web y sirven para realizar acciones de diversa índole*" [1]. Sólo funcionan en sistemas operativos Windows.

Apache: servidor web de código abierto y multiplataforma que permite al usuario trabajar desde distintos sistemas operativos en la creación de sitios web.

Array: estructura de datos que permite almacenar un conjunto finito de valores del mismo tipo. Cada uno de estos valores se identifica como elemento.

Backhaul: se denomina así a la red que permite interconectar la red troncal con los nodos que ofrecen servicio a los usuarios.

Backup: copia de seguridad que se hace de determinados datos o archivos con el fin de tenerlos redundados por si se perdiera la copia original. En este documento se hace referencia a encaminadores *backup*, los cuales actúan como respaldo de los principales, es decir, en caso de que en los primeros surja algún tipo de fallo, el tráfico se balancea hacia los encaminadores *backup* de manera que el cliente no se ve afectado.

Best-Effort (BE): es el tipo de servicio por defecto con el que se marcan los paquetes en las redes de datos. Este servicio no garantiza ninguna calidad de servicio de manera que en caso que exista congestión, estos serán los primeros paquetes descartados.

Bonding: mediante esta palabra se define la capacidad de utilizar varios canales de trasmisión de datos simultáneamente, tanto de bajada como de subida, lo que supone un aumento de la velocidad.

Buffer: memoria en la que se almacenan paquetes de manera temporal a la espera de ser procesados. Su uso cobra sentido cuando existe congestión en la red, de manera que no se puedan procesar todos los paquetes que llegan en un momento dado.

Char: tipo de dato empleado en *Structured Query Language* o Lenguaje de Consulta Estructurado SQL que define una cadena de caracteres de longitud fija.

Checkpoints: Puntos de control que, en caso de que la conexión entre dos equipos se interrumpa por algún motivo, permite que sólo se envíen los datos que faltaban para completar la comunicación desde el último punto de control establecido.

Date: tipo de dato empleado en SQL que define una fecha cuyo formato es AAAA-MM-DD.

Gateway: en redes de comunicaciones, es el encaminador encargado de comunicar dos redes cuyos protocolos son diferentes. Gracias a este dispositivo puede haber intercambio de paquetes entre dichas redes.

Gaussian Minimum Shift Keying o Modulación por Desplazamiento Mínimo Gaussiano (GMSK): "*Sistema de modulación digital en el que cada bit o grupo de bits produce una variación determinada en la fase de la onda portadora*" [2].

Hop: dirección *Internet Protocol* o Protocolo de Internet (IP) que, en una red de comunicaciones compuesta por varios encaminadores en la que se envía un paquete de datos desde un origen hacia un destino, identifica el siguiente encaminador de todos a los que se puede alcanzar que suponga el mejor camino para alcanzar dicho destino.

Hub: dispositivo que actúa en la capa física (nivel 1) del modelo *Open Systems Interconnection* o Interconexión de Sistemas Abiertos (OSI) y que permite interconectar varios equipos dentro de una misma red.

Int: tipo de dato empleado en SQL que define un número entero comprendido entre -2147483648 y 2147483647 si se aplica el signo o entre 0 y 4294967295 en caso de no aplicarse.

Mediumint: tipo de dato empleado en SQL que define un número entero comprendido entre -8388608 y 8388607 si se aplica el signo o entre 0 y 16777215 en caso de no aplicarse.

MyISAM: motor de almacenamiento que se utiliza en aquellas bases de datos en las que no se necesita modificación concurrente de datos. Por tanto, un entorno en el que se realicen consultas de lectura de datos es el ideal para este motor.

MySQL: sistema de gestión de bases de datos de código abierto.

Nuevo servicio Ethernet de Banda Ancha (NEBA): servicio mayorista que sustituirá a los servicios de acceso indirecto actuales, GigADSL y ADSL-IP, y que permitirá que todos los operadores puedan dar servicios de banda ancha en todas las zonas geográficas de manera independiente. Es decir, permite que los operadores alternativos tengan más flexibilidad a la hora de configurar sus ofertas comerciales. Hasta hace poco, los operadores alternativos sólo tenían dos maneras de ofrecer servicios de banda ancha: alquilar el bucle de abonado en la central de Telefónica, método que sí permite configurar ofertas comerciales competitivas, o contratar los servicios mayoristas de acceso indirecto que permiten ofrecer servicios de banda ancha más allá de la central. En zonas donde el alquiler del bucle sea poco rentable los alternativos sólo pueden contratar servicios mayoristas que, además de ser más caros, no permiten margen de maniobra. Sin embargo, a partir de la llegada de NEBA los operadores alternativos pueden contratar un ancho de banda determinado, en función de sus necesidades, que les permite competir tanto en términos de calidad como económicos puesto que pueden configurar las ofertas comerciales como deseen.

Null: tipo de dato que sirve para determinar que el valor de la variable es desconocido o inexistente, es decir, no representa una cadena vacía o un cero, sino un valor indefinido.

Open Source: término que se le aplica a aquellos programas distribuidos bajo licencia que permiten al usuario acceder al código fuente de los mismos y modificarlo. No confundir este término con "*software* libre", que hace referencia a programas obtenidos de manera gratuita.

Orthogonal Frequency Division Multiple Access o Acceso Múltiple por División de Frecuencias Ortogonales (OFDMA): "*Técnica de modulación consistente en múltiples subportadoras ortogonales entre sí y que se modulan digitalmente de forma convencional, por ejemplo, una modulación de amplitud en cuadratura, con una baja velocidad de transmisión; ofrece gran protección frente al desvanecimiento multitrayecto*" [3].

Parsear: análisis sintáctico que se realiza sobre un texto con el objetivo de determinar si cumple con una serie de reglas definidas previamente.

Password: contraseña. Conjunto de caracteres que, junto a un usuario, permite a este autenticarse contra un determinado sistema o recurso.

Phase Shift Keying o Modulación por Desplazamiento de Fase (PSK): "*Sistema de modulación digital en el que cada bit o grupo de bits produce una variación determinada en la fase de la onda portadora*" [4].

phpMyAdmin: programa diseñado para administrar y gestionar las bases de datos *MySQL* a través de una interfaz web.

Rate: tasa que indica el ancho de banda del puerto contratado. En este documento se suele expresar en Kbps.

Router: dispositivo que actúa en la capa de red (nivel 3) del modelo OSI y que permite la interconexión entre dos o más redes.

Routing: proceso por el cual se determinan las tablas de encaminamiento utilizadas para elegir el mejor camino que debe seguir un paquete en una red de comunicaciones desde un origen hasta un destino.

Script: conjunto de instrucciones almacenadas en un fichero de texto que permiten la automatización de tareas. Estos ficheros no deben ser compilados antes de ser ejecutados.

Shell: programa que actúa como interfaz de usuario para permitir al usuario comunicarse con el sistema operativo. Pueden ser interfaces gráficos o de texto libre.

Single Frequency Division Multiple Access o Acceso Múltiple por División de Frecuencia Portadora Única (SC-FDMA): "*Tecnología de acceso usada en el enlace ascendente de los sistemas móviles Long Term Evolution o Evolución a Largo Plazo (LTE) que proporciona una buena eficiencia de potencia*" [5].

Smallint: tipo de dato empleado en SQL que define un número entero comprendido entre -32768 y 32767 si se aplica el signo o entre 0 y 65535 en caso de no aplicarse.

Software: según la Real Academia Española es un: *"conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora"* [6].

Splitter: dispositivo capaz de separar las señales de voz de las de datos.

Switch: dispositivo que actúa sobre la capa de enlace de datos (nivel 2) del modelo OSI y que permite interconectar un conjunto de equipos formando una red local.

Tinyint: tipo de dato empleado en SQL que define un número entero comprendido entre -128 y 127 si se aplica el signo o entre 0 y 255 en caso de no aplicarse.

Unsigned: atributo empleado en SQL que indica que sólo se permiten valores positivos para el dato numérico indicado.

Varchar: tipo de dato empleado en SQL que define una cadena de caracteres de longitud variable. Se puede fijar el máximo de la misma, siempre y cuando esté comprendido entre 1 y 8000 caracteres.

Wildcard mask: secuencia de 32 bits que indica qué bits de una dirección IP son relevantes a la hora de realizar alguna acción sobre la misma. Los bits significativos son los ceros, es decir, son los bits que deben ser iguales, mientras que los unos son los bits no significativos, es decir, aquellos que se pueden ignorar.

CAPÍTULO 1

INTRODUCCIÓN Y OBJETIVOS

1. Introducción y objetivos

1.1.Introducción

El departamento en el que nace la idea de desarrollar el sistema se encuentra en la parte de entrega, por lo que es un eslabón más de la cadena encargada no sólo de proporcionar los servicios contratados al cliente sino además de su satisfacción con los mismos. Desde este departamento se realizan las configuraciones necesarias en los equipos de cliente para dar los servicios tanto de voz como de datos que ofrece la empresa.

Los clientes son, a su vez, otras empresas cuyas actividades están cada día más extendidas a nivel global gracias, entre otras cosas, a Internet. En la actualidad, prácticamente el 100% de las empresas dispone de acceso a Internet, lo que ha supuesto un crecimiento del mercado que ha contribuido a la evolución de los accesos a la red, tanto minoristas (usuarios finales), como mayoristas (operadores).

Este proyecto se centra en la provisión de los servicios de datos, dejando a un lado la voz, de manera que se tratará la forma de configurar los encaminadores de cliente en función de los diferentes accesos ofrecidos, entre los que destacan: *Asymmetric Digital Subscriber Line* o Línea de Abonado Digital Asimétrica (ADSL), *Virtual Private Network* o Red Privada Virtual (VPN), Nuevo servicio Ethernet de Banda Ancha (NEBA) y 3G. La tendencia hasta este momento era la de realizar las configuraciones de manera manual, es decir, para configurar un encaminador con una serie de parámetros era necesario que una persona realizara las operaciones apropiadas con el objetivo de satisfacer las necesidades del cliente. Hay que tener en cuenta que para cada acceso la configuración es diferente puesto que los comandos y las variables utilizadas no son las mismas, por lo que el instalador debe consultar distintas documentaciones en función al acceso que esté configurando. La consecuencia de este método de trabajo era que a lo largo de la jornada laboral se invertía mucho tiempo en llevar a cabo un trabajo que podía ser automatizado generando un programa, ya que la configuración base aplicada a los encaminadores instalados en cliente es genérica para cada acceso.

Como ya se ha mencionado anteriormente, para la configuración de estos encaminadores se utilizan comandos específicos en función del acceso y del fabricante, que en nuestro caso son Cisco y Teldat. Esto supone que un simple error, como por ejemplo, un número erróneo en una dirección *Internet Protocol* o Protocolo de Internet (IP), o que se escriba una letra indeseada en el nombre de un cliente, dé como resultado una configuración inválida. Por tanto, realizar las configuraciones a mano aumenta la probabilidad de cometer algún error durante el proceso. La consecuencia de esto es que además del tiempo invertido en configurar el encaminador, en los casos en los que se cometen errores, se tiene que analizar el origen del error y subsanarlo, por lo que se emplea mucho tiempo en la ejecución de una tarea que puede realizarse varias veces al día.

Para resolver el problema de eficiencia laboral mencionado, se propuso la realización de un programa capaz de, en función a una serie de parámetros específicos para cada línea de cliente y tipo de acceso, generar un fichero que contuviera la correcta configuración base para dicha línea. Con esta solución, los miembros del departamento que anteriormente invertían tiempo en esta tarea ahora pueden dedicarlo a otros asuntos laborales. Además, como la configuración se genera de manera automática, no se cometen errores ortográficos. El fichero que se obtiene como resultado es el que los instaladores finales copiarán en el encaminador para establecer la configuración base.

Este Proyecto Fin de Grado se centra, por tanto, en el desarrollo de una herramienta mediante la cual los miembros del grupo puedan generar la configuración base de los encaminadores instalados en cliente de una forma sencilla y rápida, independientemente del acceso deseado por cliente, para poder así invertir el tiempo ganado en otras tareas. Además se ha desarrollado paralelamente una herramienta web con la que los miembros del grupo acceden a los datos de la línea a instalar consultando la base de datos y generan el fichero que contiene la configuración del encaminador, todo ello a través de una interfaz sencilla e intuitiva.

Como la base de datos de la empresa es confidencial, también se ha diseñado una base de datos a modo de simulación para poder explicar el funcionamiento de la herramienta web con mejor detalle.

Por otro lado, durante el presente trabajo se indicarán detalles de diseño o de implementación que se han realizado siguiendo directrices de los compañeros, usuarios finales de la aplicación que además tendrán que mantenerla y actualizarla en el futuro.

En conclusión, los principales puntos a destacar son:

- **Problema**
  - Excesiva inversión de tiempo en una tarea frecuente.
  - Alta probabilidad de error debido a posibles fallos humanos.
- **Solución**
  - Desarrollo de un programa en lenguaje Java capaz de generar como resultado un fichero de texto que contiene la correcta configuración base del encaminador.
  - Desarrollo de una herramienta web que permita a los usuarios obtener los datos de una línea y generar el fichero de configuración a través de un interfaz sencillo.

1.2. Objetivos

La idea principal de este Proyecto Fin de Grado (PFG) es diseñar e implementar un programa que genere un fichero que contenga la configuración genérica de los encaminadores utilizados en las sedes de los clientes de la empresa. Para ello, se han de definir una serie de objetivos a cumplimentar:

- Mejorar la eficiencia del trabajo realizado por los miembros del departamento.
- Evitar errores humanos producidos durante el proceso de configuración de los encaminadores.
- Diseñar e implementar un programa en Java capaz de leer las plantillas y añadir a las mismas los parámetros específicos de cada cliente, los cuales se tienen previamente en un fichero con extensión *Comma-Separated Values* o Valores Separados por Coma (CSV).
- Validar los datos de dicho fichero, evitando así configuraciones inválidas.
- Generar un fichero de texto que contenga la configuración necesaria para proporcionar el servicio deseado por el cliente.
- Diseñar e implementar una base de datos sencilla en MySQL que permita realizar una simulación de la base de datos utilizada en el entorno laboral.
- Implementar en *Hypertext Preprocessor* (PHP) el código necesario para obtener el mismo resultado que en el entorno laboral se obtiene a partir de una consulta a la base de datos.
- Implementar en JavaScript la parte de cliente, la cual se encarga de tratar el resultado obtenido tras la consulta a la base de datos, generar una tabla de resultados y ofrecer la posibilidad de generar el fichero de configuración.

1.3. Estructura de la memoria

El presente documento se ha divido en ocho capítulos:

En este primer capítulo se resume la temática del proyecto además de describir los objetivos que se van a abordar.

Durante el segundo capítulo se realiza un trabajo de investigación acerca de las diferentes tecnologías de acceso a Internet que un usuario puede encontrar en el mercado actual.

En el capítulo tres se da una visión resumida de las funciones de cada una de las capas de los modelos de comunicación entre redes además de introducir aspectos teóricos de los distintos dispositivos que actúan en ellas.

El capítulo cuatro se destina a la especificación de requisitos del sistema, donde se realiza una descripción y se redactan los requisitos específicos del mismo.

El capítulo cinco muestra la justificación de las tecnologías utilizadas.

El capítulo seis es en el que se describe en detalle el sistema, explicando cómo se han diseñado e implementado los distintos módulos que lo componen y para qué sirve cada uno de ellos.

Se desarrolla un manual de usuario durante el capítulo siete que tiene como objetivos explicar el funcionamiento del sistema y entender su comportamiento.

Durante el capítulo ocho se describen las conclusiones a las que se ha llegado tras la realización del proyecto y los posibles trabajos futuros que pueden realizarse en base al sistema.

Por último, se detalla la bibliografía utilizada.

CAPÍTULO 2

MARCO TECNOLÓGICO: ACCESOS A LA RED

2. Marco tecnológico. Accesos a la red.

Para poder entender la finalidad de este proyecto y ubicarlo en un marco tecnológico es necesario hacer una investigación sobre algunos aspectos como el uso de Internet por parte de las empresas, los accesos a Internet, el crecimiento de estos durante los últimos años o las diferencias de servicio que supone contratar uno u otro.

Tal y como se mencionaba en la introducción, los clientes de la empresa no son personas físicas sino otras compañías que deciden que su red de comunicaciones sea diseñada, implementada y mantenida por *British Telecom* (BT). Según un estudio realizado por el Instituto Nacional de Estadística (INE), durante el primer trimestre del año 2014 el 98,3% de las empresas españolas de diez o más empleados disponía de conexión a Internet. De estos, prácticamente el 100% utilizaban un acceso de banda ancha, ya fuera fija o móvil [7]. Esta información puede observarse en la figura 1.

Primer Trimestre de 2014	Número de empleados			
	TOTAL	10 a 49	50 a 249	250 ó más
Banda ancha (fija ó móvil)	99,9	99,8	100,0	100,0
Banda ancha fija	98,1	97,9	99,1	99,7
-DSL (ADSL,SDSL,...)	90,4	91,0	87,0	87,9
-Redes de Cable y fibra óptica (FTTH)	21,1	17,0	38,6	67,4
-Otras conexiones fijas (PLC, leased line, satélite...)	5,4	3,7	11,7	28,7
Telefonía móvil de Banda Ancha:	78,3	76,1	89,8	95,3
-Mediante módem 3G o 4G	55,0	51,1	73,7	88,6
-Mediante teléfono móvil 3G o 4G	74,2	71,7	86,4	92,4
Otras conexiones móviles (GPRS, EDGE,...)	23,7	21,0	35,8	50,6

Figura 1: tipo de conexión utilizada por las empresas españolas según número de empleados [8]

Sin embargo, no sólo existen accesos de banda ancha. Como podemos observar en la figura 1, existen otros tipos de tecnologías de acceso entre las que destacan el *Fiber To The Home* o Fibra Hasta el Hogar (FTTH), el *x Digital Subscriber Line (xDSL)* y el acceso por banda ancha móvil, ya sea 3G o 4G, las cuales representan un gran nicho de mercado.

El acceso a elegir depende de varios factores: precio a pagar, velocidad de línea que se desea o incluso posibles restricciones físicas que no permitan instalar la tecnología deseada. Por tanto, para hacer una buena elección es necesario conocer las opciones que tenemos y para ello se ha elaborado la siguiente clasificación de los tipos de acceso:

- **Acceso Telefónico o Dial Up**
- **xDSL**
- **Cable**
- **Fibra Óptica**
- **Redes inalámbricas**
- **Satélite**
- ***Power Line Communication* o Comunicaciones a través de Líneas Eléctricas (PLC)**

2.1. Acceso Telefónico o Dial Up

Sistema pionero de acceso a Internet, utiliza la línea básica telefónica para transmitir voz y datos. Las señales transmitidas y recibidas a través de este acceso son analógicas de manera que es necesario un elemento denominado módem capaz de transformar las señales digitales en analógicas y viceversa. El módem, por tanto, se comporta como enlace entre el ordenador y la red. Una de las principales ventajas de este tipo de acceso fue la escasa inversión que había que hacer para desplegar la red, puesto que la gran mayoría de hogares disponían ya de una línea telefónica. Además de ser un acceso que ofrece un ancho de banda pobre, 56 Kbps que pronto se convirtieron en insuficientes para los servicios ofrecidos en la red, no puede transmitir a la vez voz y datos ya que lo hace a la misma frecuencia. Aunque aún se utiliza en algunos lugares, ha quedado relegada y está en vías de extinción.

2.2. xDSL

Este servicio superó a su antecesor ya que, por un lado, permitió la simultaneidad de voz y datos utilizando frecuencias diferentes para el envío de las señales y, por otro, las velocidades alcanzadas eran mucho mayores que las ofrecidas por la Red Telefónica Conmutada (RTC). Esto, y el hecho de que se pudiera aprovechar la red telefónica para transmitir datos, evitando la inversión de las operadoras en el despliegue de una nueva, fueron factores determinantes para la masiva expansión de este tipo de acceso. Esta simultaneidad es posible gracias a que se utilizan tres canales para la comunicación; dos canales de datos, uno de los cuales se destina a la transmisión (datos salientes) y el otro a la recepción (datos entrantes), y un canal para la voz.

En función del ancho de banda asignado a cada uno de los canales de datos, podemos clasificar las tecnologías xDSL en asimétricas y simétricas [9]:

- **Asimétricas**: las conexiones asimétricas son aquellas en las que el ancho de banda del canal descendiente es mucho mayor que el otorgado al canal ascendente. Estas, a su vez, pueden clasificarse en:
    - **ADSL**: la velocidad ofrecida por los accesos ADSL se encuentra entre 256 Kbps-8 Mbps, en función de una serie de condicionantes entre los que destaca la distancia desde el domicilio del abonado y la central. Las versiones posteriores como ADSL2 y ADSL2+ alcanzan velocidades de hasta 24 Mbps.
    - ***Very High Bit Rate Digital Subscriber Line* o Línea de Abonado Digital de Muy Alta Transferencia (VDSL)**: aunque en esta clasificación se ha decidido incluir este acceso como asimétrico, hay que destacar que el uso del ancho de banda puede ser tanto simétrico, ofreciendo hasta 26 Mbps tanto de bajada como de subida, como asimétrico, ofreciendo hasta 52 Mbps de bajada y 16 Mbps de subida. La consecución de estas velocidades es posible gracias a que se produce un aumento de las bandas de frecuencia empleadas en el transporte de datos. La consecuencia de esto es que, en lugar de utilizar dos canales para la transmisión de datos como sucede en el ADSL, en VDSL se utilizan cuatro, dos de subida y dos de bajada.

La versión posterior, VDSL2, que utiliza un rango de frecuencias aún mayor, ofrece velocidades de descarga de hasta 100 Mbps. Este dato no deja de ser engañoso, puesto que es el resultado de unas condiciones prácticamente ideales en las que el abonado se encuentra a una distancia mínima de la central que le provee el servicio. Por eso, en condiciones normales, este servicio llega al cliente desde un nodo de fibra óptica cercano al hogar u oficina. [10].

A continuación, en la figura 2, se muestra al lector una comparativa de las velocidades ofrecidas por estas dos modalidades de servicio xDSL.

	VEL. MAX. (Subida)	VEL. MAX. (Bajada)	ANCHO DE BANDA
ADSL	8 Mbps	1.0 Mbps	1.1 MHz
ADSL2	12 Mbps	3.5 Mbps	1.1 MHz
ADSL2+	24 Mbps	3.3 Mbps	2.2 MHz
VDSL	52 Mbps	16 Mbps	12 MHz
VDSL2	100 Mbps	50 Mbps / 100 Mbps	17 Mhz / 30 MHz

Figura 2: tabla comparativa de las velocidades ofrecidas por ADSL y VDSL [11]

- ***Rate Adaptative Digital Subscriber Line* o Línea de Abonado Digital de Tasa Adaptativa (RDSL)** el fundamento tecnológico es el mismo que el del ADSL, con la única diferencia de que este tipo de tecnología es capaz de adaptarse dinámicamente a las variaciones producidas en las líneas de forma que ajusta la velocidad ofrecida al usuario en función de la calidad de la señal [12].

- **Simétricas**: en las conexiones simétricas el ancho de banda del canal ascendente es el mismo que el del canal descendente. Dentro de los accesos simétricos encontramos los siguientes:

  - ***High data rate Digital Subscriber Line* o Línea de Abonado Digital de Alta Velocidad (HDSL)**: utilizando técnicas avanzadas de modulación para la transmisión de datos por líneas de pares de cobre trenzados, este acceso es capaz de substituir las líneas T1, de dos pares y utilizadas en países como Estados Unidos o Japón, las cuales ofrecen una velocidad de 1,5 Mbps, y líneas E1, de tres pares y utilizadas en Europa, las cuales ofrecen una velocidad de 2 Mbps. La principal ventaja es que se trata de un acceso barato y de fácil montaje ya que se puede aprovechar el bucle de abonado. Sin embargo, es imposible utilizar de manera simultánea los servicios de voz y de datos, razón por la cual este tipo de acceso fue más utilizado por el sector empresarial que por el sector residencial. Posteriormente se desarrolló la tecnología HDSL2, que era exactamente igual que su antecesora a diferencia de que esta última permitía cubrir distancias mayores.

- ***Symmetric Digital Subscriber Line* o Línea de Abonado Digital Simétrica (SDSL):** Este acceso es semejante en concepto al anterior. No obstante, la característica más destacable de SDSL es que en este tipo de transmisiones tan sólo se utiliza un par, no dos o tres como ocurría en HDSL. Sobre este par pueden transmitirse de manera simultánea tramas T1, E1 y el servicio telefónico. Por otro lado, la distancia máxima de operación se reduce con respecto a HDSL. La velocidad ofrecida no sobrepasa los 1,5 Mbps.

2.3. Cable Coaxial

Haciéndole competencia directa a la tecnología xDSL existe el acceso por cable coaxial, cuyo funcionamiento se asemeja a su competidor en que por el mismo medio, en este caso por el cable coaxial, se pueden transmitir varias señales utilizando frecuencias distintas de modo que es el módem el encargado de separar y repartir las señales adecuadamente. Esta tecnología se conoce también con el nombre *Hybrid Fiber Coaxial* o Híbrido de Fibra-Coaxial (HFC). Este nombre surge por la topología de la red, la cual puede observarse en la figura 3, ya que por un lado tenemos una red de fibra óptica desde las centrales a los nodos situados cerca de los inmuebles, mientras que la conexión entre estos nodos de distribución y el usuario final se realiza mediante cable coaxial. Cada uno de los nodos distribuidores permite la conexión de un número variable de abonados, siendo este número una cifra comprendida entre los 200 y los 500 usuarios [13].

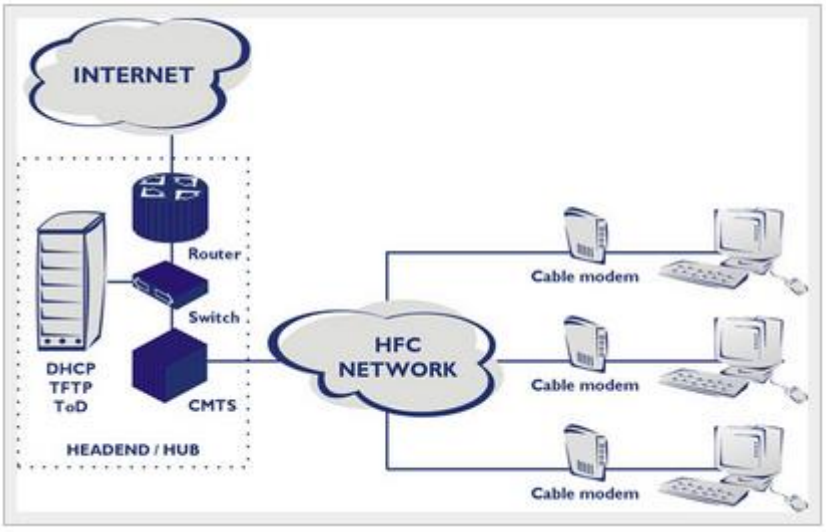

Figura 3. Topología de red HFC [14]

Si el servicio xDSL contaba con frecuencias altas para los datos y bajas para la voz, en la transmisión por cable se pueden utilizar frecuencias variadas, una por cada canal. Es importante mencionar que este tipo de acceso está regulado por el estándar *Data Over Cable Service Interface Specification* o Especificación de Interfaz para Servicios de Datos por Cable (DOCSIS), que define los requisitos que debe cumplir la interfaz de comunicaciones para los sistemas de datos por cable y que cubre todos los elementos de red, desde el *Customer Premises Equipment* o Equipo Local de Cliente (CPE) hasta el terminal del operador [15].

El estándar ha evolucionado desde la versión 1.0 hasta la 3.1, la cual ha salido a la luz en marzo de 2014. A partir de la versión 3.0 es posible utilizar canales en paralelo, lo que se

conoce como *bonding*, cuya consecuencia es que la velocidad total del acceso es la suma de la que proporcionan cada uno de los canales. En DOCSIS 3.0 el ancho de banda de los mismos es de 6 MHz en América y de 8 MHz en Europa y, en función de la modulación utilizada, cada uno de los canales puede transportar hasta un total de 40 Mbps. Según el número de canales ofrecidos por el chipset se puede ofrecer una velocidad de hasta 160 Mbps de bajada y 120 Mbps de subida. Con la nueva versión del estándar se han dejado atrás los canales de 6-8 MHz para pasar a bloques *Orthogonal Frequency Division Multiplexing* o Multiplexación por División de Frecuencias Ortogonales (OFDM) de 192 MHz. Gracias a esto se pueden ofrecer velocidades de hasta 5 Gbps de bajada y 1,5 Gbps de subida [16].

2.4. Fibra óptica (FTTx)

El concepto de la fibra óptica es totalmente diferente al resto de tecnologías vistas hasta el momento. Su característica principal es que las señales eléctricas se convierten en un rayo de luz modulada en el origen de la transmisión, el cual viaja a través de la fibra y en el receptor se vuelve a transformar la señal obteniendo de nuevo señales eléctricas. Para comprender cómo se transmite esta señal luminosa es importante conocer cómo se comporta un haz de luz cuando se transmite de un medio como el aire a un medio como el vidrio.

Para ello, es necesario mencionar varios fundamentos científicos como son el índice de refracción, la ley de Snell y el ángulo crítico [17].

- **Índice de refracción**: el índice de refracción del medio se define como la relación entre la velocidad de la luz en el vacío y la velocidad de la luz en el medio por el que se transmite el haz de luz [18]. La fórmula se presenta en la figura 4.

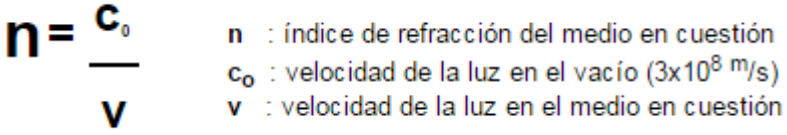

Figura 4. Fórmula del índice de refracción [19]

- **Ley de Snell**: indica el comportamiento de la luz cuando pasa de un medio a otro. Tal y como se puede observar en la figura 5, permite establecer una relación entre los ángulos de incidencia y refracción y los índices de refracción de los medios por los que se propaga la luz.

$$n_1 sen\theta_1 = n_2 sen\theta_2$$

Figura 5. Fórmula de la ley de Snell [20]

El paso de la luz de un medio a otro supone que en la frontera entre ambos se produzca una modificación en la trayectoria del haz debido a que los índices de refracción de los medios son diferentes. Esta modificación puede ser de dos tipos:

  - Por un lado, si el rayo de luz se transmite desde un medio con un índice de refracción menor hacia otro medio con un índice de refracción mayor, entonces el rayo refractado, el que se transmite por el nuevo medio, se acerca a la normal.

- Sin embargo, si la relación anteriormente descrita se da al revés y el índice de refracción del medio origen es mayor que el del medio destino, entonces el rayo refractado se alejará de la normal.

Ambos ejemplos se describen gráficamente en la figura 6:

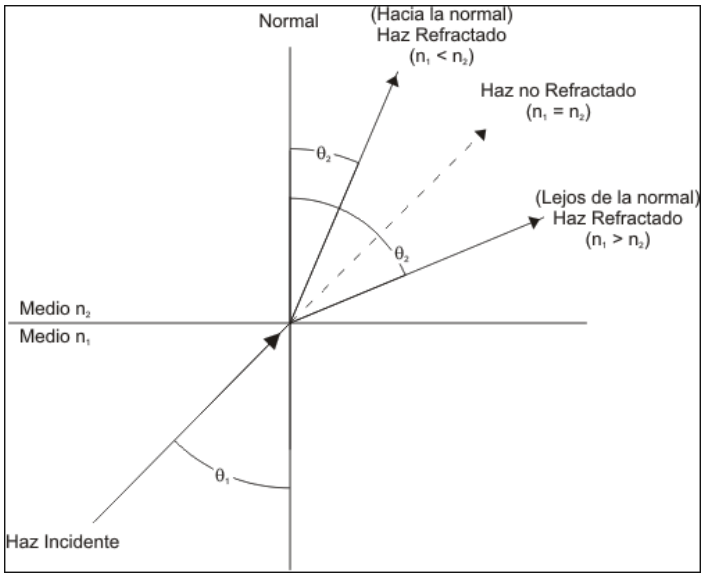

Figura 6. Representación de la ley de Snell [21]

- **Ángulo crítico**: es el ángulo límite con el que el haz de luz podrá incidir sobre la superficie del medio hacia el que se transmite sin que produzca la reflexión total. Si el ángulo con el que el rayo incide es el crítico, el ángulo refractado será de 90º con respecto a la normal, tal y como se puede comprobar en la figura 7.

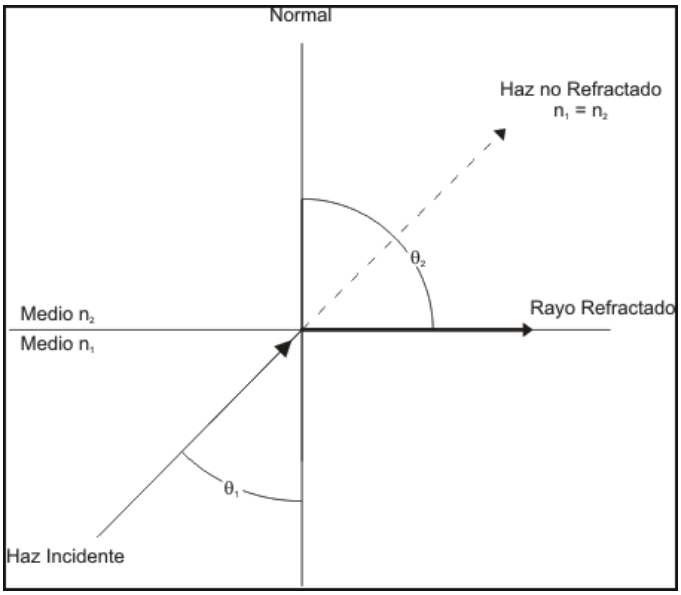

Figura 7. Ángulo crítico [22]

Por último, en la figura 8 se puede estudiar la morfología de los cables de fibra óptica:

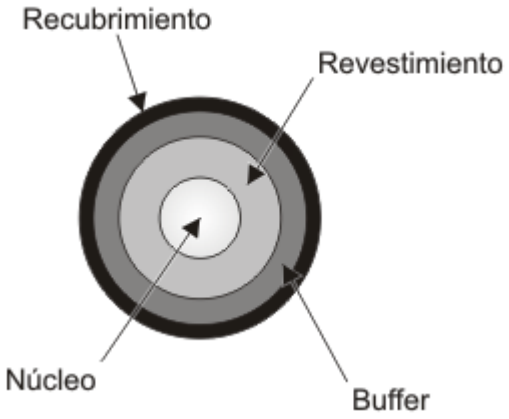

Figura 8. Morfología de la fibra óptica [23]

Tras estudiar algunos fundamentos científicos de la óptica y la estructura de un cable, es más sencillo entender cómo se transmite la luz a través de él.

El haz de luz inyectado en el núcleo de la fibra debe estar dentro del cono de aceptación. Este es el ángulo cuyo vértice se encuentra en el núcleo de la fibra y cuyo valor es igual al del ángulo crítico [24]. Como se puede observar en la figura 9, una vez que la luz se encuentra dentro del núcleo, incide en las paredes del mismo con un ángulo mayor al ángulo crítico, por lo que el haz se refleja y se propaga a lo largo del cable.

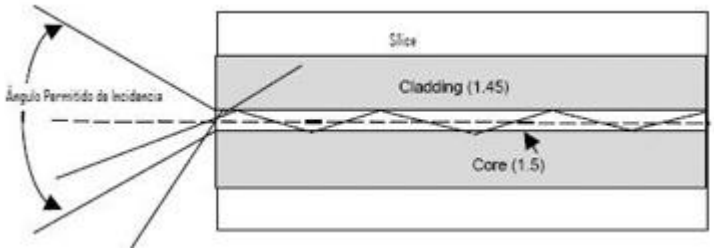

Figura 9. Propagación de la luz en el interior de la fibra [25]

Según el tipo de fibra utilizada se puede diferenciar entre *Single Mode Fiber* o Fibra Monomodo (SMF) y *Multimode Fiber* o Fibra Multimodo (MMF). La primera se caracteriza por poder transportar un solo haz de luz. Se reduce el diámetro del núcleo de tal manera que las reflexiones son prácticamente inexistentes y la luz viaja, virtualmente, en línea recta. Como casi no hay reflexiones, la dispersión es menor y la señal es más pura. La consecuencia de esto es que se puede establecer una conexión de larga distancia y con mayor tasa de tráfico que en MMF. A través del segundo tipo de fibra pueden enviarse varios haces de luz simultáneamente gracias a un mayor tamaño del núcleo de la fibra. La distancia que cubre este tipo de fibra es menor que la ofrecida por SMF.

Las causas de la atenuación en la fibra se pueden agrupar en:

- **Intrínsecas:** atenuaciones por causas propias al material:
    - Pérdidas por absorción: este tipo de pérdidas se deben a las impurezas del material del que está hecho el cable. En este caso se produce una absorción por parte del material de la luz, transformándose en calor.
    - Pérdidas por dispersión: la más común es la llamada dispersión de Rayleigh, la cual está causada por las irregularidades microscópicas de la fibra, que provocan que los haces de luz se difracten. Como consecuencia de esto la luz se dispersa y aunque una parte continúa propagándose, otra parte se pierde. Además de este tipo de dispersión, puede darse también dispersión cromática. Esta se produce cuando la luz emitida por la fibra tiene una combinación de longitudes de onda diferentes, ya que cada una de estas viaja a distinta velocidad y por tanto no llegan al extremo final al mismo tiempo. Este tipo de dispersión puede evitarse utilizando una fuente monocromática.
- **Extrínsecas:** atenuaciones por causas ajenas al material. Destacan las atenuaciones por la curvatura de la fibra, la cual puede provocar que la luz no sufra reflexión total en el núcleo, lo que origina atenuación de la señal. Este tipo de pérdidas puede dividirse en:
    - Pérdidas por microcurvaturas: durante el proceso de fabricación de la fibra pueden cometerse errores. Los defectos que causan las pérdidas por microcurvaturas son aquellos en los que durante la fabricación se producen irregularidades entre el núcleo y el revestimiento o fluctuaciones en el diámetro.
    - Pérdidas por macrocurvaturas: este tipo de pérdidas se producen cuando el radio de curvatura de la fibra es reducido, lo que provoca que los haces de luz incidan con un ángulo mayor al crítico sobre el núcleo y parte de los mismos se refracte.

Todo esto hace que las líneas con acceso por fibra sean mucho más rápidas que aquellas tecnologías que transportan señales eléctricas. Además de ser más rápido, también soportan mucha más tasa de transferencia permitiendo enviar grandes cantidades de datos en muy poco tiempo.

Esta tecnología va ganando cada vez más terreno a las instalaciones ADSL. De hecho, la intención del sector de las telecomunicaciones es llevar a cabo un despliegue masivo en los próximos años que permita llevar servicios de muy alta velocidad a hogares, oficinas y locales. Este hecho puede comprobarse en la estimación de mercado de los cuatro principales operadores de fibra en España [26].

- **Movistar**: la operadora ha doblado el número de clientes en un solo año.
    - Inmuebles pasados por fibra a en julio de 2014: 7,4 millones.
    - Previsión para finales del año 2014: 10 millones
    - Los planes para el año 2015 eran pasar por fibra 5,5 millones de inmuebles más, pero tras la decisión del Comisión Nacional de los Mercados y la Competencia (CNMC) de obligar a la operadora a abrir su red de fibra, este crecimiento se vio

frenado y las estimaciones apuntan a que el despliegue se hará sobre unos 3,6 millones de inmuebles.

- **Vodafone-Ono**: tras la fusión con Ono y la inyección económica para invertir en fibra que eso ha supuesto, se espera un crecimiento significativo de la red de fibra de esta operadora.
  - Estimación de inmuebles pasados por fibra a finales del año 2014: 8 millones.
  - Previsión para el año 2015: 11 millones.
  - Han alcanzado un acuerdo con Orange para desplegar conjuntamente 3 millones de líneas.
- **Jazztel:** a la espera de ser comprada por Orange, esta operadora ofrece un crecimiento más moderado.
  - Estimación de inmuebles pasados por fibra a finales del año 2014: 3 millones.
  - Previsión para el año 2015: 5 millones de inmuebles.
- **Orange**: es la operadora que peores números presenta en cuanto al despliegue de fibra.
  - En la actualidad sólo pasa por fibra un total de 0,8 millones de inmuebles.
  - La previsión para 2015 es de desplegar red hasta un total de 3 millones de inmuebles. Recordemos que este despliegue es el acordado con Vodafone.
  - El objetivo de comprar Jazztel les permitiría acercarse a Movistar y Vodafone-Ono.

Todas estas previsiones pueden verse frenadas por la propuesta del CNMC de que Movistar abra su red de fibra, lo que supondría un frenazo en las inversiones destinadas al despliegue de la misma [27].

Se pueden distinguir diferentes modalidades de fibra óptica según la terminación de la red ya que la fibra puede llegar hasta la casa del abonado o quedarse a una distancia relativamente pequeña y que desde ahí se utilice cobre hasta la casa del abonado. Según este criterio podemos clasificar las redes de fibra óptica en [28]:

- **FTTH**: este tipo de red se caracteriza por utilizar fibra desde la central hasta el domicilio o negocio del abonado, eliminando la infraestructura de cobre utilizada para otro tipo de accesos. La idea se muestra en la figura 10.

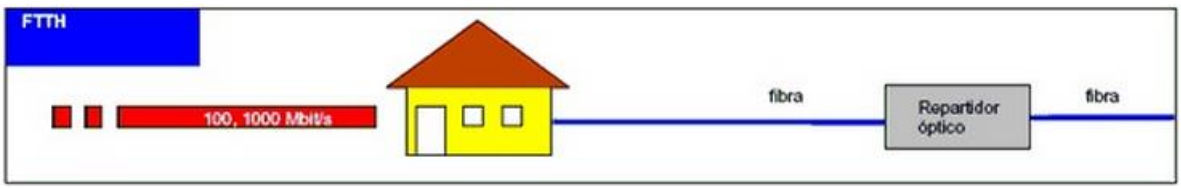

Figura 10. FTTH: fibra desde la central hasta el domicilio o negocio [29]

- ***Fiber To The Building* o Fibra Hasta el Edificio (FTTB)**: como se puede ver en la figura 11, esta red se caracteriza por utilizar fibra desde la central hasta un punto de distribución situado en el interior del edificio al que se quiere dar servicio o en las inmediaciones del mismo. Para conectar el nodo de fibra con el abonado final se utiliza la tecnología VDSL, la cual ha sido explicada en el punto 2.2.

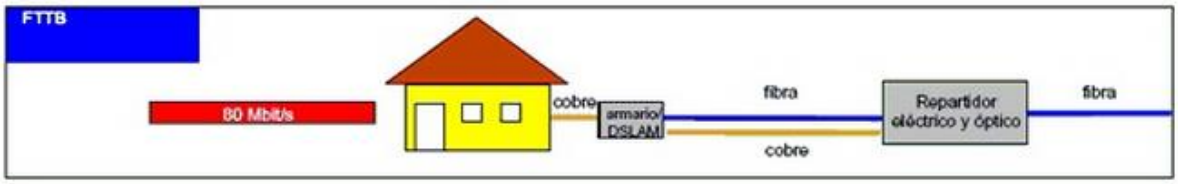

Figura 11. FTTB: fibra desde la central hasta un nodo de distribución situado en el edificio o en las inmediaciones [30]

- ***Fiber To The Node* o Fibra Hasta el Nodo (FTTN)**: esta red se caracteriza por utilizar fibra desde la central hasta un nodo situado en las inmediaciones de un conjunto de edificios a los que dar servicio. Aunque en esencia es el mismo concepto que el mencionado en FTTB, en este caso, mostrado en la figura 12, el nodo distribuidor se sitúa más lejos de los abonados.

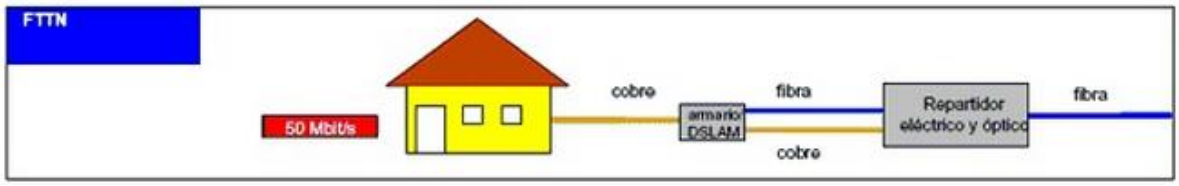

Figura 12. FTTN: fibra desde la central hasta un nodo situado en las inmediaciones de un conjunto de edificios [31]

Para ofrecer cualquiera de las modalidades vistas se puede recurrir a diferentes tipos de tecnologías como son las *Passive Optical Network* o Redes Ópticas Pasivas (PON), las cuales no tienen elementos activos entre las instalaciones del operador y las del abonado, o bien las *Active Optical Networks* o Redes Ópticas Activas (AON), las cuales sí tienen los mencionados elementos activos. Las más utilizadas para dar servicios de fibra óptica son las PON, concretamente las *Gigabit-capable Passive Optical Networks* o Redes Ópticas Pasivas con capacidad de Gigabit (GPON). Esta tecnología viene regulada por el estándar G.984.x (x = 1, 2, 3, etc.) de la *International Telecommunications Union- Telecommunication Standardization* o Sector de Normalización de las Telecomunicaciones (ITU-T), el cual describe aspectos como la gestión de los elementos de red o como compartir un medio común para varios usuarios.

Para terminar este apartado, se van a explicar los elementos de una red GPON [32], los cuales se pueden observar en la figura 13:

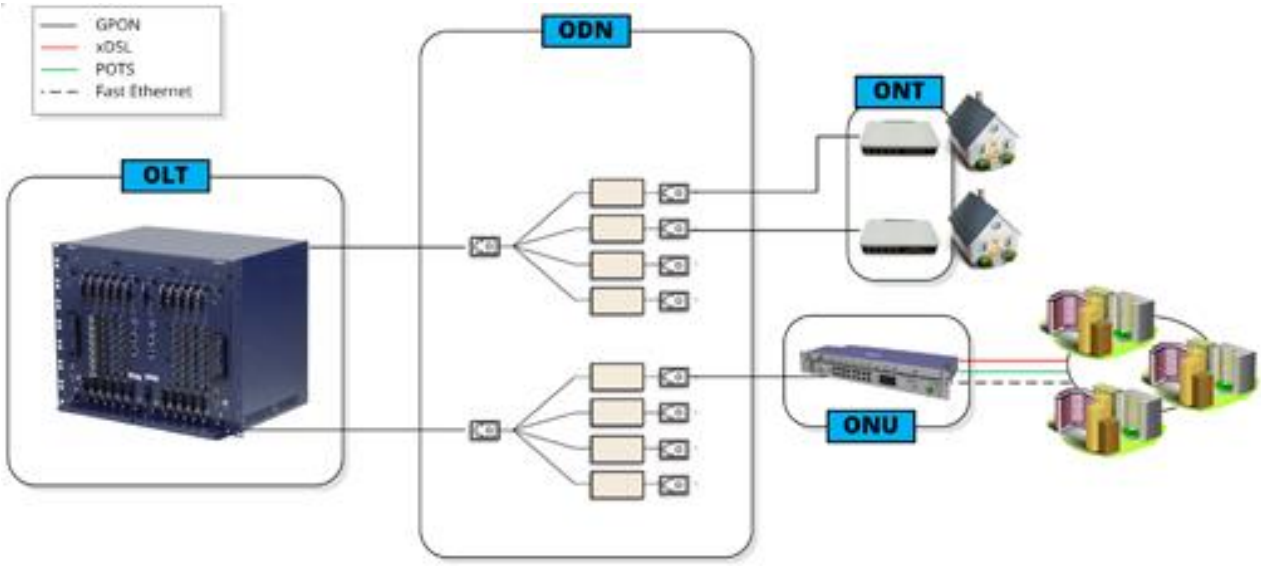

Figura 13. Estructura de una red GPON [33]

- ***Optical Line Terminal* o Terminal de Línea Óptica (OLT)**: es análogo al *Digital Subscriber Line Access Multiplexer* o Multiplexor de Línea de Acceso de Abonado Digital (DSLAM) en una red de cobre. Está situado en el lado del operador y su misión principal es convertir las señales eléctricas en ópticas.
- ***Multi Dwelling Unit* o Unidad de Vivienda Múltiple (MDU)**: característico de redes FTTB/FTTN y situado en el lado del abonado, este dispositivo recibe las señales enviadas por el OLT y las reconvierte, de tal manera que puedan ser envidas por cables de cobre. De esta forma se ofrece servicio a todos los edificios próximos.
- ***Optical Distribution Network* o Red de Distribución Óptica (ODN)**: este bloque hace referencia a los distintos elementos ópticos repartidos por la red, como pueden ser las propias fibras y los *splitters* necesarios para diversificar la red.
- ***Optical Network Terminal* o Terminal de Red Óptica (ONT)**: análogo al módem en un acceso xDSL, este dispositivo es capaz de convertir las señales ópticas recibidas en señales eléctricas (Fast Ethernet o Gigabit Ethernet).
- ***Optical Network Unit* o Unidad de Red Óptica (ONU)**: este dispositivo de distribución que ofrece servicio a más de un usuario a través de tecnologías como VDSL2 o ADSL2+.

2.5. Redes Inalámbricas

A diferencia de otras tecnologías que utilizan cables, las redes inalámbricas ofrecen la interconexión de varios dispositivos a través de ondas de radio o por señales luminosas infrarrojas. Esto supone una gran ventaja puesto que se evitan complejas instalaciones en las que para conectar todos los equipos de la misma eran necesarios muchos metros de cable, con el coste económico que eso supone.

Existen varias opciones de clasificación de las redes inalámbricas. En este documento se ha optado por clasificarlas en función del radio de alcance las mismas. La figura 14 ilustra los cuatro grandes tipos:

Type	Range	Applications	Standards
Personal area network (PAN)	Within reach of a person	Cable replacement for peripherals	Bluetooth, ZigBee, NFC
Local area network (LAN)	Within a building or campus	Wireless extension of wired network	IEEE 802.11 (WiFi)
Metropolitan area network (MAN)	Within a city	Wireless inter-network connectivity	IEEE 802.15 (WiMAX)
Wide area network (WAN)	Worldwide	Wireless network access	Cellular (UMTS, LTE, etc.)

Figura 14. Tipos de redes inalámbricas [34]

- ***Personal Area Network* o Red de Área Personal (PAN)**: red que permite la interconexión de dispositivos en un radio de pocos metros respecto al usuario. Las tecnologías más destacadas que permiten formar redes personales son:
    - *Bluetooth*: estándar inalámbrico basado en la especificación *Institute of Electrical and Electronics Engineers* o Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) 802.15.1 que permite la comunicación entre dispositivos como teléfonos móviles, auriculares o teclados sin necesidad de realizar complicadas configuraciones. Su radio de acción es variable, siendo 10 metros la distancia media en la que la conexión es satisfactoria. En cuanto a su velocidad de transferencia se puede decir que el Bluetooth puede alcanzar los 24 Mbps.
    - *ZigBee*: protocolo de comunicaciones inalámbricas basado en la especificación IEE 802.15.4 que se aplica en entornos en los que se necesitan comunicaciones seguras con bajas tasas de envío de datos y que permitan el ahorro de la batería de los dispositivos implicados. La velocidad de transmisión de esta tecnología está comprendida entre los 25 - 250 Kbps mientras que su radio de acción se sitúa en un rango de 10 a 75 metros. Este es el sistema utilizado en redes domóticas.
    - *Near Field Communication* o Comunicación de Campo Cercano (NFC): sistema de comunicación inalámbrico que permite el intercambio de información entre varios dispositivos de forma instantánea. Su radio de acción es muy corto,

aproximadamente unos 20 cm, y su tasa de transferencia llega hasta los 424 Kbps.

- ***Local Area Network* o Red de Área Local (LAN)**: red que permite la interconexión de dispositivos sin la necesidad de cablearlos mediante la transferencia de ondas de radio, permitiendo a los usuarios mayor movilidad y flexibilidad. Es un sistema alternativo o complementario a las redes locales cableadas. Este tipo de redes tienen un radio de cobertura medio de 10 a 100 metros y su mayor exponente es la tecnología WiFi.
    - *Wirelesss Fidelity* o Fidelidad Inalámbrica (WiFi)*:* basado en el estándar 802.11, esta tecnología se caracteriza por ofrecer la posibilidad de crear una red local con acceso a Internet, interconectando los dispositivos que la conforman entre sí. Desde su aparición en 1997 este estándar se ha desarrollado mejorando sus prestaciones. En la figura 15 se recoge un resumen de las capacidades de los distintos estándares:

Standard	Frequency	Bandwidth	Modulation	Max Data Rate
802.11	2.4 Ghz	20 MHz	DSSS, FHSS	2Mbps
802.11a	5 Ghz	20 MHz	DSSS	54 Mbps
802.11b	2.4 Ghz	20 MHz	OFDM	11 Mbps
802.11g	2.4 Ghz	20 MHz	OFDM	54 Mbps
802.11n	2.4 and 5 Ghz	20 MHz, 40 MHz	OFDM	600 Mbps
802.11ac	2.4 and 5 Ghz	20, 40, 80, 80+80, 160	OFDM	6.93 Gbps

Figura 15. Características del estándar 802.11 (WiFi) [35]

El estándar a su vez define dos modos de funcionamiento de las redes inalámbricas WiFi:

Ad-hoc: este modo se caracteriza porque los dispositivos que conforman la red inalámbrica se comunican entre sí sin necesidad de utilizar un *Access Point* o Punto de Acceso (AP), cuya función describiremos a continuación. La red formada por estos equipos, los cuales han de venir equipados con un adaptador inalámbrico que posibilita la comunicación, es punto a punto puesto que cada uno de ellos actúa como cliente y puede comunicarse con cualquiera de los demás dispositivos de la misma.

El radio de cobertura de esta red es menor que el que ofrece el modo infraestructura ya que las señales las envían los propios dispositivos y estos suelen tener menos potencia de radiación que los AP. Por este motivo, es evidente que un equipo sólo podrá comunicarse con otros que estén en su radio de acción.

Como consecuencia de esta peculiaridad, si la comunicación quiere establecerse entre dos dispositivos que se encuentren en la misma red pero que no tengan comunicación directa, la información se irá escalando por otros equipos hasta llegar al destino.

Además, para poder establecer comunicación es necesario que cada cliente configure su adaptador inalámbrico para que el *Basic Service Set Identifier* o Identificador de Conjuntos de Servicios Básicos (BSSID) de la red y el canal de emisión sean similares. A medida que crece el número de dispositivos conectados a la red disminuye su calidad. Esto se debe a las interferencias creadas entre los clientes, por lo tanto, se recomienda su uso en pequeñas redes de un número reducido de equipos. Para conectar estas redes a una LAN o a Internet es necesario un *gateway*.

Como alternativa a este modo, que hoy en día está en desuso, ha surgido WiFi-Direct. Este nuevo protocolo permite conectar dos equipos inalámbricamente, sin la necesidad de utilizar un AP, e intercambiar archivos a altas velocidades. Se crea una red punto a punto entre los dos dispositivos.

Infraestructura: a diferencia del modo *ad-hoc*, en el que no había un elemento central en la comunicación, este modo se caracteriza porque cada dispositivo que se conecta a la red inalámbrica lo hace a través del punto de acceso o AP. El AP es el dispositivo encargado de la transmisión y recepción de las señales radio intercambiadas en la red, por lo que es el nexo entre todos los elementos conectados a la misma, es decir, centraliza y gestiona todas las comunicaciones inalámbricas [36]. La conexión entre el AP y el encaminador puede verse en la figura 17.

Las principales redes en modo infraestructura ofrecen un radio de cobertura mayor que las del modo *ad-hoc*, así como mayor escalabilidad, estabilidad y seguridad. Sin embargo, el despliegue de estas redes es más caro debido a la necesidad de elementos como el AP. Una de las similitudes entre los dos modos es que tanto el AP como los clientes deben tener el mismo *Service Set Identifier* o Identificador de Conjuntos de Servicios (SSID) para poder comunicarse entre sí. Por el contrario, el modo infraestructura se diferencia del modo *ad-hoc* en que se puede establecer conexión entre varios AP, aumentando el alcance de la red gracias al *Distribution System* o Sistema de Distribución (DS). En estos casos existirán dos identificadores. Por un lado tendremos el BSSID, que identificará cada una de las redes gestionadas por un AP, lo que se denomina celda. Por el otro, se encuentra el *Extended Service Set Identifier* o Identificador de Conjunto de Servicios Extendidos (ESSID), que identifica la red formada por varias celdas. La itinerancia, característica del DS, permite a un usuario que se mueve dentro del *Extended Service Set* o Conjunto de Servicios Extendidos (ESS) cambiar de una *Basic Service Set* o Conjunto de Servicios Básicos (BSS) a otra sin sufrir cortes en la conexión [37]. Estos conceptos pueden verse gráficamente en la figura 16.

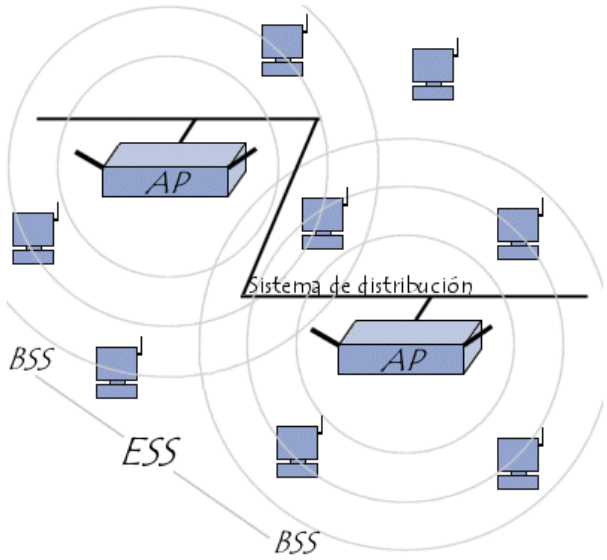

Figura 16. ESS formada por la conexión entre dos o más celdas (BSS) [38]

Por último, si en el modo *ad-hoc* necesitábamos un *gateway* para dotar a la red de conexión a Internet, en el modo infraestructura necesitamos conectar el AP a la LAN cableada que tenga acceso a Internet. En la figura 17 se ilustra la conexión del AP al encaminador que ofrece el acceso a Internet.

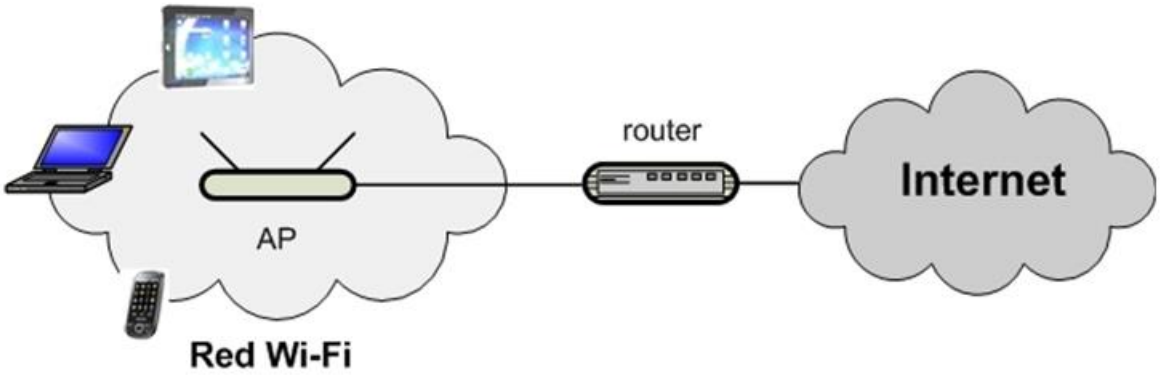

Figura 17. Conexión del AP con el encaminador para ofrecer acceso a Internet [39]

En la actualidad, cada vez son más los encaminadores que llevan incorporada la funcionalidad de los AP, por lo que la conexión mostrada en la imagen se reduce a dos redes, la inalámbrica e Internet, conectadas mediante el encaminador WiFi.

- ***Metropolitan Area Network* o Redes de Área Metropolitana (MAN)**: el concepto de este tipo de redes es el mismo que el de las LAN estudiadas en el punto anterior, es decir, es una red que permite la interconexión de diferentes equipos de manera inalámbrica. La diferencia radica en el alcance de la red. Si en las redes LAN hablábamos de coberturas máximas de 100 metros en las redes MAN se cubren áreas geográficas mucho más grandes, como campus e incluso ciudades.
  Si bien las redes MAN tienen mayor radio de acción que las LAN, no llegan a las distancias cubiertas por las redes *Wide Area Network* o Red de Área Amplia (WAN) que se estudiarán a continuación, por lo que las MAN se consideran redes intermedias.

La principal función de este tipo de redes es la posibilidad de compartir datos entre redes cercanas del mismo lugar geográfico con unas tasas de transferencia elevadas. Por tanto, se podría decir que una red MAN es la interconexión de varias redes LAN. Su radio de cobertura oscila según la red llegando a alcanzar los 45 - 50 kilómetros. La tecnología más destacada de este tipo de redes inalámbricas es WiMAX [40].

- WiMAX: esta tecnología permite el acceso a Internet en zonas rurales donde no llegan otro tipo de tecnologías como el xDSL o la fibra óptica. El funcionamiento de esta tecnología es muy similar al de la tecnología WiFi con la diferencia de que WiMAX ofrece velocidades de transferencia más altas, cubre distancias mayores y permite la conexión de más clientes. Por otro lado, si para conectarte a una red WiFi en modo infraestructura es necesario que los clientes se conecten a la red a través del AP, en WiMAX los clientes se conectan a la red a través de la estación base que ofrece el servicio.

  - **Estación base**: este elemento de red se encarga de emitir las señales y da cobertura a un área aproximada de 8000 kilómetros cuadrados.
  - **Receptor**: estos dispositivos, que como su propio nombre indica reciben la señal, suelen ser cajas situadas en el exterior de los edificios o tarjetas *Personal Computer Memory Card International Association* o Asociación Internacional de Tarjetas de Memoria para Ordenadores (PCMCIA). Se conectan a la estación base de la misma manera que se conectan los clientes WiFi al AP.

  Las estaciones base pueden establecer conexiones entre sí, las cuales se conocen como *backhaul,* con el fin de proporcionar al usuario itinerancia, es decir, aunque el usuario se mueva y durante ese movimiento se pase de una zona de cobertura ofrecida por una estación base a otra, la conexión no se pierde.

  Esta tecnología ofrece dos tipos de servicio, los cuales se pueden observar en la figura 18:

  - ***Line Of Sight* o Línea de Visión (LOS)**: esta clase de servicio se caracteriza por establecer una comunicación directa entre la estación base y una antena parabólica instalada en el cliente. Esta conexión se destaca por ser estable, pudiendo enviar una gran cantidad de datos con pocos errores. Utiliza altas frecuencias, las cuales pueden alcanzar hasta los 65 GHz.
  - ***Non Line Of Sight* o Sin Línea de Visión (NLOS)**: en este caso la comunicación no es directa. Se utiliza un rango de frecuencias bajas, entre 2 y 10 GHz, porque la longitud de onda es más grande y las señales emitidas de este modo pueden sobreponerse mejor a los obstáculos que se encuentren por el camino, rebotando más fácilmente.

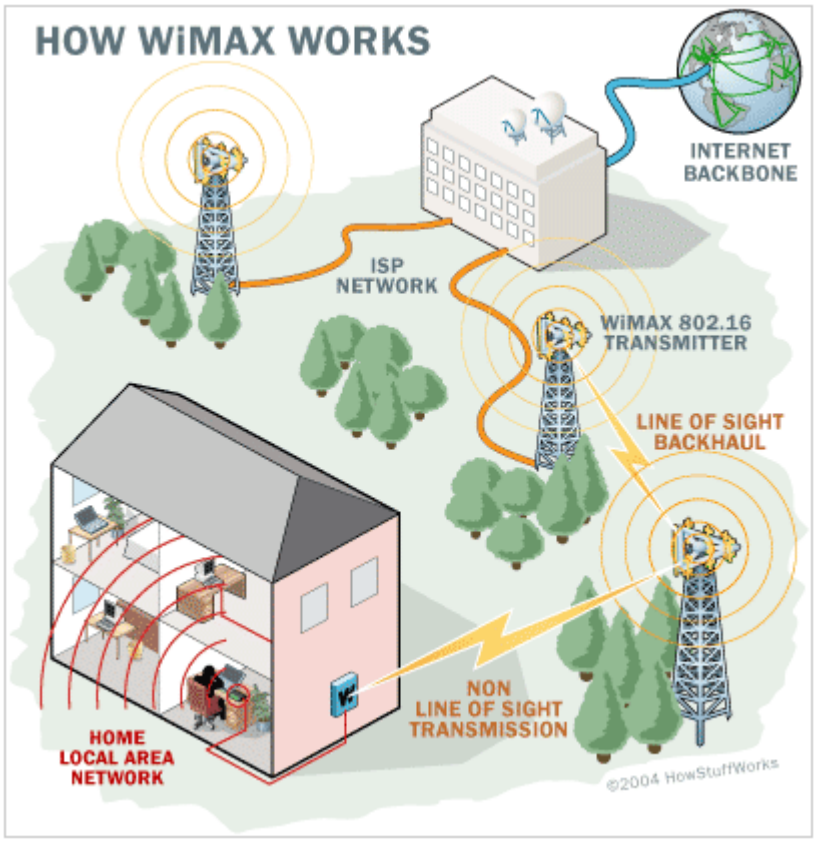

Figura 18. Funcionamiento de la red WiMAX [41]

- **WAN**: las redes de área extensa se caracterizan por ofrecer servicios inalámbricos a grandes zonas geográficas como pueden ser ciudades, países, e incluso redes que engloban todo el planeta, como es el caso de Internet. Durante los últimos años las redes móviles de banda ancha han experimentado un gran crecimiento consiguiendo ofrecer al usuario velocidades de transferencia equiparables a las de las líneas cableadas.

Este tipo de redes ofrecen conexiones a Internet a través de dispositivos móviles que se encuentren dentro de la zona cobertura ofrecida por las mismas. A continuación se van a enumerar, de manera resumida, los principales estándares de redes móviles [42]:

  - ***Global System for Mobile communications* o Sistema Global para las Comunicaciones Móviles (GSM)**: denominadas redes de segunda generación, utilizan una banda de frecuencias de 900MHz a 1800 MHz y ofrecen servicios tanto de voz como de datos. La tecnología de acceso a la interfaz radio utilizada es *Time Division Multiple Access* (TDMA), que permite dividir un canal de comunicaciones en ranuras de tiempo que son asignadas a distintos usuarios, lo que permite que varios usuarios utilicen el mismo canal sin sufrir interferencias. Sin embargo, se podría decir que no es un estándar recomendado para la transmisión de datos puesto que ofrece velocidades bajas (un máximo de 9,6 Kbps), utiliza técnicas de conmutación de circuitos y además se factura por tiempo de conexión.

Se considera que existen dos extensiones de GSM: *General Packet Radio Service* o Servicio General de Paquetes vía Radio (GPRS) y *Enhanced Data rates for GSM of Evolution* o Tasas de Datos Mejoradas para la Evolución del GSM (EDGE). GPRS se caracteriza por ser una tecnología basada en la conmutación de paquetes en la que se factura por la cantidad de datos intercambiados. Ofrece velocidades de bajada de hasta 170 Kbps teóricos, aunque en la práctica apenas se llega a 50 Kbps. EDGE se caracteriza por ser una tecnología que utiliza esquemas de modulación y codificación alternativos, como son *Gaussian Minimum Shift Keying* o Modulación por Desplazamiento Mínimo Gaussiano (GMSK) y *8 Phase Shift Keying* o Modulación por Desplazamiento de Fase (8PSK). Ofrece velocidades de bajada desde 115 Kbps hasta 384 Kbps.

- ***Universal Mobile Telecommunications System* (UMTS)**: denominadas redes de tercera generación, se caracterizan principalmente por utilizar *Wideband Code Division Multiple Access* o Acceso Múltiple por División de Código de Banda Ancha (WCDMA) como tecnología de acceso a la interfaz radio. Mediante esta tecnología se pueden transmitir varias comunicaciones sobre una misma señal portadora utilizando códigos diferentes. Las velocidades que se pueden alcanzar en estas redes pueden ser de hasta 2 Mbps. La optimización de esta tecnología da como resultado *High Speed Downlink Packet Access* o Acceso Descendente de Paquetes a Alta Velocidad (HSDPA) y *High-Speed Uplink Packet Access* o Acceso Ascendente de Paquetes a Alta Velocidad (HSUPA). Con la primera se optimiza la capacidad de transferencia de bajada, consiguiendo tasas de hasta 14 Mbps, aunque la tasa promedio es de 1 Mbps. Con la segunda se optimiza la capacidad de transferencia de subida, llegando hasta los 5,76 Mbps.
- ***Long Term Evolution* o Evolución a Largo Plazo (LTE)**: denominadas redes de cuarta generación, se caracteriza por utilizar las tecnologías *Orthogonal Frequency Division Multiple Access* o Acceso Múltiple por División de Frecuencias Ortogonales (OFDMA) para el enlace descendente y *Single Frequency Division Multiple Access* o Acceso Múltiple por División de Frecuencia Portadora Única (SC-FDMA) para el enlace ascendente. Gracias a esta tecnología se puede disfrutar de velocidades de hasta 100 Mbps de bajada y 50 Mbps de subida.

Los elementos a destacar para establecer la conexión con una red móvil son [43]:

- **Tarjeta de datos**: dispositivo que permite establecer la conexión entre el equipo y la red móvil. Los formatos más habituales en los que nos encontramos estos elementos son o integrados en los equipos portátiles o como dispositivos *Universal Serial Bus* o Bus Universal en Serie (USB).
- **Tarjeta *Subscriber Identity Module* o Módulo de Identificación de Abonado (SIM)**: tarjeta en la que se almacena de forma segura la identidad del usuario, necesaria para identificarse en la red, así como información de seguridad de ciertos servicios.

- ***Access Point Name* o Nombre del Punto de Acceso (APN)**: nombre del punto de acceso proporcionado por el operador de la red móvil que identifica el tipo de configuración necesaria para establecer una conexión entre el dispositivo móvil e Internet. Cada operador puede ofrecer distintos APN en función del tipo de servicios ofrecidos.

2.6. Satélite

No siempre se pueden realizar las instalaciones de accesos como ADSL o FTTx en todos los lugares, ya sea por problemas geográficos que impidan el despliegue, o bien por el precio de dicho despliegue. Estos aspectos geográficos y económicos hacen que zonas rurales o de difícil acceso sean susceptibles de no tener ninguna tecnología de acceso a Internet, lo que en el mundo actual es, prácticamente, estar aislado. Estos impedimentos hacen que, en España, alrededor de 500.000 personas carezcan de acceso a Internet de banda ancha. Esta población se caracteriza por vivir en pequeños municipios de difícil acceso, donde "*la cobertura de Internet de banda ancha no supera el 10%*" [44]. Para solucionar este problema se utiliza el acceso a Internet vía satélite. El elemento fundamental en esta red es el satélite de comunicaciones ya que permite recibir peticiones desde el hogar, enviar dichas señales a Internet y, finalmente, devolver la información solicitada. La cobertura ofrecida por el satélite cubre prácticamente el 100% de la geografía española, ofreciendo una velocidad de hasta 22 Mbps.

Los satélites encargados de ofrecer conexión a Internet se comunican a través de la banda Ka. La banda Ka es una banda de frecuencias situadas en un rango comprendido entre los 18.3 GHz y los 31 GHz. La comunicación entre un dispositivo y el satélite se establece por dos canales diferentes, el de subida, en el que las señales van desde el hogar u oficina hasta el satélite y que funciona con frecuencias entre los 27.5 GHz y los 31 GHz, y el de bajada, en el que las señales viajan desde el satélite hasta el hogar u oficina y cuyo rango de frecuencias se encuentra entre los 18.3 GHz y los 20.2 GHz. Las frecuencias de estos canales son del orden de los GHz puesto que, basándose en los teoremas de Nyquist y Shannon que a continuación se exponen brevemente, a mayores frecuencias, mayores velocidades de transferencia se pueden soportar.

Partiendo de un canal ideal en el que no hay ruido que distorsione la señal, el teorema de Nyquist manifiesta que para poder reconstruir una señal a partir de sus muestras, la frecuencia de muestreo debe ser, al menos, 2 veces mayor que la máxima frecuencia de la señal de entrada. Dicho de otro modo, dado un ancho de banda B, la mayor tasa de transferencia que se puede transportar es de 2B. En base a esta idea se llega a la conclusión de que, en este caso, la tasa de transferencia está limitada exclusivamente por el ancho de banda de la señal [45].

En función de estas aseveraciones se concluye que la velocidad máxima de transmisión para un canal sin ruido es la mostrada en la figura 19:

$$C = 2 * B * \log M,$$

Figura 19. Velocidad máxima de transmisión para un canal sin ruido [46]

Donde C es la velocidad medida en bits por segundo (bps), B el ancho de banda de la señal y M el número de niveles de tensión.

Sin embargo, las situaciones ideales nunca se dan por lo que siempre hay que contar con un cierto nivel de ruido. El ruido, que se mide en decibelios (dB), se puede definir como la relación entre la potencia de la señal y la potencia del ruido en un punto determinado de la comunicación y viene representado por la fórmula de la figura 20:

$$SNR_{dB} = 10 \log \frac{Signal - Power}{Noise - Power}$$

Figura 20. Fórmula del ruido en la comunicación [47]

Según el teorema de Shannon, la velocidad máxima de transmisión de un canal con ruido se representa mediante la fórmula de la figura 21:

$$C = B \log (1 + S/N)$$

Figura 21. Velocidad máxima de transmisión para un canal sin ruido [48]

Donde C es la velocidad medida en bps, B el ancho de banda de la señal y S/N la relación señal – ruido.

Por tanto, se puede concluir que la tasa de transferencia depende tanto del ancho de banda de la señal como de la relación señal - ruido.

Existen dos tipos de acceso a Internet vía satélite [49], los cuales se pueden observar en las figuras 22 y 23:

- **Unidireccional**: en este tipo de acceso, mostrado en la figura 22, sólo existe un canal de comunicación de alta velocidad entre el satélite y el usuario. Este canal es descendente desde el punto de vista del usuario. El usuario realiza peticiones a través de la red convencional terrestre, la cual fue explicada en el punto 2.1. Los servidores de Internet ofrecen la respuesta a dichas peticiones y se la envían a la estación terrena encargada de transmitir la respuesta al usuario. Las estaciones terrenas son estaciones terrestres cuya misión fundamental es establecer comunicación directa tanto de transmisión como de recepción con los satélites de comunicaciones. Finalmente, desde esta estación se transmite la respuesta a la petición que hizo el usuario a través del enlace por satélite. La información captada por la parabólica del usuario se transmite entonces al módem satelital unidireccional o *Digital Video Broadcasting- Internet Protocol* (DVB-IP), el cual sólo tiene un canal de entrada y se encarga de transformar la señal [50].

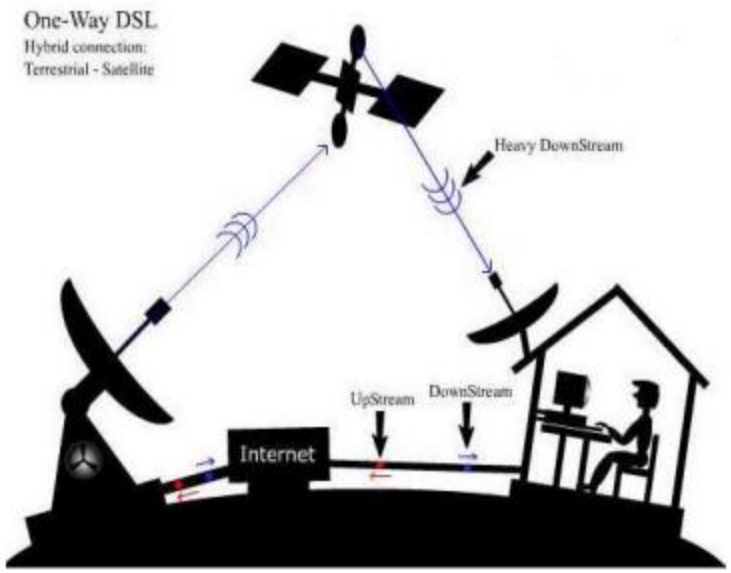

Figura 22. Acceso satelital unidireccional [51]

- **Bidireccional**: en el caso mostrado en la figura 23, se establecen dos canales de comunicación entre el usuario y el satélite, uno ascendente y otro descendente, ambos de alta velocidad. En este caso las peticiones a la red se realizarían a través del canal ascendente y, una vez obtenida la respuesta de los servidores de Internet, esta se devuelve por el canal descendente. En este caso, tanto la señal enviada para realizar la petición como la señal recibida con la información de respuesta pasan a través de un módem bidireccional, el cual se denomina *Digital Video Broadcasting- Return Channel Satellite* (DVB-RCS), capaz de recibir y enviar datos.

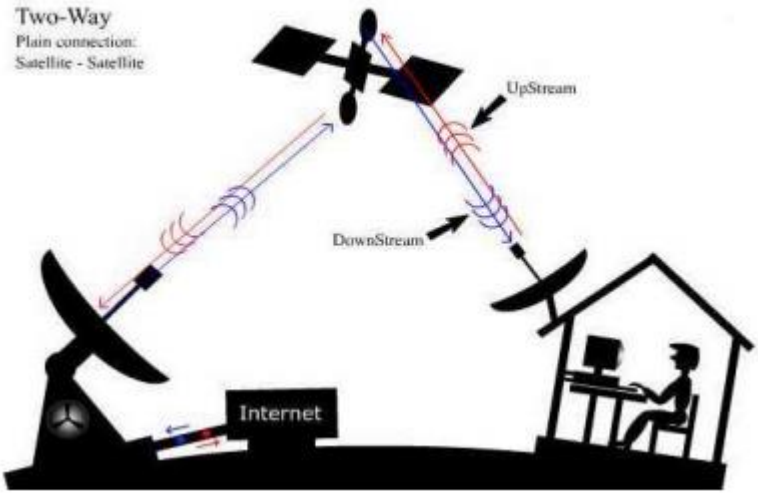

Figura 23. Acceso satelital bidireccional [52]

2.7. PLC

La tecnología PLC se caracteriza por utilizar las líneas eléctricas de hogares y empresas para ofrecernos conectividad a redes locales transmitiendo de forma simultánea datos y la propia corriente eléctrica. Tal y como ocurre en otros métodos de acceso, es necesario separar las señales de datos y la corriente eléctrica. Esto se consigue con un filtro instalado en el propio dispositivo que es capaz de diferenciar las frecuencias de las señales de datos (del orden de MHz) y las frecuencias de la corriente alterna (alrededor de los 50 Hz).

Antes de la creación de los estándares definidos por la *HomePlug Powerline Alliance* y la *Universal Powerline Association*, podía darse la situación de que dispositivos situados en la misma instalación que tuvieran chipsets diferentes fueran compatibles o no compatibles. Los primeros podían convivir pero no comunicarse entre sí, mientras que los segundos ni siquiera podían convivir, provocando un mal funcionamiento. Gracias a los estándares de HomePlug este problema se resolvió y hoy en día los adaptadores son interoperables entre sí, es decir, pueden convivir y comunicarse entre sí. Los productos basados en las nuevas especificaciones HomePlugAV2 ofrecen velocidades teóricas de hasta 200 Mbps.

La principal ventaja que nos ofrecen los accesos mediante adaptadores PLC es que no necesitamos complicadas instalaciones de cables para conectar diferentes dispositivos, como pueden ser ordenadores, consolas de videojuegos o televisores, a la red. Es, por tanto, una alternativa al WIFI estudiado en el punto 2.5.

Por el contrario, las sobrecargas y alteraciones que se produzcan en la red pueden dar como resultado interferencias que afectan al servicio ofrecido. Otro aspecto que puede afectar al rendimiento de los PLC es el estado de la red eléctrica del hogar u oficina.

Al ser una red inalámbrica podría darse la situación de algún extraño accediese a nuestra red, con los peligros que eso supone. Sin embargo, a no ser que algún vecino se conecte físicamente a la red eléctrica de tu hogar u oficina, es improbable que la señal llegue a otros inmuebles puesto que los contadores y cajas de distribución actúan como barreras para la misma. Además, existe encriptación de datos mediante cifrados *Advanced Encryption Standard* o Estándar de Encriptación Avanzada (AES), *Data Encryption Standard* o Estándar de Encriptación de Datos (DES) o Triple DES que mejoran la seguridad de la red [53].

Tal y como podemos observar en la figura 24, lo primero que hay que hacer es conectar un adaptador PLC al encaminador a través de un cable Ethernet. Con esto lo que conseguimos es distribuir la señal de datos por la red eléctrica del hogar o de la oficina. Una vez que la red transmite simultáneamente la corriente eléctrica y los datos, colocaremos tantos PLC en los enchufes del inmueble como puntos de conexión a la red queramos tener. A su vez, estos PLC hay que conectarlos con el dispositivo (televisor, ordenador, consola) a través de otro cable Ethernet.

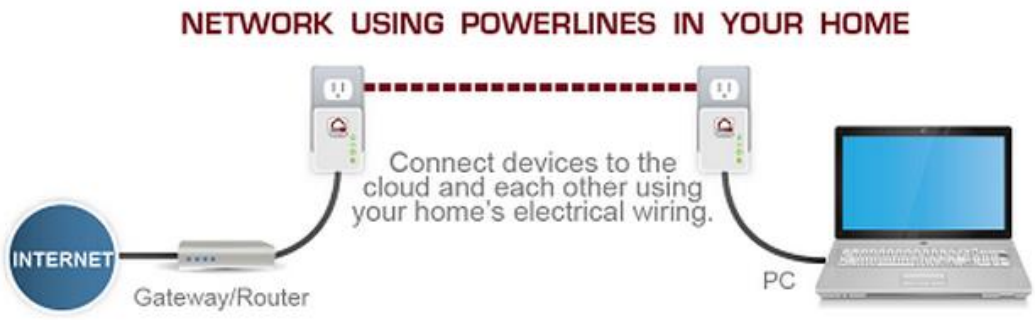

Figura 24. Funcionamiento del PLC [54]

Como conclusiones finales podemos observar que la tecnología PLC:

- No ofrece acceso directo a Internet sino que, a partir de un acceso como puede ADSL, fibra óptica o Internet móvil, se puede distribuir la señal de datos por el hogar u oficina sin la necesidad de utilizar cables.
- Es una alternativa al WIFI. Son dos tecnologías similares pero no excluyentes, ya que pueden ser combinadas si el PLC se conecta a un punto de acceso WIFI allí donde la señal emitida por el encaminador principal llega levemente.
- Puede sufrir interferencias que afecten a la calidad del servicio ofrecido, afectando gravemente a las velocidades de transferencia.
- Es una red segura ya que además de que la señal no sale de la instalación eléctrica del hogar u oficina, tiene métodos de encriptación de los datos.

CAPÍTULO 3

MODELOS DE COMUNICACIONES Y DISPOSITIVOS

3. Modelos de comunicaciones y dispositivos

Tras hacer un estudio detallado de los diferentes accesos que existen hoy en día para poder disfrutar de Internet, durante este apartado se va a explicar qué es un encaminador o *router* y qué diferencias hay entre este dispositivo y otros con funciones similares con los que se puede confundir. El objetivo es que el lector tenga una visión clara de algunos aspectos teóricos del equipo que se va a configurar con el fichero generado por el programa desarrollado. Los dispositivos que se van a estudiar durante este apartado son:

- ***Router* o encaminador.**
- **Switch o conmutador.**
- **Hub o concentrador.**

Para comprender las funcionalidades de estos dispositivos es necesario conocer cómo se estructuran los protocolos de comunicación que permiten tanto la comunicación entre dispositivos pertenecientes a una misma red como la comunicación entre dos o más redes. Para ello se van a estudiar el modelo de referencia *Open Systems Interconnection* o Interconexión de Sistemas Abiertos (OSI) y el modelo *Transmission Control Protocol-Internet Protocol* o Protocolo de Control de Transmisión IP (TCP-IP).

3.1.OSI y TCP-IP.

La necesidad de poder comunicar unas redes con otras dio lugar a la creación de modelos de referencia que fueran capaces de estandarizar los protocolos de comunicación. Fundamentalmente se puede hablar de dos modelos: el modelo de referencia OSI y el modelo TCP-IP.

3.1.1. Modelo de referencia OSI.

Desarrollado por la *International Organization for Standardization* u Organización Internacional de Estandarización (ISO) en 1978, este modelo está constituido por 7 capas o niveles, cada una de las cuales define las distintas funciones de los protocolos de comunicaciones. Pese a que se ideó como modelo de referencia para las comunicaciones entre redes, con el paso del tiempo no se extendió como se esperaba. Los protocolos de cada una de las capas sólo se comunican con sus homólogos del sistema destino, sin necesidad de establecer más comunicación con las capas superiores o inferiores que acordar cómo intercambiar los datos entre ellas. La principal ventaja de este tipo de arquitectura es que, al aislar las funciones de comunicación en capas, se minimiza el impacto de posibles cambios tecnológicos de manera que las modificaciones realizadas en alguna de las capas no afecta a las demás.

Tal como se puede observar en la figura 25, existen siete capas o niveles de red [55]:

Figura 25. Niveles de red del modelo OSI [56]

- **Nivel de aplicación**: esta es la capa del modelo más cercano al usuario final y ofrece interfaces de comunicación y servicios a los distintos programas que el usuario utiliza para interactuar con la red. Como ejemplo podemos mencionar los servicios que soportan aplicaciones para la transferencia de archivos o sistemas de correo electrónico.
- **Nivel de presentación**: para poder entender este nivel se puede comparar con un traductor. En esta capa se gestiona la manera de reflejar los datos transmitidos entre dos sistemas que utilicen distintos modos de representación de los mismos. Además, en aquellas aplicaciones en las que se necesite, este nivel es capaz de realizar funciones tanto de cifrado como de compresión.
- **Nivel de sesión**: la función de esta capa es establecer, gestionar y finalizar las comunicaciones entre dos equipos. Del mismo modo, sincroniza la comunicación entre las capas de aplicación de los dos sistemas. Además, establece puntos de control o *checkpoints* cuya función es que, en caso de que la conexión se interrumpa por algún motivo, sólo se envían los datos que faltaban para completar la comunicación desde el último *checkpoint* establecido.
- **Nivel de transporte:** primer nivel de los cuatro inferiores que se encargan del transporte de datos. Este nivel se caracteriza por controlar el flujo de datos intercambiados en la comunicación de forma que asegure una entrega sin errores y en secuencia extremo a extremo. Esta capa también se encarga de la segmentación de los paquetes para cumplir con el tamaño máximo requerido por los protocolos de capas inferiores.

- **Nivel de red**: en esta capa se encaminan los paquetes enviados por el emisor desde una red y se entregan en el destino adecuado, el cual puede encontrarse en una red distinta. Para ello se determina la ruta física decidiendo el camino, en función de una serie de factores como pueden ser las condiciones de la red o la prioridad del servicio, que han de seguir los datos a través de la red. Este nivel también se caracteriza por ser aquel en el que realiza una traducción de direcciones lógicas a direcciones físicas. El elemento que realiza estas operaciones es el encaminador o *router*, el cual se estudiará específicamente en el apartado 3.2.
- **Nivel de enlace de datos**: esta capa se encarga de la transmisión de los datos a través del enlace físico. Los datos a transmitir se encapsulan en tramas de red, cuya morfología vendrá definida por la arquitectura de la red específica en la que se realiza la comunicación. Además, este nivel se encarga de asegurar que todas las tramas enviadas por el enlace físico se reciben sin error. Este nivel se subdivide en dos:
    - ***Logical Link Control* o Control de Enlace Lógico (LLC)**: este subnivel es el encargado del control de flujo y control de errores durante la transmisión de datos a través del enlace físico. Además, LLC es capaz de identificar el protocolo de línea así como asignar un número de secuencia a las tramas que, junto a los *Acknowledgement (*ACK), permiten asegurar la correcta entrega de las mismas.
    - ***Media Access Control* o Control de Acceso al Medio (MAC)**: este subnivel establece la forma de comunicación entre los nodos, los cuales pueden estar en la misma red o en redes diferentes. Cada uno de los nodos tiene una dirección MAC que lo identifica de manera unívoca.
- **Nivel físico**: en esta capa, las tramas generadas en el nivel superior, son enviadas como una secuencia de bits. En este nivel se definen las interfaces eléctrica/óptica, mecánica y funcional del medio físico.

Tal y como se mencionaba al principio del apartado, el modelo de referencia OSI es un modelo teórico que sirve de guía estructural para el resto de protocolos de comunicaciones. A continuación se va a describir el modelo TCP-IP ya que incluye los protocolos fundamentales de Internet.

3.1.2. TCP-IP

TCP-IP puede definirse como un conjunto de protocolos que es soportado por casi todos los sistemas operativos como protocolo de red predeterminado. Este modelo permite la comunicación entre dos nodos situados en distintas redes y puede considerarse el estándar a partir del cual surgió Internet [57].

En este caso no tenemos siete niveles de red como ocurría en el modelo de referencia OSI, sino que se agrupan alguna de las capas dando como resultado un modelo de cuatro capas. En la figura 26 se puede observar la agrupación de capas mencionada:

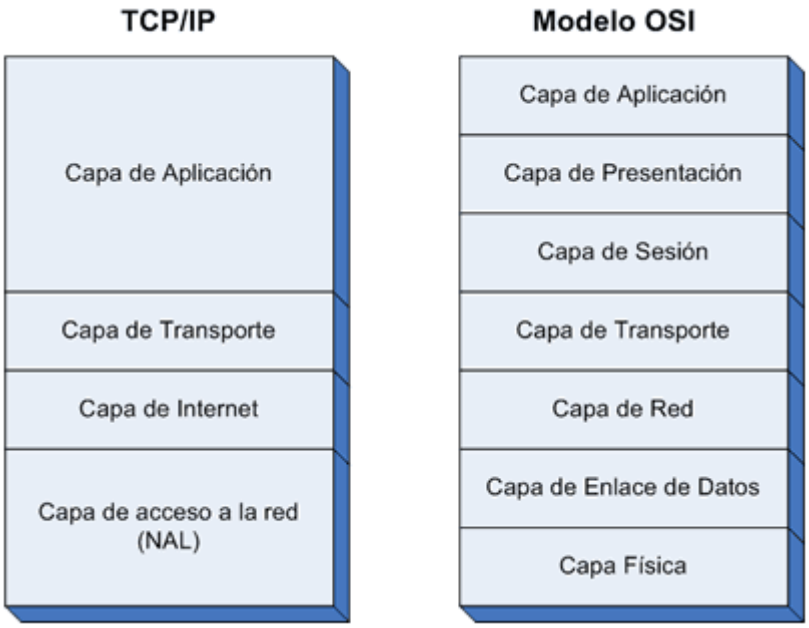

Figura 26. Relación entre capas del modelo OSI y del modelo TCP-IP [58]

- **Nivel de aplicación**: esta capa combina todos los aspectos relacionados con las aplicaciones. En este nivel se definen los protocolos y servicios utilizados en el intercambio de datos con la capa de transporte. De entre todos los protocolos de aplicación soportados en esta capa destacamos los siguientes:
  - **Telnet:** protocolo que permite la conexión con terminales remotos para realizar acciones de configuración y control.
  - ***File Transfer Protocol* o Protocolo de Transferencia de Archivos (FTP)**: protocolo orientado a la transferencia fiable de archivos ya que está basado en el protocolo TCP.
  - ***Simple Mail Transfer Protocol* o Protocolo de Transferencia Simple de Correo Electrónico (SMTP)**: protocolo cuya función es permitir el funcionamiento del servicio de correo electrónico en las redes.
  - ***Hipertxt Transfer Protocol* o Protocolo de Transferencia de Hipertexto (HTTP)**: permite el intercambio de archivos de texto, gráficos, sonidos e imágenes.

- ***Domain Name Service* o Sistema de Nombres de Dominio (DNS)**: convierte los nombres de los dominios y sus nodos en direcciones IP.
- ***Simple Network Management Protocol* o Protocolo Simple de Administración de Red (SNMP)**: protocolo utilizado para recopilar e intercambiar información de gestión de la red.

- **Nivel de Transporte**: este nivel se encarga de llevar a cabo la comunicación entre el origen y el destino. Los dos protocolos utilizados en este nivel son TCP y *User Datagram Protocol* o Protocolo de Datagrama de Usuario (UDP).
  - **TCP**: protocolo fiable orientado a la conexión que se encarga de establecer la conexión, controlar el flujo de datos y corregir errores gracias al uso de los números de secuencia, que identifican los paquetes, y de los acuses de recibo o ACK para que si un paquete no es confirmado se reenvía de nuevo.
  - **UDP**: este protocolo no orientado a la conexión no es fiable puesto que no ofrece ninguna de las características anteriormente mencionadas en el apartado de TCP.
- **Nivel de Internet**: esta capa es la responsable de las funciones de direccionamiento y de selección de la mejor ruta para el envío de los paquetes, la cual se conoce como *routing.* Los principales protocolos de esta capa son:
  - **IP**: protocolo no orientado a la conexión que confía en los servicios de control de errores y control de flujo ofrecidos por el protocolo TCP de la capa superior. Los paquetes enviados por la red mediante este protocolo se denominan datagramas y han de tener un formato como el mostrado en la figura 27:

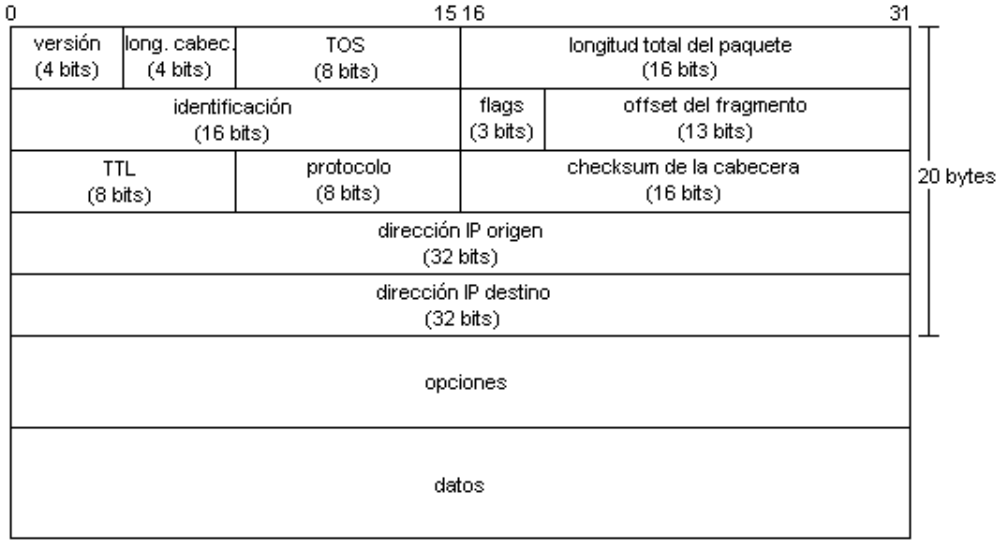

Figura 27. Formato de un datagrama IP [59]

Dentro del protocolo IP es fundamental destacar el papel de las direcciones IP. La dirección IP es un número que identifica a un equipo de manera única dentro de una determinada red de comunicaciones. Dependiendo de la versión, la dirección IP está compuesta por 32 bits en IPv4 o por 128 en la nueva versión IPv6. Tomando como ejemplo la versión IPv4, estos 32 bits se agrupan en bytes u octetos de los que se obtiene su representación decimal. Por tanto, una dirección IP está compuesta por cuatro números, cuyo valor varía entre 0 y 255, separados por puntos. Se puede observar gráficamente en el siguiente ejemplo:

Ejemplo: 32 bits: 11000000 . 10101000 . 0000001 . 0000001

Decimal: 192.168.1.1

Una dirección IP consta de dos partes. Por un lado tenemos la dirección de red y por el otro tenemos los equipos que forman parte de dicha red. Para poder diferenciar las dos partes se utiliza la máscara de subred.

El formato de la máscara de red es similar al de una dirección IP con la particularidad de que, en este caso, todos los bits que hagan referencia a la red estarán a 1 mientras que todos los que pertenezcan al grupo de equipos que pertenecen a dicha red estarán a 0. Se puede observar gráficamente en el siguiente ejemplo:

Ejemplo: 32 bits: 11111111 . 11111111 . 11111111 . 000000

Decimal: 255.255.255.0

Gracias a la combinación de la dirección IP y de las máscaras de subred se pueden determinar con exactitud tanto la red como el equipo, por lo que cuando un datagrama se va a enviar por la red se comprueban la dirección IP origen y la dirección IP destino. Si ambas pertenecen a equipos que se encuentren en la misma red el datagrama se envía directamente. El problema surge cuando los equipos se encuentran en redes diferentes. En este caso es necesaria la presencia de un *gateway* o puerta de enlace, que permitirá que el datagrama pueda enviarse de una red a otra. Este paso o salto de una red a otra es lo que se denomina como *hop*.

Además de poder saltar de red en red para llegar al destino es necesario que se establezca la mejor ruta desde el origen al destino, la cual no siempre tiene que ser la misma. Para ello es necesario un elemento de red denominado encaminador, el cual estudiaremos en más detalle en el apartado 3.2.

- ***Address Resolution Protocol* o Protocolo de Resolución de Direcciones (ARP)**: protocolo utilizado para traducir las direcciones IP de la capa de Internet a direcciones físicas denominadas MAC, las cuales son únicas para cada dispositivo. Las MAC son las direcciones utilizadas en la capa de enlace de red.

- ***Internet Control Message Protocol* o Protocolo de Mensajes de Control de Internet (ICMP)**: protocolo que realiza funciones de control de flujo, diagnóstico de problemas o pruebas de conectividad, es decir, permite notificar los errores producidos durante el procesamiento y envío de los datagramas.
- ***Internet Group Management Protocol* o Protocolo de Administración de Grupos de Internet (IGMP)**: protocolo responsable de la gestión de los grupos de multidifusión IP.

- **Nivel de acceso a la red**: es el nivel más bajo y se encarga de encapsular los datagramas IP en tramas que puedan ser transmitidas por la red. Como TCP-IP se diseñó para ser independiente del método de acceso a la red, podemos encontrar tecnologías para redes LAN, como Ethernet o Token Ring, para redes WAN, como X.25 o Frame Relay, u otras tecnologías como *Asynchronous Transfer Mode* o Modo de Transferencia Asíncrona (ATM).

Tras conocer la teoría básica del modelo de referencia OSI y del modelo TCP-IP se puede entender mejor en qué nivel trabajan los dispositivos que se van a estudiar a continuación con el fin de comprender qué hace el equipo que será configurado con la aplicación desarrollada en este proyecto.

3.2.Encaminador o *router*

El encaminador es un dispositivo que actúa en la capa de red (nivel 3) del modelo OSI y que permite la interconexión entre dos o más redes. El objetivo de estos dispositivos es gestionar los datagramas enviados entre las redes de manera que los paquetes enviados desde un origen lleguen a un destino de la manera más rápida posible, que no tiene porqué ser la más corta [60].

Lo primero que tiene que comprobar el encaminador es si el destinatario del paquete de datos se encuentra en la misma red que el emisor o por el contrario está situado en una red remota. Para ello utiliza la dirección IP y la máscara de subred, conceptos teóricos descritos en el apartado 3.1.2. Si el encaminador determina que la dirección de destino hace referencia a una máquina que se encuentra en la misma red que la máquina emisora, entonces el paquete se envía directamente hacia la máquina destino. Sin embargo, en el caso en el que las dos máquinas se encuentren en redes diferentes, el encaminador tiene que decidir qué ruta es la más efectiva, para lo que consulta sus tablas de direccionamiento.

La función de las tablas de direccionamiento es determinar cuál será el siguiente salto que tendrá que dar el paquete de datos para llegar a su destino. Para ello, cada nodo debe almacenar información relativa a cómo alcanzar determinados grupos de direcciones en función de la red en la que se encuentren. El aspecto de una tabla de enrutamiento en Windows es el observado en la figura 28:

Destino de red	Máscara de red	Gateway	Interfaz	Métrica
0.0.0.0	0.0.0.0	172.16.255.254	172.16.1.2	1
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0.1	1
172.16.0.0	255.255.0.0	172.16.1.2	172.16.1.2	20
172.16.1.2	255.255.255.255	127.0.0.1	127.0.0.1	20
172.16.255.255	255.255.255.255	172.16.1.2	172.16.1.2	20
255.255.255.255	255.255.255.255	172.16.1.2	172.16.1.2	1

Figura 28. Ejemplo de tabla de enrutamiento [61]

La información almacenada en estas tablas se recoge en los siguientes atributos [62]:

- **Destino de red**: se trata de la dirección IP que se va a comparar con la dirección IP de destino del paquete analizado. Lo primero que hace el encaminador es buscar una coincidencia exacta entre estas dos direcciones IP.
- **Máscara de red**: este atributo sirve para que, en caso de no coincidir la búsqueda exacta, el encaminador sea capaz de determinar en qué red está la máquina a la que se quiere entregar el paquete.
- **Puerta de enlace**: es la dirección a la que se envía el paquete una vez que se ha determinado la coincidencia.
- **Interfaz**: este atributo hace referencia a la dirección IP configurada para el adaptador de red utilizado para el envío de paquetes.
- **Métrica**: número que identifica el coste asociado a utilizar esa ruta determinada para alcanzar el destino y valorar si se trata del mejor camino. Este número puede hacer referencia a la velocidad del enlace, al número de saltos hasta el destino, a la fiabilidad del enlace o al tiempo que los paquetes tardan en llegar a la red de destino.

Estas tablas se pueden crear estáticamente, introduciendo una ruta de forma manual a través de comandos específicos según el sistema operativo, o bien de manera dinámica, gracias a diversos protocolos como son *Routing Information Protocol* o Protocolo de Información de Enrutamiento (RIP), *Open Shortest Path First* o Abrir la Ruta Más Corta Primero (OSPF), *Border Gateway Protocol* (BGP) o *Interior Gateway Routing Protocol* o Protocolo de Enrutamiento de *Gateway* Interior (IGRP) y su versión evolucionada *Enhanced Interior Gateway Routing Protocol* o Protocolo de Enrutamiento de *Gateway* Interior Mejorado (EIGRP).

Una vez que el encaminador ha determinado el siguiente salto que tiene que dar el paquete de datos lo envía, bien a otro encaminador, bien a la máquina de destino, y no vuelve a interactuar con él.

3.3.Conmutador o *switch*

Un conmutador o *switch* es un dispositivo que actúa sobre la capa de enlace de datos (nivel 2) del modelo OSI y que permite interconectar un conjunto de equipos formando una red local. Estos equipos se conectan al conmutador a través de puertos utilizando un interfaz de red Ethernet, es decir, mediante cables de par trenzado (*Registered Jack* (RJ)-45) o fibra óptica. A diferencia del encaminador, el conmutador no permite la interconexión entre distintas redes [63].

La función del conmutador consiste en transmitir los paquetes de datos entre los equipos de la red, cuya trama se puede observar en la figura 29. Para ello analiza las tramas Ethernet, concretamente las direcciones de origen y destino.

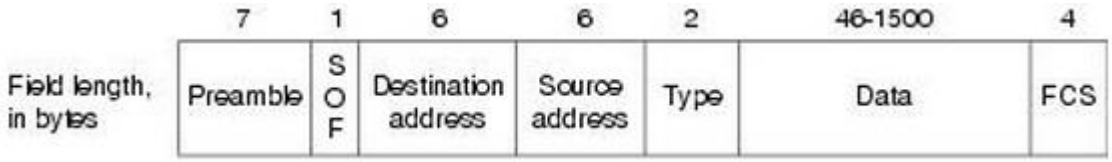

Figura 29. Trama Ethernet [64]

Estas direcciones, a las cuales se denomina MAC, son direcciones físicas que identifican de manera unívoca cada uno de los dispositivos conectados a la red. El conmutador es capaz de almacenar en una tabla una asociación entre la dirección MAC de una máquina y el puerto al que está conectada dicha máquina. Gracias a esta asociación, cada vez que al conmutador le llegue un paquete de datos analizará la trama, obtendrá la dirección MAC de destino, consultará por qué puerto ha de enviar el tráfico y finalmente redirigirá dicho paquete a través del puerto correcto.

La tabla, en un primer momento, está vacía. A medida que el conmutador va recibiendo paquetes guarda la dirección origen y la asocia al puerto por el que ha recibido dicho paquete, creando así una entrada en la tabla. De esta manera identifica que todos los paquetes que le llegan de un determinado puerto, pertenecen a una máquina específica Así, la próxima vez que al conmutador le llegue tráfico cuya dirección de destino sea una de las ya almacenadas en la tabla, sabrá por qué puerto redirigir el paquete. En el caso en el que la dirección destino no esté almacenada en la tabla el paquete de datos se redirigirá a todos y cada uno de los dispositivos conectados al conmutador.

Como conclusión, aunque el conmutador y el encaminador son elementos que en un primer momento parecen tener funciones similares, se ha podido comprobar que existen diferencias significativas entre ellos. A diferencia del conmutador, que sólo puede interconectar dispositivos dentro de la misma red local, el encaminador es capaz de conectar dos redes independientes. Además, el encaminador realiza sus funciones en el nivel 3 mientras que el conmutador actúa en el nivel 2.

3.4. Concentrador o *hub*

El concentrador o *hub* es un dispositivo que actúa en la capa física (nivel 1) del modelo OSI y que permite interconectar varios equipos dentro de una misma red. Al igual que ocurría con el conmutador, la conexión de estos equipos al concentrador se realiza mediante un cable Ethernet.

Pese a que la función del concentrador es parecida a la del conmutador, existe una gran diferencia entre ambos. El concentrador no tiene inteligencia, de manera que no analiza ninguno de los paquetes de datos que recibe. Esto supone que no conoce ni el emisor ni el destinatario de los datos. Por este motivo el concentrador envía las tramas que recibe a todos los equipos conectados a él. Además, a diferencia del conmutador, el concentrador es incapaz de enviar y recibir datos al mismo tiempo.

CAPÍTULO 4

ESPECIFICACIÓN DE REQUISITOS

4. Especificación de requisitos

4.1.Introducción

Durante este apartado se va a realizar una especificación de requisitos *software* para la aplicación desarrollada durante este PFG. La estructura de esta especificación sigue las directrices marcadas por el estándar IEEE *Recomended Practice for Software Requirement Especifications ANSI/IEEE 830 1998.*

4.1.1. Objetivo

El motivo por el cual se ha decidido llevar a cabo esta especificación de requisitos es definir de manera clara y concisa tanto las funcionalidades como las restricciones del sistema que se va a desarrollar. De esta manera, los compañeros del departamento en el que se va a aplicar dicho sistema podrán comprender los distintos aspectos del mismo, lo que les permitirá desarrollarlo o modificarlo con el objetivo de ampliar su funcionalidad. Además, cualquier lector del PFG podrá tener unas ideas generales de para qué sirve.

4.1.2. Ámbito de la aplicación

La idea sobre la que se desarrolla la aplicación es poder generar de manera automática un documento que contenga la configuración básica de un encaminador. Por este motivo el sistema se denomina Generador Automático de Ficheros de Configuración, GAFC.

La aplicación genera como resultado un fichero de texto con la configuración del encaminador. De esta manera el instalador responsable simplemente tendrá que cargar el contenido de ese fichero en el equipo correspondiente. La aplicación, en ningún caso, accederá automáticamente al encaminador para realizar la configuración.

El objetivo es reducir el tiempo empleado por cada instalador en la configuración básica de un encaminador cuando se produce un alta de nuevos servicios. Reduciendo estos tiempo se mejora la eficiencia y por tanto la productividad del departamento. Además, que los ficheros se generen de manera automática es un método que evita posibles errores humanos a la hora de aplicar los comandos y variables específicas de cada línea instalada.

4.1.3. Definiciones, acrónimos y abreviaturas

GAFC: Generador Automático de Ficheros de Configuración.

PFG: Proyecto Fin de Grado.

RFXX: cada requisito funcional se identifica de la siguiente manera:

- R = Requisito
- F = Funcional
- XX = Secuencia de dos dígitos que enumera cada requisito

COD3: identifica de manera unívoca al cliente con un código constituido por 3 letras que hacen referencia al nombre del mismo.

4.1.4. Visión General

Este capítulo consta de tres secciones. La primera de ellas es esta, denominada Introducción. La Introducción proporciona una visión general de la especificación de requisitos que se va a llevar a cabo durante las siguientes subsecciones. Durante la segunda sección, denominada Descripción, se le dará al lector una visión general del sistema cuyo objetivo es que conozca las principales funciones, restricciones, y todo lo relacionado con el desarrollo del mismo sin entrar en profundidad a describirlo. Durante la tercera sección, Requisitos Específicos, se definirán detalladamente los requisitos que debe cumplir.

4.2.Descripción

4.2.1. Perspectiva del producto

El GAFC será un sistema que funcionará encapsulado en una página web en la red interna de la compañía. Desde esta web se realizará una consulta sobre la base de datos de la empresa que proporcionará la información necesaria para poder generar el fichero de texto con la configuración del encaminador.

4.2.2. Funciones del sistema

Las principales funciones que el sistema brinda al usuario son las explicadas a continuación:

- **Búsqueda de datos**: permite al usuario buscar todos los datos necesarios para generar el fichero de configuración. Para ello, el usuario tendrá que conocer el número de tarea que identifica la línea a configurar. Tras realizar la búsqueda, se obtendrá por pantalla una tabla con los parámetros específicos de la línea buscada.
- **Modificación de datos**: una vez que el usuario comprueba los datos obtenidos, puede modificar cualquiera de las variables antes de generar la configuración. Es importante destacar que estos cambios sólo se verán reflejados en el fichero resultado. En ningún caso se podrán modificar los datos de las tablas de la base de datos consultada.
- **Generar CSV**: gracias a esta función el usuario puede generar un fichero previo al fichero de configuración en el que se almacenarán los parámetros necesarios para llevar a cabo la configuración. La aplicación principal leerá los parámetros de este fichero y, sin este, es imposible generar ninguna configuración.
- **Generar fichero de configuración**: esta funcionalidad permite generar directamente el fichero de configuración.

El diagrama de casos de uso que escenifica lo anterior se muestra en la figura 30:

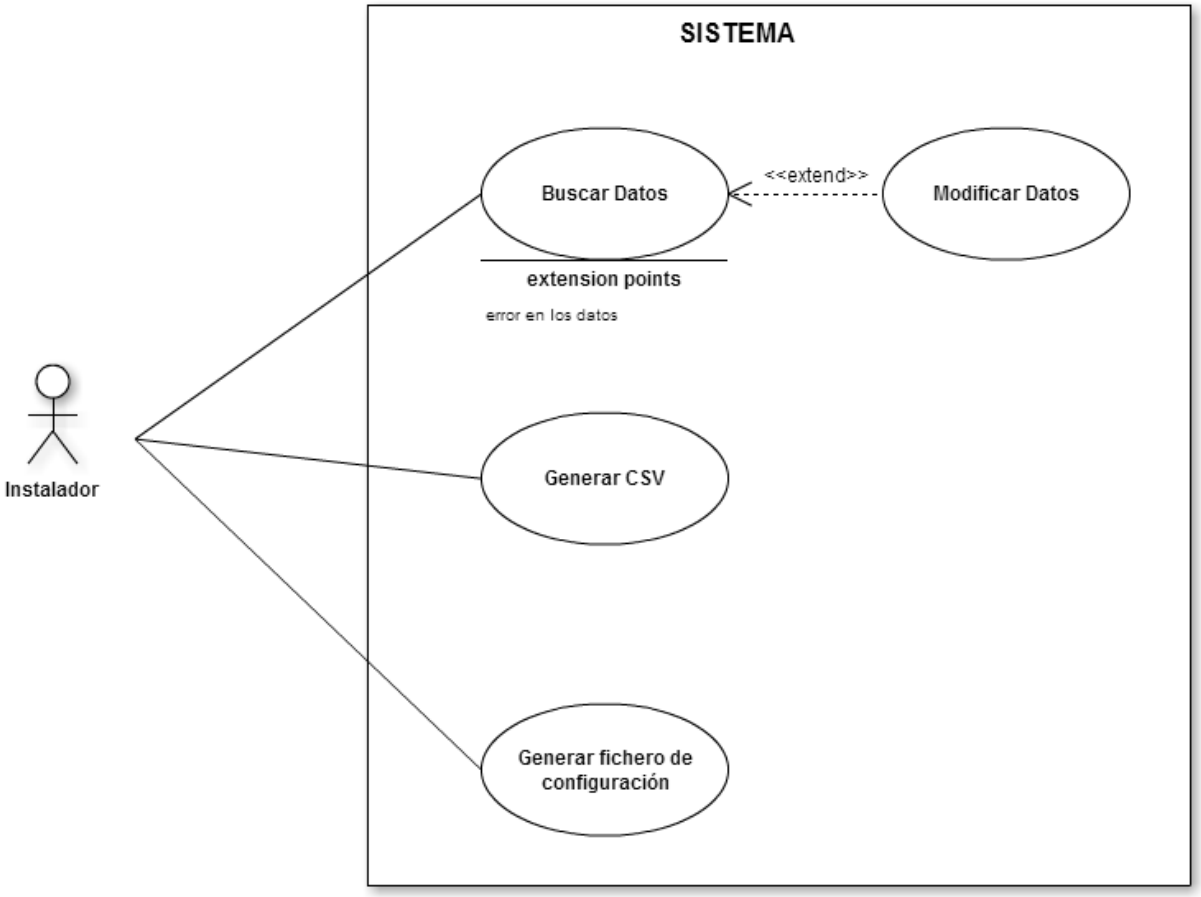

Figura 30. Diagrama de casos de uso

4.2.3. Características del usuario

La aplicación va dirigida al personal que compone el departamento *Customer Solutions and Engineering Delivery* o (CSED). Por lo tanto, el usuario no sólo tendrá un alto nivel de conocimiento en cuanto a aspectos técnicos relacionados con la configuración de equipos, ofimática y aplicaciones *software* sino que además conoce la topología de red y sus características.

4.2.4. Limitaciones de desarrollo

Este apartado se antoja de vital importancia para comprender el desarrollo de la aplicación en este PFG. Al ser un *software* que se va a poner en funcionamiento en un entorno determinado, que accede a información confidencial y que requiere de unas características concretas existen numerosas limitaciones, las cuales se han recogido en esta subsección:

- La aplicación se ha desarrollado en un entorno laboral y su funcionamiento ha de cumplir con la política de confidencialidad de datos de la empresa. Por ese motivo no se pueden especificar aspectos como topologías de red, datos reales de la base de datos y, en general, cualquier tipo de información que pueda comprometer bien a los clientes bien a la empresa.

- La web donde se encapsula el funcionamiento de la aplicación ha de ser soportada por Internet Explorer 7 o posterior, ya que este es el navegador utilizado oficialmente en la compañía.
- Algunos aspectos del diseño, desarrollo y funcionalidad vienen definidos por las necesidades o capacidades de los trabajadores que van a hacer uso y mantenimiento de esta aplicación.
- El servidor a utilizar en el que se ubicará la base de datos y el programa que procesa las consultas a la misma será Apache MySQL PHP Perl (XAMPP). Tal y como se define en el sitio web de XAMPP, "*es una distribución de Apache completamente gratuita y fácil de instalar que contiene MySQL, PHP y Perl*" [65]
- El sistema debe estar desarrollado en los siguientes lenguajes de programación:
    - JavaScript: pese a las desventajas en cuanto a seguridad y vulnerabilidad del código, este lenguaje se debe utilizar por varios motivos. En primer lugar, es un lenguaje sencillo de aprender, por lo que la persona encargada del mantenimiento y mejora de la aplicación podrá familiarizarse rápido con él y será capaz de hacer los cambios necesarios. En segundo lugar, es un lenguaje que permite tratar la información devuelta tras la consulta a la base de datos fácilmente.
    - PHP: se utiliza este lenguaje porque es con el que trabaja la compañía para el lado del servidor. Además, su capacidad de conexión con *MySQL* hace que sea un lenguaje idóneo para simular la consulta en la base de datos y el preparado de la información.
    - *MySQL*: es un sistema de gestión de bases de datos de código abierto. Se ha utilizado para crear una sencilla base de datos que hace las veces de la utilizada en el entorno laboral.
    - Java: lenguaje de programación orientado a objetos utilizado para generar el fichero de configuración del encaminador. Es un lenguaje conocido por la mayoría del personal del departamento.
    - Para la página web se utilizan *HyperText Markup Language* o Lenguaje de Marcas de Hipertexto (HTML) y *Cascading Style Sheets* u Hojas de Estilos en Cascada (CSS) por su sencillez y compatibilidad.
- Por petición expresa del personal del departamento, es necesario generar un fichero de información con extensión CSV que sea la fuente de donde la aplicación obtiene los datos para generar la configuración del encaminador. Estos no siempre estarán compuestos por los mismos parámetros, sino que dependerá del servicio del que se desea generar la configuración. Contienen las variables específicas de cada cliente, línea o tipo de servicio que se utilizarán para conformar el fichero resultado.

4.2.5. Suposiciones y dependencias

- La aplicación desarrollada funciona sobre el sistema operativo Windows.
- La versión de Java utilizada para el desarrollo de la aplicación es la 1.8.0_25.
- El navegador para el que el sitio web debe estar optimizado es Internet Explorer a partir de su versión 7.
- Los equipos en los que se va a ejecutar la herramienta deben estar conectados a la red de la empresa.

4.2.6. Requisitos futuros

- Acceder automáticamente al encaminador y volcar la configuración necesaria directamente.
- Realizar una batería de pruebas para comprobar la correcta configuración del equipo.
- Securizar el sistema de manera que quede registro de quien lo utiliza, por si alguna vez se intenta realizar un uso malintencionado.

4.3.Requisitos específicos

4.3.1. Requisitos funcionales

Los requisitos funcionales son aquellos que definen una serie de características que el sistema debe tener con el fin de satisfacer los deseos o necesidades del cliente. En este caso, los requisitos funcionales del GAFC son:

- **RF01**: el usuario podrá realizar una búsqueda de los datos introduciendo el número de tarea que identifica la línea de cuyo encaminador se ha de realizar la configuración y pulsando el botón *Buscar*. Los datos obtenidos se recogerán en una tabla la cual será mostrada por pantalla.
- **RF02**: se podrán modificar los datos obtenidos a gusto del usuario. Este, si tras analizar los datos obtenidos de la consulta comprueba que existe algún error podrá cambiar manualmente el contenido de cada una de las celdas ofrecidas en la tabla. Es importante destacar que estos cambios sólo tendrán efecto a la hora de crear el archivo CSV, no se modificará ningún valor en la base de datos.
- **RF03**: se podrá generar el archivo CSV sin necesidad de lanzar la aplicación que genera el fichero de configuración pulsando el botón *Generar CSV*. Esta posibilidad permitirá al usuario tener preparada la información necesaria en caso de que desee lanzar la aplicación en un momento determinado del día por el motivo que considere oportuno. Tras generarse el archivo se le mostrará al usuario la carpeta donde se ubica.

- **RF04**: se podrá lanzar la aplicación que genera el fichero de configuración de forma automática pulsando el botón *Generar fichero de configuración*. El usuario, tras asegurarse de que los datos que ha obtenido/modificado son los correctos puede pulsar el botón mencionado. Tras esto, se generará el archivo CSV que contiene la información de la tabla y se lanzará la aplicación que crea el archivo de texto con la configuración del encaminador. Una vez finalizada la acción se le mostrará al usuario el contenido de dicho fichero.
- **RF05**: el sistema debe comprobar que el archivo CSV que lee no está vacío, puesto que si se da esta circunstancia no se podrá realizar ninguna acción a partir de este momento. En caso de que el fichero leído esté vacío se notificará el error al usuario.
- **RF06**: el sistema realizará una comparación entre el número de parámetros leído y el número de parámetros que espera recibir en función al servicio que se va a configurar en el encaminador. Cada servicio tiene un archivo modelo cuya estructura debe respetar el CSV generado. En caso de que este número de parámetros sea diferente el programa debe detenerse y se informa al usuario del problema ocurrido.
- **RF07**: el sistema debe determinar qué servicio de los ofrecidos es el que se va a configurar. Cada servicio tiene una serie de características por lo que es fundamental dirimir este aspecto para llevar a cabo la correcta configuración del encaminador.
- **RF08**: el sistema debe diferenciar la marca del encaminador con el objetivo de ser fiel a las peculiaridades del mismo.
- **RF09**: el sistema debe generar un documento *EXtensible Markup Language* o Lenguaje de Marcas Extensible (XML) con el objetivo de validar los datos leídos del fichero fuente contra un esquema. En este estema se recogen todas las reglas que debe cumplir cada campo. En caso de este documento esté mal formado o que se incumpla una sola de las reglas definidas en el esquema se abortará la ejecución de la aplicación y se notificará el error producido al usuario.
- **RF10**: el sistema debe determinar qué tipo de conexión se va a configurar y para ello debe obtener el parámetro que hace referencia al protocolo de red utilizado. En función a este se determinará la plantilla base que se va a utilizar. El programa se va a apoyar en una serie de plantillas ya generadas por el personal del departamento en base a la marca del encaminador, al servicio y a las diferentes calidades del servicio.
- **RF11**: el sistema debe determinar la interfaz WAN, es decir, la interfaz que permite la comunicación entre el encaminador y la red ofrecida por el *Internet Service Provider* o Proveedor de Servicio de Internet ISP. Esta interfaz se configurará en función de tres aspectos: la marca del encaminador, el modelo del mismo y el tipo de servicio que se está instalando. Esta información se recogerá en un fichero específico denominado “formato_interface.txt”.

- **RF12**: el sistema debe ser capaz de generar la *password* específica para cada uno de los clientes. Para generar esta *password* el sistema debe conocer el COD3 del cliente o código de identificación. Este es un código que identifica inequívocamente al cliente y que se obtiene a partir del parámetro *userid*.
- **RF13**: el sistema debe determinar la interfaz virtual en función, nuevamente, al modelo del encaminador que se va a configurar.
- **RF14**: para realizar una correcta configuración el sistema debe ser capaz de realizar los cálculos necesarios para determinar correctamente tanto los porcentajes de ancho de banda contratados para cada clase de servicio como la propia tasa de transferencia de estas clases, expresada en Kbps.
- **RF15**: el sistema debe reservar un ancho de banda mínimo para asegurar la gestión del encaminador.
- **RF16**: una vez determinadas todas las características que ha de cumplir la configuración pedida por el cliente se procede a la confección de la configuración definitiva, reemplazando los valores genéricos de las plantillas parciales por los valores definitivos y conformando el archivo final.
- **RF17**: si la confección de la configuración se realiza de manera satisfactoria entonces el sistema deberá generar un archivo resultado que contenga dicha configuración.
- **RF18**: en caso de que la confección de la configuración se vea truncada por algún motivo el sistema debe crear un archivo de error en el que almacenará el problema que ha tenido como consecuencia el fallo en la creación del archivo de configuración del encaminador. Así mismo se notificará al cliente dónde puede encontrar dicho archivo para revisar las causas del error.
- **RF19**: el sistema debe de ir generando de manera dinámica un archivo que haga las funciones de LOG, es decir, se deben ir guardando los diferentes eventos ocurridos así como los posibles errores que se puedan ir produciendo de manera que en un simple vistazo a este archivo podamos ver qué ha hecho la herramienta, a qué hora, qué provocó el fallo de la aplicación, si la generación del fichero se realizó de manera satisfactoria y toda clase de marcas o hitos destacables.

4.3.2. Requisitos no funcionales

4.3.2.1. Requisitos de rendimiento

- Para la base de datos diseñada en este PFG, el tiempo de respuesta tras realizar una consulta debe ser inferior a 5 segundos.
- El tiempo invertido en generar el archivo CSV y/o el fichero de texto con la configuración del encaminador debe ser inferior a 5 segundos.

4.3.2.2. Seguridad

En el programa desarrollado en el servidor que consulta a la base de datos se han desarrollado unas técnicas que impiden uno de los ataques más comunes y peligrosos: la inyección de código *Structured Query Language* o Lenguaje de Consulta Estructurado (SQL). Estas técnicas son el escapado de la cadena a buscar y la aseveración de que esta cadena, que debe ser un identificador de tarea, es numérica.

4.3.2.3. Sencillez

El objetivo del sistema es disponer de una interfaz web muy sencilla e intuitiva que permita llevar a cabo acciones en el menor tiempo posible y de la manera más simple.

4.3.2.4. Mantenibilidad

El sistema debe disponer de la documentación necesaria para que las operaciones de mantenimiento se realicen con el menor esfuerzo técnico y temporal posible.

4.3.2.5. Extensibilidad

El sistema debe ser extensible. Es fundamental que se puedan implementar nuevas características sin afectar al funcionamiento de lo ya desarrollado con el fin de mejorar o ampliar la utilidad del mismo.

CAPÍTULO 5

JUSTIFICACIÓN DE LA TECNOLOGÍA EMPLEADA

5. Justificación de la tecnología empleada

En este apartado se va a explicar por qué se han seleccionado las diferentes tecnologías para llevar a cabo el desarrollo del sistema.

Tal y como se describió en el apartado 4.2.4, los lenguajes de programación utilizados en este PFG son: HTML, CSS, JavaScript, PHP, MySQL y Java.

5.1.HTML

HTML, siglas de *HyperText Markup Language,* es un lenguaje de marcado que permite la creación de páginas web [66].

Diseñado y mantenido por la *World Wide Web Consortium,* más conocida como W3C, este lenguaje se caracteriza por la utilización de etiquetas. Estas etiquetas permiten describir el aspecto visual que se desea en una página web, es decir, determinan cómo debe aparecer el texto, imágenes u otros elementos en el navegador. Otra de las principales ventajas con las que cuenta HTML es que puede ser modificado con cualquier editor, por rudimentario que sea, lo que le da una gran versatilidad.

La página web ha sido diseñada con este lenguaje por su sencillez y su capacidad de organizar el contenido de la misma, pudiendo definir tablas, imágenes, enlaces o formularios.

5.2.CSS

CSS, o *Cascading Style Sheets,* es un lenguaje de hojas de estilo desarrollado por el W3C que define la forma de mostrar un documento web definido en HTML o XML. Gracias a CSS se puede separar el contenido de la web de la presentación de tal manera que el desarrollador es capaz de gestionar el aspecto de una o varias páginas web al mismo tiempo. Esta separación también supone una simplificación de los archivos HTML puesto que no se repite código en aquellos elementos que impliquen el mismo estilo. El código CSS puede desarrollarse en el propio documento HTML o bien puede ser un fichero distinto que es referenciado desde el HTML. Dependiendo de dónde se sitúe el código podemos hablar de [67]:

- CSS Externo: el código se desarrolla en un archivo diferente al HTML. Ambos archivos han de estar vinculados. Separa completamente el contenido de la web de la presentación.
- CSS en el documento HTML: el código se desarrolla en la cabecera del archivo HTML. Separa el código CSS del HTML pero ambos se encuentran en el mismo archivo.
- CSS en elementos HTML: el código se desarrolla directamente en el cuerpo de la página web. En este caso ambos lenguajes están mezclados continuamente.

Cada una de las reglas que rigen el estilo del documento consiste en un selector, que identifica los elementos del código HTML afectados por la declaración, y una declaración, la cual va entre corchetes y especifica cómo se va a ver el elemento seleccionado.

Este lenguaje reúne unas características idóneas para utilizarlo como herramienta para dar formato a la página web diseñada.

5.3.JavaScript

JavaScript es un lenguaje interpretado por los navegadores del lado del cliente, por lo que no es necesario compilar los programas desarrollados para ejecutarlos. Pese a que la sintaxis es similar a la de Java, no podemos confundir ambos lenguajes.

Permite desarrollar páginas web dinámicas dónde se pueden programar automatismos que interactúan con el usuario o crean efectos visuales, lo que ayuda a generar una sensación de comunicación entre la web y el usuario de la misma.

La elección de desarrollar algunas funcionalidades de la web en este lenguaje se ha tomado tras analizar, sobre todo, dos aspectos. En primer lugar, la similitud en la sintaxis con Java. Los compañeros que pueden mantener y desarrollar la herramienta saben programar en este lenguaje. En segundo lugar, el no tener un servidor web en propiedad donde alojar páginas diseñadas en otros lenguajes de servidor, como PHP o *Active Server Pages* o Páginas de Servidor Activo (ASP), hace que JavaScript sea idóneo para tratar los datos obtenidos.

5.4.PHP

PHP, o *HyperText Preprocessor*, es un lenguaje *Open Source* utilizado para el desarrollo web que puede ser insertado dentro del código HTML [68].

Pese a que el concepto es bastante parecido a lo comentado en el apartado anterior, lo que caracteriza a PHP es que los programas se ejecutan en el lado del servidor de manera que el usuario recibirá el resultado de su ejecución y no tendrá constancia del código que ha generado dicho resultado.

Para que los programas desarrollados con PHP funcionen necesitan un intérprete, el cual debe estar instalado en un servidor web, en el que se alojan los *scripts*.

Cuando el cliente realiza una petición para obtener el resultado de un programa desarrollado en PHP, el servidor que recibe dicha petición ejecuta el intérprete, procesa el programa y obtiene un resultado, el cual es enviado de vuelta al lado cliente que realizó la petición. Las principales ventajas de usar este lenguaje son:

- Es un lenguaje multiplataforma ya que puede emplearse en los principales sistemas operativos.
- Admite muchos de los servidores web utilizados hoy en día, como por ejemplo Apache o *Internet Information Services* o Servicios de Información de Internet (IIS).
- Capacidad de conexión con un gran número de bases de datos.
- *Software* libre y código abierto, lo que posibilita un fácil acceso para cualquier desarrollador.

- Permite utilizar programación orientada a procedimientos, programación orientada a objetos o una mezcla entre ambas.

En este trabajo, PHP se ha utilizado para generar un programa que permite realizar una consulta a la base de datos, que hace las funciones de la utilizada en la empresa, y devolverle el resultado de la misma a la página web utilizada por el usuario, desde la cual se realiza la petición de información.

5.5.MySQL

MySQL es un sistema de administración de bases de datos relacionales, por lo que la información almacenada en estas bases de datos se distribuirá en tablas, proporcionando velocidad y flexibilidad. Una de sus características principales es que permite que múltiples usuarios puedan ejecutar distintas tareas sobre las bases de datos alojadas en un servidor. Otra de las características que lo hacen destacar por encima de los demás es que es un *software* de código abierto.

En función del tipo de almacenamiento de cada tabla tendremos uno de los motores de almacenamiento ofrecidos por *MySQL*. Un motor de almacenamiento es el conjunto de técnicas y funcionalidades aplicadas en el almacenamiento de la información, es decir, cómo se guardan los datos en una tabla. Los motores más conocidos son [69]:

- MyISAM: motor de almacenamiento por defecto, se utiliza en aquellas bases de datos en las que no se necesita modificación concurrente de datos. Por tanto, un entorno en el que se realicen consultas de lectura de datos es el ideal para este motor.
- InnoDB: es un motor de almacenamiento más robusto que se utiliza para asegurar la integridad de los datos en las bases transaccionales en las que hay modificación concurrente de datos.

Aunque el motor MyISAM puede dar mejor rendimiento en bases de datos con pocas escrituras, en este caso se ha decidido utilizar el motor InnoDB por dos motivos. En primer lugar, para este PFG la diferencia de rendimiento entre ambos motores no es significativa, en segundo lugar, las funcionalidades que ofrece InnoDB, como por ejemplo el uso de las claves foráneas para garantizar la integridad referencial, no las ofrece MyISAM.

5.6.Java

Java es un lenguaje orientado a objetos que se utiliza para desarrollar aplicaciones para un amplio número de entornos, siempre y cuando dichos entornos tengan instalada la máquina virtual Java. Esto es gracias a que lo que realmente interpreta la máquina virtual son los ficheros de clases compilados, por lo que la ejecución de los programas desarrollados en Java es independiente de la plataforma.

Otra de las características que hacen de Java un lenguaje muy competitivo es que se dispone de la *Application Programming Interface* o Interfaz de Programación de Aplicaciones (API) de forma totalmente gratuita y es *open source*, por lo que los desarrolladores pueden estudiar en profundidad en incluso desarrollar el código nativo [70].

Se ha elegido este lenguaje para desarrollar la aplicación que genera el fichero que contiene la configuración del encaminador porque, además de que es el lenguaje con el que se está más familiarizado por el hecho de ser el que se ha utilizado en muchas de las asignaturas de la carrera, al igual que ocurría con JavaScript, es el que mejor conocen los compañeros del departamento que van a mantener esta herramienta.

5.7.Servidor Web

Un servidor es una máquina que brinda servicio a otros equipos, a los cuales se les denomina clientes. Si centramos la definición en los servidores web, se puede decir que la función principal de estos es alojar páginas web o aplicaciones a las cuales un cliente accede a través de Internet. Por tanto, un servidor web estará a la espera de que un cliente realice la petición de uno de sus recursos a través del navegador, para lo que se utiliza el protocolo HTTP.

Los servidores más utilizados en la actualidad son Apache e IIS. Apache es el más utilizado a nivel global. Además de ser gratuito y *open source*, es el que aplican la gran mayoría de proveedores de páginas web, se integra fácilmente con PHP y tiene muchos módulos para extender su funcionalidad. Además, es un servidor web multiplataforma, a diferencia de IIS, el cual funciona sólo sobre plataformas Windows.

El servidor utilizado para alojar la base de datos y el programa que realiza la consulta sobre la misma y devuelve el resultado a la página web desde la que el cliente realiza la petición es Apache.

5.8.XAMPP

La instalación de los programas requeridos para la simulación de la consulta a la base de datos de la empresa, PHP, *MySQL* y Apache, se ha llevado a cabo mediante la aplicación XAMPP por su eficiencia y facilidad de uso.

XAMPP es un servidor independiente de *software* libre que permite instalar de forma sencilla una serie de módulos, entre los que destacan la base de datos *MySQL*, el servidor web Apache y los intérpretes para lenguajes PHP y *Perl*, en cualquier sistema operativo. Esta herramienta permite crear un servidor web local de prueba para los desarrolladores, por lo que no necesitas conexión a Internet para probar el trabajo realizado.

Una de las herramientas fundamentales utilizadas en el desarrollo de la base de datos ha sido *phpMyAdmin*. Este *software* está diseñado para administrar y gestionar las bases de datos *MySQL* a través de una interfaz web. Esta gestión permite crear, modificar y eliminar tanto bases de datos como tablas y campos de estas. Además, también permite ejecutar cualquier declaración en lenguaje SQL.

CAPÍTULO 6

ANÁLISIS DEL SISTEMA

6. Análisis del sistema

Durante el siguiente capítulo se desarrolla la especificación del sistema, la cual se apoyará en diagramas *Unified Modeling Language* o Lenguaje Unificado de Modelado (UML) [71], con el objetivo de explicar en detalle la implementación final del mismo.

6.1.Descripción

El sistema se puede dividir en tres partes:

- **Servidor web y base de datos**: en este servidor se alojarán tanto la base de datos como el programa desarrollado en PHP que, cuando le llega una petición del cliente, realiza una consulta a la misma. La base de datos simula, en menor escala, la base de datos que contiene toda la información de los clientes de la empresa.
- **Página web del cliente**: esta página es a la que tiene acceso el instalador del departamento. A través de ella se pueden consultar los datos de cliente, modificar dichos datos, y lanzar la aplicación que generará el fichero que contiene la configuración del encaminador.
- **Aplicación**: esta es la encargada de seleccionar y procesar las distintas plantillas base, aplicar los cambios de variables necesarios en cada línea de estas y finalmente crear el fichero de texto cuyo contenido recoge todos los comandos necesarios para llevar a cabo la configuración base de un encaminador con el objetivo de dar el servicio contratado por el cliente.

En la figura 31se muestra el diagrama de subsistemas, que representa gráficamente los tres grandes bloques de los que está compuesto el sistema así como la relación de uso entre ellos:

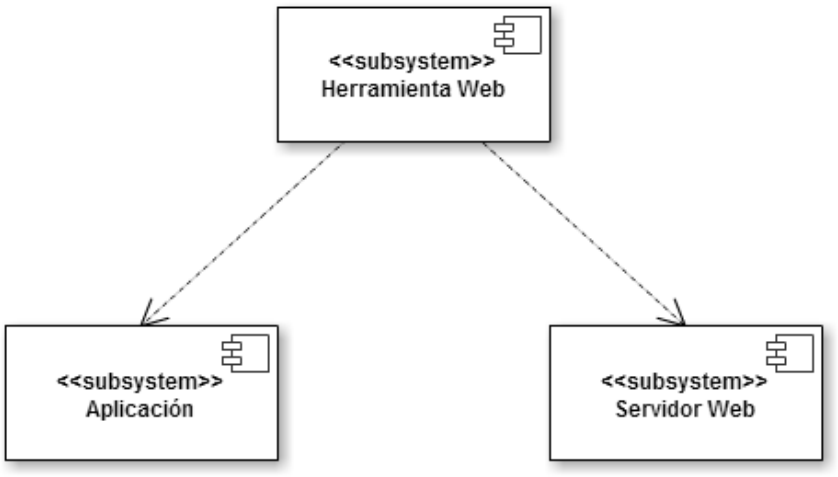

Figura 31. Diagrama de subsistemas

Tras presentar el sistema desarrollado con una visión al más alto nivel, se realiza un estudio más detallado. Para ello, se utiliza el diagrama de componentes, en la figura 32. El objetivo de este diagrama es visualizar la estructura general del sistema y el comportamiento de los servicios ofrecidos por cada uno de los componentes:

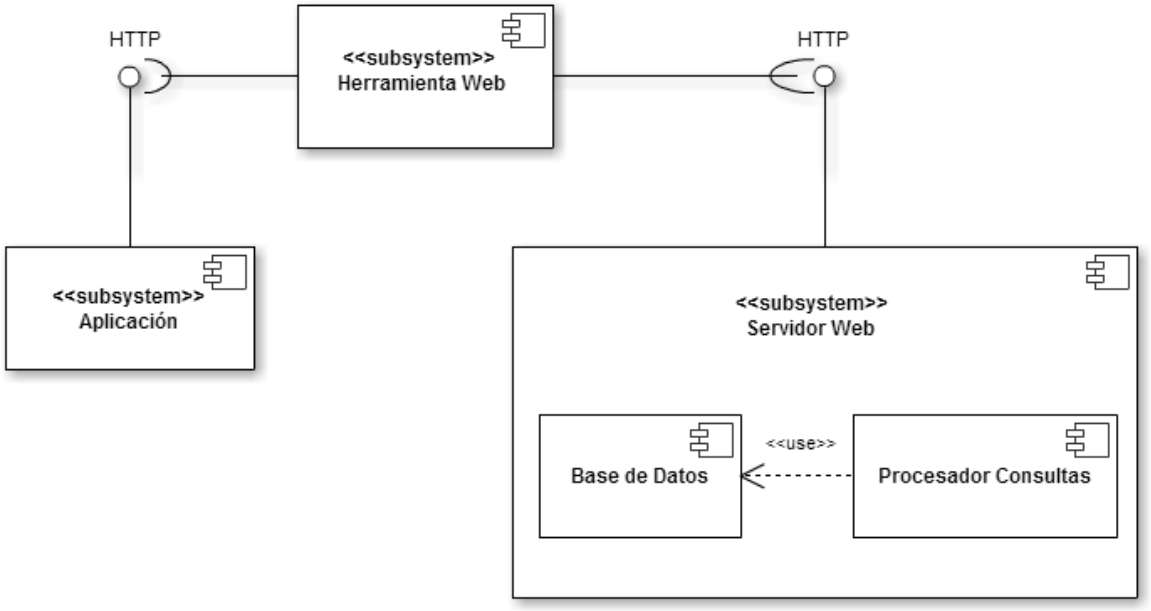

Figura 32. Diagrama de componentes

Los usuarios que utilicen este sistema deben conocer el número que identifica de manera unívoca cada línea de la cual tiene que realizarse la configuración del encaminador. Este número será el que introduzcan en la caja de texto que la página web les presentará nada más abrirse en el navegador. En el momento en el que se pulsa el botón *Buscar* se realiza una petición al servidor web. El programa alojado en el servidor realiza a su vez una consulta sobre la base de datos de la que obtiene la información necesaria para generar la configuración correctamente. El servidor devuelve el resultado a la página web del cliente, que presenta dicho resultado en forma de tabla. El usuario, tras revisar los datos, puede mantenerlos o modificarlos antes de elegir qué opción desea llevar a cabo, generar un archivo con extensión CSV en el que se almacene la información obtenida, o lanzar directamente la aplicación que genera el fichero de texto con la configuración.

La secuencia de acción expuesta anteriormente puede verse en las figuras 33 y 34:

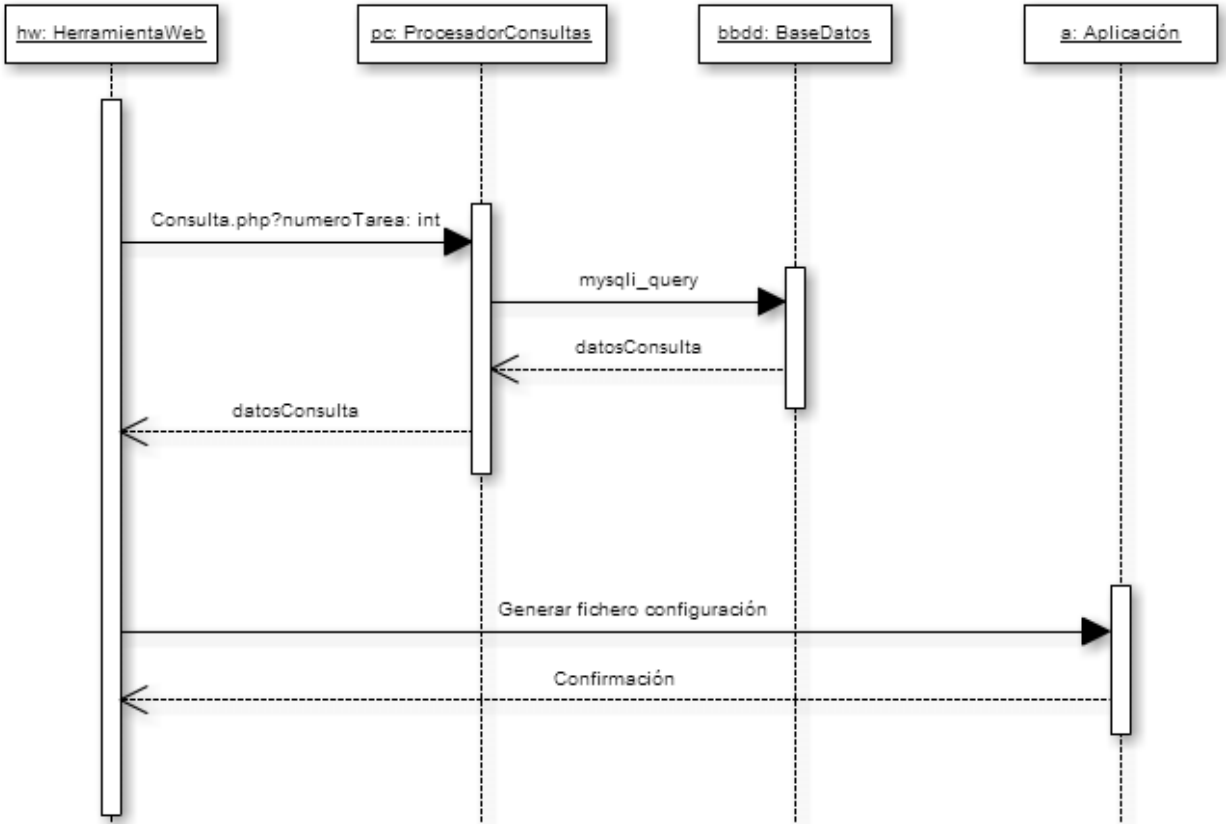

Figura 33. Diagrama de secuencia: generar fichero de configuración

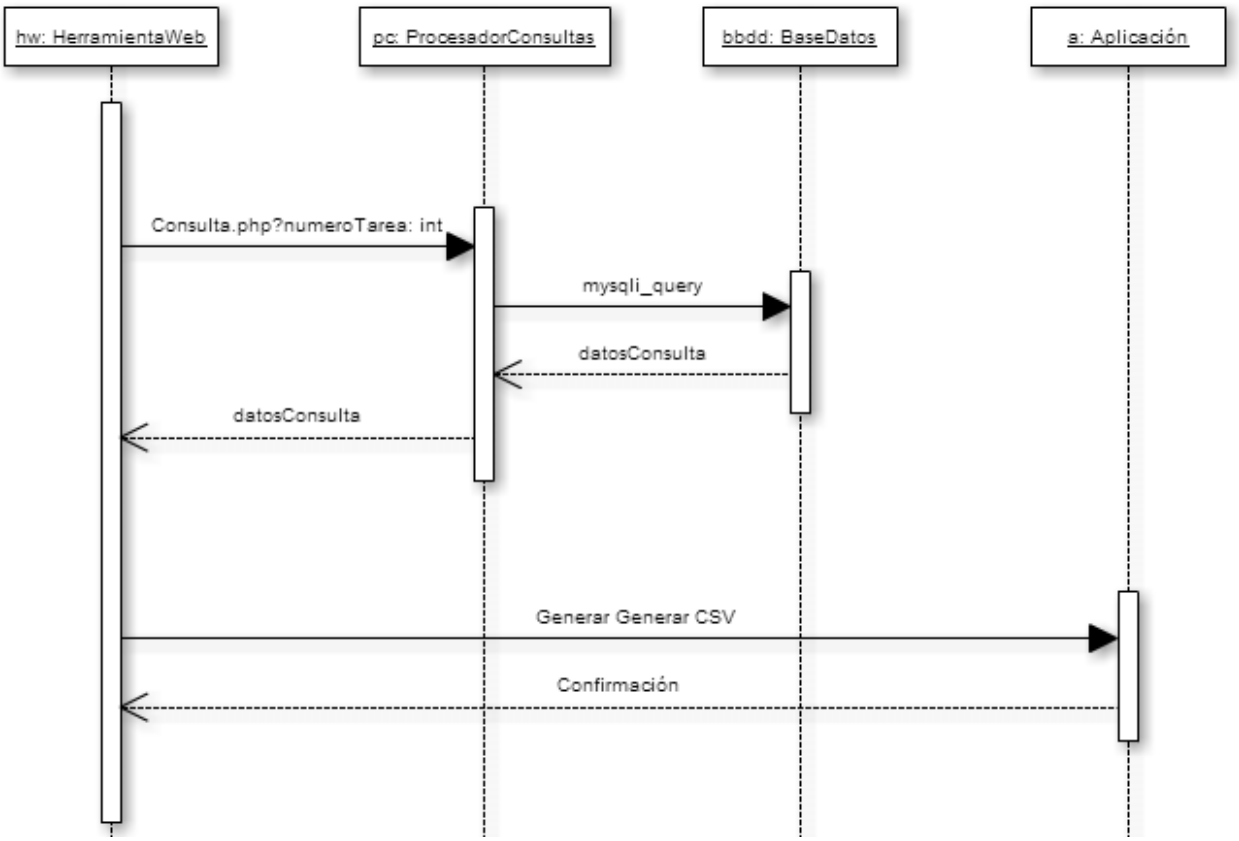

Figura 34. Diagrama de secuencia: generar CSV

Para finalizar este apartado de descripción del sistema, se ha desarrollado un último diagrama, el de actividad, en la figura 35. El objetivo es mostrar el flujo de trabajo del sistema a través de las acciones y decisiones tomadas por el usuario:

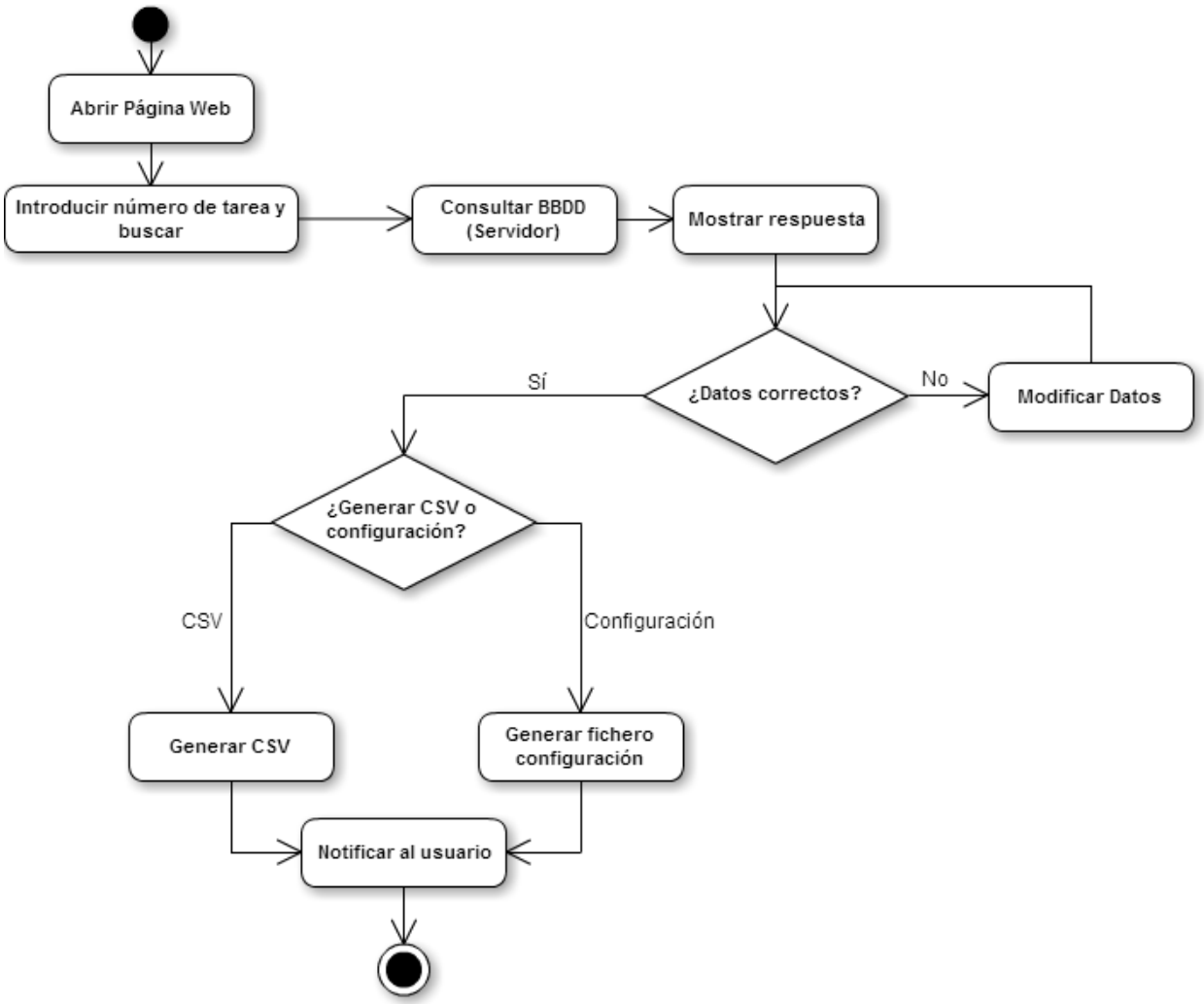

Figura 35. Diagrama de actividad: sistema

6.2.Diseño de la base de datos

Durante esta subsección se pretende presentar al lector el diseño de la base de datos. Esta se ha diseñado con el objetivo de simular, de forma simplificada, la base de datos de la compañía. La idea es que en el entorno real en el que este sistema se utilizará, las consultas realizadas desde la página web se hagan a un servidor que contenga una copia de la base de datos ya que, por políticas de seguridad, no es posible acceder directamente a la original.

Sin embargo, como esos datos son confidenciales, la solución adoptada ha sido crear una base de datos en un servidor local que permita realizar una simulación completa del funcionamiento del sistema.

Tal y como se expuso en el punto 5.5, la base de datos creada sigue un modelo relacional, por lo que los elementos se pueden relacionar sin necesidad de duplicar información. Estas relaciones se establecen en base a las claves, ya sean primarias o externas y gracias al uso de SQL se puede obtener información de varias fuentes en una misma consulta.

6.2.1. Tablas de la base de datos

La base de datos está formada por 13 tablas. Estas contienen los datos relevantes de los clientes, de manera que cuando se realice una consulta se establecerán relaciones entre algunas de ellas para obtener toda la información necesaria.

Con el objetivo de ser tan fiel a la base de datos real como sea posible, todos los datos que recogen información técnica de las tecnologías instaladas en cada cliente se han clasificado, según su tipo, en unas tablas cuyo prefijo siempre es CT, o lo que es lo mismo, Componente Técnico. El resto de las tablas tienen nombres que permiten interpretar qué información almacenan. Además, puede darse el caso de tener valores repetidos en distintas tablas. Esto se debe a que, en el entorno laboral, un mismo dato puede estar definido en dos componentes técnicos distintos, lo que no supone que ambos tengan información ya que uno de ellos puede venir vacío. Por tanto, para asegurar que se obtiene el dato, ya sea de un componente o del otro, se incluyen ambos. Las tablas, las cuales analizaremos en detalle a continuación, son:

- **clientes**
- **ct_3g**
- **ct_alta_genérica**
- **ct_ivpn**
- **ct_linea**
- **ct_qosid**
- **ct_router**
- **ct_sap**
- **operadoras**
- **proyectos**
- **sites**
- **tareas**
- **tareas_ct**

Antes de entrar en el detalle de cada tabla y como norma general, es importante destacar que todas las direcciones IP o máscaras de red se han definido como *varchar* de longitud máxima de dieciocho caracteres. Esta longitud está condicionada porque en algunas direcciones puede venir la máscara en notación *Classless Inter-Domain Routing* o Enrutamiento entre Dominios sin Clases (CIDR), por lo que a los quince caracteres como máximo que podría tener una dirección IP (tres correspondientes a los puntos que separan los grupos numéricos y doce correspondientes a los dígitos de la dirección, tres como máximo por cada uno de los cuatro grupos) se le deben añadir los posibles tres (el símbolo "/" y dos dígitos) de este tipo de notación. Esta manera de definir las direcciones IP puede generar errores de formato, los cuales son analizados mediante el manejador de errores descrito en el apartado 6.5.2.

Tabla 'clientes'

Esta tabla guarda una sencilla relación entre el nombre de cada uno de los clientes de la base de datos y un número que lo identifica.

- *CODIGO_ORGANIZACION*: este número identifica de manera única a cada cliente en la base de datos. Actúa como clave primaria de la tabla de manera que dos clientes nunca podrán ser identificados con el mismo código. El dato se ha definido como *smallint* sin signo de manera que permitiría almacenar hasta un total de 65535 clientes.
- *CLIENTE*: esta cadena hace referencia al nombre del cliente. Se define como *varchar* cuya longitud máxima es de treinta caracteres.

La estructura de la tabla es la mostrada en la figura 36:

Columna	Tipo	Nulo
CODIGO_ORGANIZACION	smallint(5)	No
CLIENTE	varchar(30)	No

Figura 36. Estructura de la tabla 'clientes'

Tabla 'ct_3g'

Esta tabla almacena información relacionada con las características técnicas de una línea 3G.

- *CTID*: identificador del componente técnico. Todos los componentes técnicos de este tipo comenzarán por tres. Ejemplo: 3XX. Se ha definido como tipo *smallint* sin signo y se ha limitado a tres dígitos de longitud máxima.

  Para implantarlo en el entorno laboral sería necesario o aumentar la limitación de dígitos hasta cinco, siendo 301 el primer componente y 39999 el último o, en caso de necesitar aún más, cambiar el tipo de dato numérico.

  Actúa como clave primaria de la tabla de manera que se asegura que no puede haber dos componentes técnicos de este tipo con el mismo identificador.

Además, también actúa como *FOREIGN KEY* o clave foránea puesto que se vincula con la clave primaria de la tabla *TAREAS_CT* de manera que no podrá existir de forma independiente sino que tendrá que estar asociado a una tarea. Esta relación se crea de manera que en caso de borrar o actualizar uno de los registros de la tabla padre se realizará la misma operación en los registros asociados de la tabla hija.

- *SIM*: número de teléfono del abonado. Se elige tipo de dato *varchar* porque no se realizan operaciones aritméticas con este campo.
- *APN*: nombre del punto de acceso que contiene la configuración necesaria para establecer una conexión entre el dispositivo móvil e Internet. Se define como *varchar* cuya longitud máxima es de veinte caracteres.
- *NOMBRE_IVPN*: identificador de la red privada configurada para cada cliente. Se define como *varchar* cuya longitud máxima es de veinte caracteres.
- *QOSID*: identificador que hace referencia a la distribución de los tráficos prioritarios. Aporta información acerca del tipo de línea, velocidad de línea, velocidad de puerto y distribución de las clases de servicio. El tipo de dato seleccionado es *varchar* de longitud máxima de cinco caracteres puesto que no se realizan operaciones aritméticas con él.
- *NOMBRE_USUARIO*: este campo hace referencia al identificador de usuario, el cual es único para cada cliente. Se define como *varchar* cuya longitud máxima es de quince caracteres y está formado por:
  - *COD3*: código de tres letras que identifica a cada uno de los clientes.
  - *Nomenclatura de línea:* conjunto de caracteres que identifican la línea instalada.
  - *SITEID*: código que identifica cada una de las localizaciones en las que hay instalada una línea. Estos códigos están compuestos por una S mayúscula acompañada de un número de tres cifras que crece secuencialmente a medida que se van añadiendo nuevas localizaciones a la base de datos.
- *DIRECCION_IP_FIJA*: dirección WAN del equipo. Es la dirección IP asignada por el ISP a través de la cual el encaminador es capaz de conectarse a Internet.
- *IP_GESTION*: dirección IP asociada a la interfaz de gestión del equipo. Es la dirección mediante la que los técnicos pueden acceder al encaminador para realizar distintas operaciones.
- *NEMÓNICO:* Nombre que identifica el dominio al que se conecta la línea del cliente. Se define como *varchar* cuya longitud máxima es de tres caracteres.
- *IP_LAN*: dirección LAN del equipo. Esta es la dirección que permitirá la comunicación entre el encaminador y todos los equipos conectados al mismo dentro de la red local

- *MASCARA_IP_LAN*: máscara de subred que delimita qué parte de la dirección IP hace referencia a la red y qué parte hace referencia a los equipos que puede haber en dicha red.
- *PRIORIDAD*: en caso de que en una misma sede exista más de una línea, mediante este campo se establece la ruta preferida, siendo 1 la prioridad máxima. Se escoge el tipo numérico más pequeño, *tinyint*, sin signo y se limita a un solo dígito puesto que este campo, en ningún caso, superará el valor 9.

La estructura de la tabla es la mostrada en la figura 37:

Columna	Tipo	Nulo	Enlaces a
CTID	smallint(3)	No	tareas_ct -> CTID
SIM	varchar(9)	No	
APN	varchar(20)	No	
NOMBRE_VPN	varchar(20)	No	
QOSID	varchar(5)	No	
NOMBRE_USUARIO	varchar(15)	No	
DIRECCION_IP_FIJA	varchar(18)	No	
IP_GESTION	varchar(18)	No	
NEMONICO	varchar(3)	No	
IP_LAN	varchar(18)	No	
MASCARA_IP_LAN	varchar(18)	No	
PRIORIDAD	tinyint(1)	No	

Figura 37. Estructura de la tabla 'ct_3g'

Tabla 'ct_alta_generica'

Esta tabla almacena información relacionada con las características técnicas de una línea ADSL.

- *CTID*: identificador del componente técnico. Todos los componentes técnicos de este tipo comenzarán por cuatro. Ejemplo: 4XX. Se ha definido como tipo *smallint* sin signo y se ha limitado a tres dígitos de longitud máxima. Para implantarlo en el entorno laboral sería necesario o aumentar la limitación de dígitos hasta cinco, siendo 401 el primer componente y 49999 el último o, en caso de necesitar aún más, cambiar el tipo de dato numérico.

  Actúa como clave primaria de la tabla de manera que se asegura que no puede haber dos componentes técnicos de este tipo con el mismo identificador. Además, también actúa como *FOREIGN KEY* o clave foránea puesto que se vincula con la clave primaria de la tabla *TAREAS_CT* de manera que no podrá existir de forma independiente sino que tendrá que estar asociado a una tarea. Esta relación se crea de manera que en caso de borrar o actualizar uno de los registros de la tabla padre se realizará la misma operación en los registros asociados de la tabla hija.

- *TELEFONO*: representa el número de teléfono del cliente. Se elige tipo de dato *varchar* por dos motivos: no se realizan operaciones aritméticas con este campo y puede haber letras en el mismo. Su longitud máxima es de quince caracteres.
- *ADM_ADSL*: número de doce dígitos que identifica la línea ofrecida por el proveedor de servicios. El tipo de dato seleccionado es *char* porque no se van a realizar operaciones aritméticas con el mismo y su longitud es fija.
- *DIRECCION_GESTION*: dirección IP asociada a la interfaz de gestión del equipo. Es la dirección a través de la cual los técnicos pueden acceder al encaminador para realizar distintas operaciones.
- *IP_FIJA_WAN*: dirección WAN del equipo. Es la dirección IP asignada por el ISP a través de la cual el encaminador es capaz de conectarse a Internet.
- *TIPO_CONEXION*: contiene el tipo de protocolo utilizado en la línea. Las opciones son *Point-to-Point Protocol Over Ethernet* o Protocolo Punto a Punto Sobre Ethernet (PPPoE) o *Point-to-Point Protocol Over ATM* o Protocolo Punto a Punto Sobre ATM (PPPoA). La principal diferencia entre ambos es que PPPoE encapsula el *Point-to-Point Protocol* o Protocolo Punto a Punto (PPP) sobre una capa Ethernet mientras que el protocolo PPPoA lo hace sobre una capa ATM. Como la longitud del campo siempre va a ser cinco, el tipo de dato seleccionado es *char*.
- *TIPO_BACKUP*: este parámetro sirve para identificar si el encaminador configurado es el principal o por el contrario se trata de un encaminador *backup* instalado para balancear el tráfico por él en caso de fallo del principal. Se limita su longitud a cuatro porque este es el tamaño de la cadena más grande que se almacenará en el campo.
- *Virtual Path Identifier* o Identificador de Camino Virtual (*VPI)*: es un parámetro del protocolo ATM que, junto al *Virtual Channel Identifier* o Identificador de Canal Virtual (VCI), sirve para determinar el siguiente destino de la celda que se está transmitiendo por la red. Este parámetro identifica el camino virtual que debe seguir una celda a través del enlace físico, el cual está compuesto por un grupo de *Virtual Path* o Camino Virtual (VP). Se escoge el tipo numérico más pequeño, *tinyint,* sin signo puesto que este campo, en ningún caso, superará el valor 255. Su tamaño se limita a dos porque no se utilizarán números mayores de dos cifras.
- *VCI*: es un parámetro del protocolo ATM que, junto al VPI, sirve para determinar el siguiente destino de la celda que se está transmitiendo por la red. Este parámetro identifica el canal virtual por el que conectan los dos extremos de una comunicación a través del VP, el cual está compuesto por un grupo de VC. Se escoge el tipo numérico más pequeño, *tinyint,* sin signo puesto que este valor, en ningún caso, superará el valor 255. Su tamaño se limita a dos porque no se utilizarán números mayores de dos cifras.
- *TECNOLOGIA*: tipo de tecnología utilizada en la línea, que puede ser ADSL, ADSL2+, VDSL o FTTH. Por este motivo se limita su tamaño máximo a seis caracteres y se define como *varchar*.

- *PROVEEDOR*: operadora a la que se alquila el circuito por el que se da servicio al cliente. Se escoge el tipo numérico más pequeño, *tinyint*, sin signo y limitado a un dígito como máximo puesto que, en ningún caso, superará el valor 9. Este parámetro se configura como *FOREIGN KEY* o clave foránea puesto que se vincula con la clave primaria de la tabla *OPERADORAID*. Esta relación se crea de manera que en caso de borrar o actualizar uno de los registros de la tabla padre se realizará la misma operación en los registros asociados de la tabla hija.
- *QOSID*: identificador que hace referencia a la distribución de los tráficos prioritarios. Aporta información acerca del tipo de línea, velocidad de línea, velocidad de puerto y distribución de las clases de servicio. El tipo de dato seleccionado es *varchar* puesto que no se realizan operaciones aritméticas con él y su tamaño máximo es de cinco caracteres.
- *USUARIO_PPP*: usuario de validación contra el *Remote Authentication Dial-In User Service* o Servicio Remoto de Autenticación de Usuario (RADIUS) para tener acceso a la red. Este campo está formado por los dígitos del *SITEID* y una nomenclatura específica que identifica el tipo de línea, por lo que el tipo de dato seleccionado es *varchar*. El tamaño de esta composición nunca superará diez caracteres de longitud.
- *PASSWORD_PPP*: contraseña asociada a un usuario de RADIUS que sirve para autenticar y autorizar a este con el objetivo de tener acceso a la red. Está formado por una "P" seguida de los dígitos del campo *SITEID*. La composición nunca superará cuatro caracteres de longitud puesto que se limita el número máximo de sedes a 999, por lo que el tipo elegido es *char*.
- *DIRECCION_IP_LAN*: dirección LAN del equipo. Esta es la dirección que permitirá la comunicación entre el encaminador y todos los equipos conectados al mismo dentro de la red local.
- *MASCARA_IP_LAN*: máscara de subred que delimita qué parte de la dirección IP hace referencia a la red y qué parte hace referencia a los equipos que puede haber en dicha red.
- *PRIORIDAD*: en caso de que en una misma sede exista más de una línea, mediante este campo se establece la ruta preferida, siendo 1 la prioridad máxima. Se escoge el tipo numérico más pequeño, *tinyint*, sin signo y limitado a un dígito como máximo puesto que, en ningún caso, superará el valor 9.
- *EF_VALUE (Expedited Forwarding)*: clase de servicio designada para las aplicaciones de tráfico de voz sobre IP. Se le da tratamiento prioritario sobre otro tipo de tráfico. Se selecciona el tipo *mediumint* y se limita su longitud a seis caracteres puesto que el mayor número que podrá almacenar esta variable será 999999 Kbps en caso de configurar una línea de 1 Gbps.

- *AF4_VALUE (Assured Forwarding):* mediante el método AF se pueden ofrecer diferentes garantías de entrega. A cada clase, identificada por un número que va de 4 a 1, se le asigna un cierto ancho de banda y de *buffer*. En caso de que se produzca congestión, se irán descartando los paquetes de las clases menos preferentes, la cual viene establecida de la siguiente manera: AF4 > AF3 > AF2 >AF1. Aunque dentro de cada clase podrían establecerse tres subclases, en este caso no se aplica. AF4 es la clase de servicio que identifica el tráfico multimedia y señalización que tendrá un tratamiento priorizado en la red respecto al resto de clases de tipo AF.

  Se selecciona el tipo *mediumint* y se limita su longitud a seis caracteres puesto que el mayor número que podrá almacenar esta variable será 999999 Kbps en caso de configurar una línea de 1 Gbps.

- *AF3_VALUE*: clase de servicio más prioritaria que AF1 y AF2 pero menos que AF4. Se selecciona el tipo *mediumint* y se limita su longitud a seis caracteres puesto que el mayor número que podrá almacenar esta variable será 999999 Kbps en caso de configurar una línea de 1 Gbps.

- *AF2_VALUE*: clase de servicio más prioritaria que AF1 pero menos que AF4 y AF3. Se selecciona el tipo *mediumint* y se limita su longitud a seis caracteres puesto que el mayor número que podrá almacenar esta variable será 999999 Kbps en caso de configurar una línea de 1 Gbps.

- *AF1_VALUE*: clase de servicio menos prioritaria. Se selecciona el tipo *mediumint* y se limita su longitud a seis caracteres puesto que el mayor número que podrá almacenar esta variable será 999999 Kbps en caso de configurar una línea de 1 Gbps.

- *DE_VALUE*: clase de servicio que identifica el tráfico *best-effort.* Esta clase no asegura ninguna garantía ni prioridad de entrega. En caso de congestión, el tráfico marcado con esta clase será el primero en ser descartado. Se selecciona el tipo *mediumint* y se limita su longitud a seis caracteres puesto que el mayor número que podrá almacenar esta variable será 999999 Kbps en caso de configurar una línea de 1 Gbps.

  La suma de todas estas tiene que ser igual a la velocidad del puerto contratada.

La estructura de la tabla es la mostrada en la figura 38:

Columna	Tipo	Nulo	Enlaces a
CTID	smallint(3)	No	tareas_ct -> CTID
TELEFONO	varchar(15)	No	
ADM_ADSL	char(12)	No	
DIRECCION_GESTION	varchar(18)	No	
IP_FIJA_WAN	varchar(18)	No	
TIPO_CONEXION	char(5)	No	
TIPO_BACKUP	varchar(4)	No	
VPI	tinyint(2)	No	
VCI	tinyint(2)	No	
TECNOLOGIA	varchar(6)	No	
PROVEEDOR	tinyint(1)	No	operadoras -> OPERADORAID
QOSID	varchar(5)	No	
USUARIO_PPP	varchar(10)	No	
PASSWORD_PPP	char(4)	No	
DIRECCION_IP_LAN	varchar(18)	No	
MASCARA_IP_LAN	varchar(18)	No	
PRIORIDAD	tinyint(1)	No	
EF_VALUE	mediumint(6)	No	
AF4_VALUE	mediumint(6)	No	
AF3_VALUE	mediumint(6)	No	
AF2_VALUE	mediumint(6)	No	
AF1_VALUE	mediumint(6)	No	
DE_VALUE	mediumint(6)	No	

Figura 38. Estructura de la tabla 'ct_alta_generica'

Tabla 'ct_ivpn'

Esta tabla almacena información relacionada con las características técnicas del puerto contratado:

- *CTID*: identificador del componente técnico. Todos los componentes técnicos de este tipo comenzarán por cinco. Ejemplo: 5XX. Se ha definido como tipo *smallint* sin signo y se ha limitado a tres dígitos de longitud máxima. Para implantarlo en el entorno laboral sería necesario o aumentar la limitación de dígitos hasta cinco, siendo 501 el primero y 59999 el último o, en caso de necesitar aún más, cambiar el tipo de dato numérico.

  Actúa como clave primaria de la tabla de manera que se asegura que no puede haber dos componentes técnicos de este tipo con el mismo identificador. Además, también actúa como *FOREIGN KEY* o clave foránea puesto que se vincula con la clave primaria de la tabla *TAREAS_CT* de manera que no podrá existir de forma independiente sino que tendrá que estar asociado a una tarea. Esta relación se crea

de manera que en caso de borrar o actualizar uno de los registros de la tabla padre se realizará la misma operación en los registros asociados de la tabla hija.

- *USER_IVPN*: este campo hace referencia al identificador de usuario, el cual es único para cada cliente. El contenido de este campo tiene una longitud fija de diez caracteres por lo que se define como *char* y está formado por:
  - *COD3*: código de tres letras que identifica a cada uno de los clientes.
  - *Número de tres cifras*: número que se va asignando de manera secuencial a medida que crece el cliente.
  - *SITEID.*
- *TIPO_PUERTO*: este campo indica si el puerto es punto a punto o si por el contrario es un agregado. Se define como *varchar* cuya máxima longitud es de ocho caracteres.
- *ROUTING_CPE*: parámetro que indica el protocolo de encaminamiento utilizado en el equipo. De momento sólo se usa BGP, pero se decide utilizar un tipo *varchar* con longitud máxima de cinco caracteres por si en un futuro se empleasen protocolos como IGRP o EIGRP.
- *AS_NUMBER*: un *Autonomous System* o Sistema Autónomo (AS) identifica un conjunto de redes o de encaminadores que tienen una única política de encaminamiento. Cada AS se identifica mediante un número entero de 16 o 32 bits (16 bits en este proyecto). La comunicación entre los sistemas autónomos se realiza mediante el protocolo BGP. Este campo se define como *smallint* sin signo, que cubre el máximo indicado.
- *REFLECTOR*: este parámetro indica si el encaminador que se está configurando funciona como reflector o no. El encaminador reflector es aquel que, cuando uno de sus vecinos actualiza una ruta, reenvía dicha actualización al resto. Como los valores pueden ser “SI” o “NO”, el tipo de dato seleccionado es *char* de longitud 2.
- *LINE_VEL*: este parámetro contiene la velocidad de la línea contratada en Mbps. Como las líneas configuradas serán, como mucho, de 1 Gbps, el tipo de dato es *smallint* sin signo y se limita a cuatro el número de dígitos del mismo.
- *PUERTO_CONEXION*: este campo contiene el nombre del equipo y el puerto en el que se conectará la línea de cliente. Como el contenido de este parámetro puede variar, se define como *varchar* cuya longitud máxima es de treinta caracteres.
- *QOSID*: identificador que hace referencia a la distribución de los tráficos prioritarios. Aporta información acerca del tipo de línea, velocidad de línea, velocidad de puerto y distribución de las clases de servicio. El tipo de dato seleccionado es *varchar* puesto que no se realizan operaciones aritméticas con él y su tamaño máximo es de cinco caracteres.

- *NUMERO_SERVICIOS*: cada uno de los servicios de cliente configurados se identifican con un número el cual es almacenado en este campo. Se define como *smallint* sin signo de cuatro dígitos.
- *FRIENDLY_NAME*: nombre que se le asigna a la VPN del cliente en red. De esta manera, será más sencillo identificar dicha red mediante un nombre concreto que mediante un conjunto de números. Por si se añaden nuevos nombres, el tipo de este campo se define como *varchar* y su tamaño máximo es de veinte caracteres.
- *IP_WAN_VPLS*: dirección WAN del equipo. Es la dirección IP asignada por el ISP a través de la cual el encaminador es capaz de conectarse a Internet.

La estructura de la tabla es la mostrada en la figura 39:

Columna	Tipo	Nulo	Enlaces a
CTID	smallint(3)	No	tareas_ct -> CTID
USER_IVPN	char(10)	No	
TIPO_PUERTO	varchar(8)	No	
ROUTING_CPE	varchar(5)	No	
AS_NUMBER	smallint(5)	No	
REFLECTOR	char(2)	No	
LINE_VEL	smallint(4)	No	
PUERTO_CONEXION	varchar(30)	No	
QOSID	varchar(5)	No	
NUMERO_SERVICIOS	smallint(4)	No	
FRIENDLY_NAME	varchar(20)	No	
IP_WAN_VPLS	varchar(18)	No	

Figura 39. Estructura de la tabla ‘ct_ivpn’

Tabla ‘ct_linea’

Esta tabla almacena información relacionada con las características técnicas de la línea del cliente.

- *CTID*: identificador del componente técnico. Todos los componentes técnicos de este tipo comenzarán por seis. Ejemplo: 6XX. Se ha definido como tipo *smallint* sin signo y se ha limitado a tres dígitos de longitud máxima. Para implantarlo en el entorno laboral sería necesario o aumentar la limitación de dígitos hasta cinco, siendo 601 el primero y 69999 el último o, en caso de necesitar aún más, cambiar el tipo de dato numérico.

  Actúa como clave primaria de la tabla de manera que se asegura que no puede haber dos componentes técnicos de este tipo con el mismo identificador. Además, también actúa como *FOREIGN KEY* o clave foránea puesto que se vincula con la clave primaria de la tabla *TAREAS_CT* de manera que no podrá existir de forma independiente sino que tendrá que estar asociado a una tarea. Esta relación se crea

de manera que en caso de borrar o actualizar uno de los registros de la tabla padre se realizará la misma operación en los registros asociados de la tabla hija.

- *ADMINISTRATIVO*: número de doce dígitos que identifica la línea ofrecida por el proveedor de servicios. El tipo de dato seleccionado es *char* porque no se van a realizar operaciones aritméticas con el mismo y su longitud es fija.
- *TIPO_LINEA*: campo que indica el tipo de línea configurada. Como su tamaño máximo puede llegar a ser de trece caracteres, este campo se define como *varchar*.
- *VLAN_AGREGADO*: en caso de que la línea sea del tipo agregado, a través de un puerto físico se pueden configurar hasta 4094 *Virtual Local Area Network* o Red Virtual de Área Local (VLAN), cada una de las cuales se identifican con un número. Este parámetro hace referencia a dicho número, por lo que el tipo elegido es *smallint* sin signo de cuatro dígitos.
- *OPERADORA*: empresa a la que se alquila el circuito por el que se da servicio al cliente. Se escoge el tipo numérico más pequeño, *tinyint*, sin signo y limitado a un dígito como máximo puesto que, en ningún caso, superará el valor 9. Este parámetro se configura como *FOREIGN KEY* o clave foránea puesto que se vincula con la clave primaria de la tabla *OPERADORAID*. Esta relación se crea de manera que en caso de borrar o actualizar uno de los registros de la tabla padre se realizará la misma operación en los registros asociados de la tabla hija

La estructura de la tabla es la mostrada en la figura 40:

Columna	Tipo	Nulo	Enlaces a
CTID	smallint(3)	No	tareas_ct -> CTID
ADMINISTRATIVO	char(12)	No	
TIPO_LINEA	varchar(13)	No	
VLAN_AGREGADO	smallint(4)	No	
OPERADORA	tinyint(1)	No	operadoras -> OPERADORAID

Figura 40. Estructura de la tabla del 'ct_linea'

Tabla 'ct_qosid'

Esta tabla almacena información relacionada con las distintas calidades de servicio que la línea configurada debe proporcionar al cliente.

- *CTID*: identificador del componente técnico. Todos los componentes técnicos de este tipo comenzarán por ocho. Ejemplo: 8XX. Se ha definido como tipo *smallint* sin signo y se ha limitado a tres dígitos de longitud máxima. Para implantarlo en el entorno laboral sería necesario o aumentar la limitación de dígitos hasta cinco, siendo 801 el primero y 89999 el último o, en caso de necesitar aún más, cambiar el tipo de dato numérico. Actúa como clave primaria de la tabla de manera que se asegura que no puede haber dos componentes técnicos de este tipo con el mismo identificador.

Además, también actúa como *FOREIGN KEY* o clave foránea puesto que se vincula con la clave primaria de la tabla *TAREAS_CT* de manera que no podrá existir de forma independiente sino que tendrá que estar asociado a una tarea. Esta relación se crea de manera que en caso de borrar o actualizar uno de los registros de la tabla padre se realizarán la misma operación en los registros asociados de la tabla hija.

- *SERVICE_PIR*: este parámetro indica la velocidad del puerto contratado en Kbps, la cual debe ser menor o igual a la velocidad de la línea. Suponiendo que la velocidad máxima del puerto sea de 1 Gbps, el tipo de este campo se define como *mediumint* y se limita su longitud a siete caracteres.
- *SERVICE_EF_PCTJ*: porcentaje de la velocidad del puerto contratado destinado a la clase de servicio EF. Al ser un porcentaje cuyo máximo será 99%, el tipo de dato será *tinyint* sin signo y su tamaño será de dos dígitos. Este tipo de dato se repite para todos los valores que hacen referencia a un porcentaje.
- *SERVICE_AF4_PCTJ*: porcentaje de la velocidad del puerto contratado destinado a la clase de servicio AF4.
- *SERVICE_AF3_PCTJ*: porcentaje de la velocidad del puerto contratado destinado a la clase de servicio AF3.
- *SERVICE_AF2_PCTJ*: porcentaje de la velocidad del puerto contratado destinado a la clase de servicio AF2.
- *SERVICE_AF1: PCTJ*: porcentaje de la velocidad del puerto contratado destinado a la clase de servicio AF1.
- *SERVICE_DE_PCTJ*: porcentaje de la velocidad del puerto contratado destinado a la clase de servicio DE.
- *EF_SERVICE_PIR*: velocidad en Kbps reservada para la clase de servicio EF. Los números almacenados en este tipo de campos tendrán un tamaño máximo de seis dígitos, por lo que se definen como *mediumint* sin signo de longitud seis.
- *AF4_ SERVICE_PIR*: velocidad en Kbps reservada para la clase de servicio AF4.
- *AF3_ SERVICE_PIR*: velocidad en Kbps reservada para la clase de servicio AF3.
- *AF2_ SERVICE_PIR*: velocidad en Kbps reservada para la clase de servicio AF2.
- *AF1_ SERVICE_PIR*: velocidad en Kbps reservada para la clase de servicio AF1
- DE_ *SERVICE_PIR*: velocidad en Kbps reservada para la clase de servicio DE
- SERVICE_MGNT_PIR: velocidad en Kbps reservada para la gestión del equipo en líneas IVPN. De esta manera se asegura una pequeña porción del ancho de banda para que el técnico siempre pueda realizar operaciones en el encaminador. Este valor nunca será superior al 5% de la velocidad del puerto, por lo que el tamaño máximo del mismo será de cinco dígitos.

La suma del ancho de banda de cada una de las clases debe ser igual a la velocidad de puerto contratada.

La estructura de la tabla es la mostrada en la figura 41:

Columna	Tipo	Nulo	Enlaces a
CTID	smallint(3)	No	tareas_ct -> CTID
SERVICE_PIR	mediumint(7)	No	
SERVICE_EF_PCTJ	tinyint(2)	No	
SERVICE_AF4_PCTJ	tinyint(2)	No	
SERVICE_AF3_PCTJ	tinyint(2)	No	
SERVICE_AF2_PCTJ	tinyint(2)	No	
SERVICE_AF1_PCTJ	tinyint(2)	No	
SERVICE_DE_PCTJ	tinyint(2)	No	
EF_SERVICE_PIR	mediumint(6)	No	
AF4_SERVICE_PIR	mediumint(6)	No	
AF3_SERVICE_PIR	mediumint(6)	No	
AF2_SERVICE_PIR	mediumint(6)	No	
AF1_SERVICE_PIR	mediumint(6)	No	
DE_SERVICE_PIR	mediumint(6)	No	
SERVICE_MGNT_PIR	mediumint(5)	No	

Figura 41. Estructura de la tabla 'ct_qosid'

Tabla 'ct_router'

Esta tabla almacena información del *router* o encaminador instalado.

- *CTID*: identificador del componente técnico. Todos los componentes técnicos de este tipo comenzarán por dos. Ejemplo: 2XX. Se ha definido como tipo *smallint* sin signo y se ha limitado a tres dígitos de longitud máxima. Para implantarlo en el entorno laboral sería necesario o aumentar la limitación de dígitos hasta cinco, siendo 201 el primero y 29999 el último o, en caso de necesitar aún más, cambiar el tipo de dato numérico.

  Actúa como clave primaria de la tabla de manera que se asegura que no puede haber dos componentes técnicos de este tipo con el mismo identificador. Además, también actúa como *FOREIGN KEY* o clave foránea puesto que se vincula con la clave primaria de la tabla *TAREAS_CT* de manera que no podrá existir de forma independiente sino que tendrá que estar asociado a una tarea. Esta relación se crea de manera que en caso de borrar o actualizar uno de los registros de la tabla padre se realizarán la misma operación en los registros asociados de la tabla hija.

- *USERID*: este campo almacena el nombre que se le asigna al encaminador. En función del tipo de servicio configurado la sintaxis de este parámetro varía por lo que se define como *varchar* cuya longitud máxima es de quince caracteres. Su estructura puede ser:

  - Igual a la del parámetro *NOMBRE_USUARIO* de la tabla *ct_3g* para servicios ADSL o NEBA.

  - Prefijo R seguido del COD3 del cliente, un número de 3 cifras que aumenta de manera secuencial a medida que crece el cliente desde 001 hasta 999 y el valor del campo *SITEID* para líneas IVPN.

- *IP_GESTION*: dirección IP asociada a la interfaz de gestión del equipo. Es la dirección a través de la cual los técnicos pueden acceder al encaminador para realizar distintas operaciones.
- *NEXT_HOP*: dirección IP que hace referencia al siguiente encaminador al que se tendrán que redirigir los paquetes que vayan por la red para llegar a su destino.
- *MASCARA_GESTION*: campo que, junto a la dirección IP, sirve para determinar qué parte de esta se destina a la red y qué parte a las máquinas.
- *MARCA*: campo en el que se almacena la marca del encaminador de la línea correspondiente. Como los equipos utilizados son Cisco y Teldat, el tipo de dato es *varchar* cuya longitud máxima es de seis caracteres.
- *MODELO*: campo en el que se almacena el modelo concreto del encaminador. Este campo se define como *varchar* cuya longitud máxima es de quince caracteres.
- *PLANTILLA*: mediante este campo cada cliente podrá solicitar una plantilla específica de configuración. Su uso será real a partir de siguientes versiones de la herramienta. Se define como *varchar* cuya longitud máxima es de veinte caracteres.

La estructura de la tabla es la mostrada en la figura 42:

Columna	Tipo	Nulo	Enlaces a
CTID	smallint(3)	No	tareas_ct -> CTID
USERID	varchar(15)	No	
IP_GESTION	varchar(18)	No	
NEXT_HOP	varchar(18)	No	
MASCARA_GESTION	varchar(18)	No	
MARCA	varchar(6)	No	
MODELO	varchar(15)	No	
PLANTILLA	varchar(20)	No	

Figura 42. Estructura de la tabla 'ct_router'

Tabla 'ct_sap'

Esta tabla almacena información del *Service Access Point* o Punto de Acceso al Servicio (SAP):

- *CTID*: identificador del componente técnico. Todos los componentes técnicos de este tipo comenzarán por tres. Ejemplo: 7XX. Se ha definido como tipo *smallint* sin signo y se ha limitado a tres dígitos de longitud máxima. Para implantarlo en el entorno laboral sería necesario o aumentar la limitación de dígitos hasta cinco, siendo 701 el primero y 79999 el último o, en caso de necesitar aún más, cambiar el tipo de dato numérico.

Actúa como clave primaria de la tabla de manera que se asegura que no puede haber dos componentes técnicos de este tipo con el mismo identificador. Además, también actúa como *FOREIGN KEY* o clave foránea puesto que se vincula con la clave primaria de la tabla *TAREAS_CT* de manera que no podrá existir de forma independiente sino que tendrá que estar asociado a una tarea. Esta relación se crea de manera que en caso de borrar o actualizar uno de los registros de la tabla padre se realizarán la misma operación en los registros asociados de la tabla hija.

- *SERVICE_ID*: número que identifica el servicio configurado, cuyo tamaño puede variar no superando nunca las ocho cifras. El tipo *mediumint* no nos ofrece el tamaño suficiente porque su valor puede ser 99999999, de manera que el tipo de dato elegido es *int* sin signo.
- *SERVICE_VLAN*: número que identifica la VLAN por la que se proporciona servicio al cliente. Como su valor máximo puede ser 4094 se ha definido como *smallint* sin signo de cuatro cifras de tamaño máximo.
- *SERVICE_IP:* dirección IP asociada al servicio configurado.
- *SERVICE_QOS*: identificador que hace referencia a la distribución de los tráficos prioritarios del servicio contratado. El tipo de dato seleccionado es *varchar* puesto que no se realizan operaciones aritméticas con él y su tamaño máximo es de cinco caracteres.

La estructura de la tabla es la mostrada en la figura 43:

Columna	Tipo	Nulo	Enlaces a
CTID	smallint(3)	No	tareas_ct -> CTID
SERVICE_ID	int(8)	No	
SERVICE_VLAN	smallint(4)	No	
SERVICE_IP	varchar(18)	No	
SERVICE_QOS	varchar(5)	No	

Figura 43. Estructura de la tabla 'ct_sap'

Tabla 'operadoras'

Esta tabla establece la relación entre el código de una operadora y su nombre. Así, se evita que se referencie a la misma operadora de maneras distintas en registros diferentes y ofrece la posibilidad de realizar consultas en las que, por ejemplo, se busquen todas las líneas contratadas a una operadora:

- *OPERADORAID:* identificador numérico de la operadora. Se escoge el tipo numérico más pequeño, *tinyint*, sin signo y limitado a un dígito como máximo puesto que este valor, en ningún caso, superará el valor 9.

  Actúa como clave primaria de la tabla de manera que se asegura que no puede haber dos operadoras identificadas con el mismo número.

- *NOMBRE:* nombre de la operadora o proveedor. Este campo es de tipo *varchar* cuya longitud máxima de veinte caracteres.

La estructura de la tabla es la mostrada en la figura 44:

Columna	Tipo	Nulo
OPERADORAID	tinyint(1)	No
NOMBRE	varchar(20)	No

Figura 44. Estructura de la tabla 'operadoras'

Tabla 'proyectos'

En esta tabla se recoge información acerca de los proyectos que están en desarrollo.

- *PROID*: identificador numérico de cada proyecto. Todos los identificadores comenzaran por el número diez. Ejemplo: 10XX. Se ha definido como tipo *smallint* sin signo y se ha limitado a cuatro dígitos de longitud máxima. Para implantarlo en el entorno laboral sería necesario o aumentar la limitación de dígitos hasta cinco, siendo 1001 el primero y 10999 el último o, en caso de necesitar aún más, cambiar el tipo de dato numérico.

  Actúa como clave primaria de la tabla de manera que se asegura que no puede haber dos proyectos identificados con el mismo número. Cada proyecto puede estar compuesto por una o más tareas.

- *NOMBRE*: campo que almacena la información relativa al nombre que se le da al nuevo proyecto. Se define como *varchar* cuya longitud máxima es de cincuenta caracteres.

- *FECHA_INICIO*: fecha estimada de comienzo del trabajo a realizar. Se define como tipo *date.*

- *FECHA_FIN*: fecha estimada de finalización del proyecto. Se define como tipo *date.*

- *SITES*: códigos que identifican cada una de las localizaciones en las que hay instalada una línea. Estos códigos están compuestos por una *"S"* mayúscula acompañada de un número de tres cifras que crece secuencialmente a medida que se van añadiendo nuevas localizaciones a la base de datos, por lo se define como *char* de cuatro caracteres de tamaño fijo. En este caso identifican la localización en la que se va a llevar a cabo el proyecto. Este parámetro se configura como *FOREIGN KEY* o clave foránea puesto que se vincula con la clave primaria de la tabla *SITES*. Esto es así porque cada proyecto debe estar asociado a una sede en concreto. Esta relación se crea de manera que en caso de borrar o actualizar uno de los registros de la tabla padre se realizarán la misma operación en los registros asociados de la tabla hija.

La estructura de la tabla es la mostrada en la figura 45:

Columna	Tipo	Nulo	Enlaces a
PROID	smallint(4)	No	
NOMBRE	varchar(50)	No	
FECHA_INICIO	date	No	
FECHA_FIN	date	No	
SITES	char(4)	No	sites -> SITEID

Figura 45. Estructura de la tabla 'proyectos'

Tabla 'sites'

Esta tabla almacena información referente a la localización de la sede en la que se realiza una instalación.

- *SITEID*: códigos que identifican cada una de las localizaciones en las que hay instalada una línea. Estos códigos están compuestos por una *"S"* mayúscula acompañada de un número de tres cifras que crece secuencialmente a medida que se van añadiendo nuevas localizaciones a la base de datos, por lo se define como *char* de cuatro caracteres de tamaño fijo.

  Actúa como clave primaria por lo que no se podrán repetir dos códigos en la misma base de datos Para implementarlo en el entorno laboral sería necesario aumentar el tamaño tanto de este campo como el de todos los que contengan el código de localización de la sede ya que con cuatro caracteres sólo se pueden almacenar identificadores comprendidos entre 001 y 999.

- *PROVINCIA*: provincia donde se encuentra la sede. Se define como *varchar* cuya longitud máxima es de treinta caracteres.

- *MUNICIPIO*: municipio en el que se encuentra la sede. Se define como *varchar* cuya longitud máxima es de treinta caracteres.

- *DIRECCION*: localización de la sede. En este campo vendrá la calle, polígono industrial, puerta o cualquier dato relevante que identifique físicamente la situación de la sede. Por este motivo se define como *varchar* cuya longitud máxima es de cincuenta caracteres.

- *COD_POSTAL*: "*Relación de números formados por cifras que funcionan como clave de zonas, poblaciones y distritos, a efectos de la clasificación y distribución del correo*" [72]. Para este PFG sólo se han tenido en cuenta códigos postales de España, por lo que este número constará de cinco cifras. Como el código postal más alto empieza por 52, para Melilla, se ha definido este campo como *smallint* sin signo. En caso de tener sedes en el extranjero, habría que revisar el tamaño y contenido de este campo.

- *CLIENTE*: valor que identifica el código del cliente, por lo que, como ya se vio en la tabla *'clientes'*, se define como *smallint* sin signo. Este parámetro se configura como *FOREIGN KEY* o clave foránea puesto que se vincula con la clave primaria de la tabla *CLIENTES*.

  Esto es así porque una sede debe estar asociada a un determinado cliente. Esta relación se crea de manera que en caso de borrar o actualizar uno de los registros de la tabla padre se realizarán la misma operación en los registros asociados de la tabla hija.

La estructura de la tabla es la mostrada en la figura 46:

Columna	Tipo	Nulo	Enlaces a
SITEID	char(4)	No	
PROVINCIA	varchar(30)	No	
MUNICIPIO	varchar(30)	No	
DIRECCION	varchar(50)	No	
COD_POSTAL	smallint(5)	No	
CLIENTE	smallint(5)	No	clientes -> CODIGO_ORGANIZACION

Figura 46. Estructura de la tabla 'sites'

Tabla 'tareas'

En esta tabla se almacena toda la información relacionada con las tareas que definen los trabajos realizados en la empresa.

- *TAREAID*: identificador numérico de cada tarea. Todos los identificadores comenzaran por el número cien. Ejemplo: 100XX. Para este PFG el campo se define como *smallint* sin signo de cinco cifras. Para implantarlo en el entorno laboral sería necesario cambiar el tipo de dato numérico, ya que con lo definido sólo se pueden almacenar desde 10001 hasta 10099 tareas.

  Actúa como clave primaria de la tabla de manera que se asegura que no puede haber dos tareas identificadas con el mismo número.

- *PROID:* identificador numérico de cada proyecto. Una tarea sólo puede estar asociada a un proyecto. Al igual que en la tabla *'proyectos'*, se define como *smallint* sin signo limitado a cuatro cifras. Este parámetro se configura como *FOREIGN KEY* o clave foránea puesto que se vincula con la clave primaria de la tabla *PROYECTOS*. Esto es así porque una tarea sólo existe si está incluida dentro de un proyecto. Esta relación se crea de manera que en caso de borrar o actualizar uno de los registros de la tabla padre se realizarán la misma operación en los registros asociados de la tabla hija.

- *NOMBRE*: este campo almacena el nombre que se le ha asignado a la tarea. Se define como *varchar* cuya máxima longitud es veinte caracteres.

- *DURACION*: en este campo se anota la duración teórica de cada tarea, es decir, cuantos días se tienen que invertir en realizar la misma. Como una tarea nunca puede superar medio año, es decir, 180 días, este campo se define como *tinyint* sin signo de tres cifras, que permite almacenar números del 0 al 255.
- *SERVICIO*: tipo de servicio que debe configurar. Como el servicio a configurar cuyo nombre es mayor tiene cuatro caracteres, este campo se define como *varchar* de longitud máxima cuatro.
- *ELEMENTO*: tipo de equipo que se debe configurar. Por el momento, el único tipo de dispositivo configurado será un encaminador o *router*, por lo que este campo tiene longitud fija de seis caracteres y se define como *char*.

La estructura de la tabla es la mostrada en la figura 47:

Columna	Tipo	Nulo	Enlaces a
TAREAID	smallint(5)	No	
PROID	smallint(4)	No	proyectos -> PROID
NOMBRE	varchar(20)	No	
DURACION	tinyint(3)	No	
SERVICIO	varchar(4)	No	
ELEMENTO	char(6)	No	

Figura 47. Estructura de la tabla 'tareas'

Tabla 'tareas_ct'

El objetivo de esta tabla es establecer una relación entre la tabla *Tareas* y las tablas correspondientes a los distintos componentes técnicos involucrados en la línea instalada para facilitar la estructura de la base de datos y del código utilizado para realizar las consultas. Cada uno de los componentes técnicos de la base de datos es único. Sin embargo, una tarea puede tener asociados más de uno de estos componentes.

- *CTID*: identificador del componente técnico. Actúa como clave primaria puesto que no puede haber dos registros iguales. El tipo de dato utilizado es *smallint* sin signo limitado a un máximo de tres dígitos.
- *TAREAID*: identificador numérico de cada tarea. Todos los identificadores comenzaran por el número cien. Ejemplo: 100XX. El tipo de dato utilizado es *smallint* sin signo limitado a un máximo de cinco dígitos. Este parámetro se configura como *FOREIGN KEY* o clave foránea puesto que se vincula con la clave primaria de la tabla *TAREAS*. Gracias a esta tabla se pueden vincular la tabla *TAREAS* y las tablas de los distintos *CT* de una manera ordenada que permite organizar mejor la información. De esta manera, un componente técnico sólo puede existir si existe una tarea. Esta relación se crea de manera que en caso de borrar o actualizar uno de los registros de la tabla padre se realizarán la misma operación en los registros asociados de la tabla hija.

La estructura de la tabla es la mostrada en la figura 48:

Columna	Tipo	Nulo	Enlaces a
CTID	smallint(3)	No	
TAREAID	smallint(5)	No	tareas -> TAREAID

Figura 48. Estructura de la tabla 'tareas_ct'

Estructura

Finalmente en la figura 49 se muestra la estructura general de las tablas de la base de datos:

Figura 49. Estructura de la base de datos

6.2.2. Creación y definición de las tablas

A continuación se van a incluir fragmentos del código utilizado para la creación de las tablas de la base de datos. El objetivo es que el lector sepa algunos de los comandos básicos utilizados en lenguaje *MySQL* y conozca mejor la estructura interna de una tabla desde el punto de vista de la programación.

Pese a que son trece las tablas existentes en la base de datos, en este apartado sólo analizaremos las sentencias utilizadas para crear y definir una de ellas. El motivo por el que se decide esto es porque como el resto de las tablas se definen de manera similar se considera innecesario repetir el concepto.

La herramienta utilizada para la generación de las bases de datos, *phpMyAdmin*, crea las tablas con los datos que las componen y todos los cambios adicionales los realiza mediante el comando *ALTER TABLE*. Sin embargo, en la creación de la propia tabla se pueden incluir sentencias para indicar qué campos son claves primarias, si el valor de alguno de ellos es incremental, etc. En este PFG, para que el lector conozca las dos modalidades que puede utilizar a la hora de crear una base de datos, se van a mostrar ambos códigos.

El código utilizado para crear la tabla *proyectos* es el mostrado en la figura 50:

```
CREATE TABLE IF NOT EXISTS `proyectos` (
  `PROID` smallint(4) unsigned NOT NULL AUTO_INCREMENT,
  `NOMBRE` varchar(50) NOT NULL,
  `FECHA_INICIO` date NOT NULL,
  `FECHA_FIN` date NOT NULL,
  `SITES` char(4) NOT NULL
  PRIMARY KEY (`PROID`),
  KEY `SITES` (`SITES`),
  CONSTRAINT `fk_proyectos2sites` FOREIGN KEY (`SITES`) REFERENCES `sites` (`SITEID`)
    ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=1001 ;
```

Figura 50. Código para la creación de una tabla

La primera línea del código sirve para crear la tabla en caso de que otra con el mismo nombre no exista.

Las sentencias utilizadas a continuación son para definir qué atributos va a contener la tabla. En las líneas mencionadas, lo primero que se define es el nombre del atributo, que servirá para identificar la información almacenada. En segundo lugar se define el tipo de ese atributo así como la longitud del mismo. Por último se identifica si un valor puede o no ser del tipo *NULL*. Como se puede observar, el campo *PROID* también se define como autoincremental. Esto indica que, cada vez que se inserte un nuevo registro, el número identificador asignado al atributo será una unidad mayor que el asignado al registro anterior. Si no se especifica ningún número como inicio de secuencia el primer valor asignado por defecto será 1.

Una vez definidos los atributos de la tabla, se crean las claves que permitirán asegurar aspectos como que dos registros no estén repetidos o la integridad referencial de la tabla.

La *PRIMARY KEY* o clave primaria asegura que no se puedan repetir registros en la tabla, es decir, en este caso concreto dos proyectos nunca podrán tener el mismo número identificador.

La *KEY* o índice es necesario siempre que se utilicen claves foráneas. Esto es así para que la verificación de las claves sea más rápida. En este caso se crea para el atributo *SITES*, que será el que referencia al atributo *SITEID* de la tabla *sites*.

Por último se crea la *FOREIGN KEY* o clave foránea. Para ello, se define la restricción referenciando un campo de la tabla actual a otro campo indexado de otra tabla. Además, se indica las acciones a realizar en caso de que el campo de la tabla padre sufra una actualización o borrado. En este caso se indica que, si el campo *SITEID* de la tabla *sites* se borra o se actualiza, automáticamente se haga lo mismo en el campo *SITES* de la tabla *proyectos*.

Finalmente se define, por este orden, el motor de almacenamiento, que como se definió en el apartado 5.5 es InnoDB, el conjunto de caracteres permitidos y el valor por el que se iniciará la secuencia incremental.

La otra modalidad de creación de la tabla es la que ofrece la herramienta *phpMyAdmin*. En este caso, en primer lugar se crea la tabla sólo con los campos que la componen. Después se añaden las claves, los valores autoincrementales, etc. El código utilizado es el de la figura 51 y 52:

```
CREATE TABLE IF NOT EXISTS `proyectos` (
  `PROID` smallint(4) unsigned NOT NULL,
  `NOMBRE` varchar(50) NOT NULL,
  `FECHA_INICIO` date NOT NULL,
  `FECHA_FIN` date NOT NULL,
  `SITES` char(4) NOT NULL
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 ;
```

Figura 51. Código simple para la creación de una tabla

```
ALTER TABLE `proyectos`
    ADD PRIMARY KEY (`PROID`), ADD KEY `SITES` (`SITES`);

ALTER TABLE `proyectos`
    MODIFY `PROID` smallint(4) unsigned NOT NULL AUTO_INCREMENT,AUTO_INCREMENT=1001;

ALTER TABLE `proyectos`
    ADD CONSTRAINT `fk_proyectos2sites` FOREIGN KEY (`SITES`) REFERENCES `sites` (`SITEID`)
    ON DELETE CASCADE ON UPDATE CASCADE;
```

Figura 52. ALTER TABLE para la asignación de claves, modificación de campos, etc.

Como se puede observar, las operaciones sobre la tabla son las mismas que las de la figura 50, con la diferencia que en este caso se han hecho sobre una tabla ya existente. Para ello se utiliza el comando ALTER TABLE que, como su propio nombre indica, va a efectuar una alteración o modificación en la tabla existente.

Para el primer caso añadiría la clave primaria al atributo *PROID* y el índice al atributo *SITES*. En el segundo caso se modifica el atributo *PROID* indicando que los valores de este van a ser autoincrementales y que la secuencia comenzará por el valor 1001. En el tercer caso se añade la clave foránea al atributo *SITES*.

Una vez creada la tabla, lo siguiente que se va a analizar es la inclusión de los datos en la misma. Para ello se utiliza el código dela figura 53:

```
INSERT INTO `proyectos` (`NOMBRE`, `FECHA_INICIO`, `FECHA_FIN`, `SITES`) VALUES
('ALTA_ADSL_Concha Espina,1', '2011-11-10', '2011-11-23', 'S101'),
('ALTA_IVPN-AGRE_Concha Espina,3', '2014-04-11', '2014-04-24', 'S102'),
('ALTA_ADSL_Avenida Los Naranjos,93', '2010-02-17', '2010-03-02', 'S106'),
('ALTA_3G_Concha Espina,1', '2010-12-08', '2010-12-12', 'S101'),
('ALTA_NEBA_Av.Diagonal,10', '2014-05-23', '2014-06-06', 'S103'),
('ALTA_ADSL_Calle Alcala, 101', '2012-06-20', '2012-07-03', 'S105'),
('ALTA_IVPN-AGRE_Calle de la Concordia, 33', '2013-07-24', '2013-07-27', 'S104'),
('ALTA_ADSL_Calle Canfranc, 15', '2015-01-20', '2015-01-23', 'S107'),
('ALTA_NEBA_Rua San Pedro, 11', '2015-02-02', '2015-02-16', 'S108'),
('ALTA_3G_Calle Canfranc, 15', '2015-03-03', '2015-03-06', 'S107'),
('ALTA_IVPN_PTO_Calle Parras. 46', '2015-03-16', '2015-03-19', 'S109'),
('ALTA_IVPN_PTO_Carrera del Darro, 5', '2015-03-23', '2015-03-26', 'S110');
```

Figura 53. Comandos de inserción de datos en una tabla

En la primera línea se puede observar que la instrucción indica que se van a insertar datos en la tabla *proyectos*. Entre paréntesis se indica sobre qué campos se va a realizar la inserción de los datos y finalmente, con la instrucción *VALUES* se da paso a los datos que se van a incluir.

En las líneas sucesivas se insertan los datos deseados siguiendo el orden establecido en la línea 1. El atributo *PROID* se ha obviado, puesto que será un valor que comenzará en 1001 y se irá incrementando automáticamente cada vez que se añada un registro a la tabla. Sin embargo, se podría incluir este parámetro durante la inserción de datos de manera explícita sin afectar al proceso, como se observa en la figura 54:

```
INSERT INTO `proyectos` (`PROID`, `NOMBRE`, `FECHA_INICIO`, `FECHA_FIN`, `SITES`) VALUES
(1001, 'ALTA_ADSL_Concha Espina,1', '2011-11-10', '2011-11-23', 'S101'),
(1002, 'ALTA_IVPN-AGRE_Concha Espina,3', '2014-04-11', '2014-04-24', 'S102'),
(1003, 'ALTA_ADSL_Avenida Los Naranjos,93', '2010-02-17', '2010-03-02', 'S106'),
(1004, 'ALTA_3G_Concha Espina,1', '2010-12-08', '2010-12-12', 'S101'),
(1005, 'ALTA_NEBA_Av.Diagonal,10', '2014-05-23', '2014-06-06', 'S103'),
(1006, 'ALTA_ADSL_Calle Alcala, 101', '2012-06-20', '2012-07-03', 'S105'),
(1007, 'ALTA_IVPN-AGRE_Calle de la Concordia, 33', '2013-07-24', '2013-07-27', 'S104'),
(1008, 'ALTA_ADSL_Calle Canfranc, 15', '2015-01-20', '2015-01-23', 'S107'),
(1009, 'ALTA_NEBA_Rua San Pedro, 11', '2015-02-02', '2015-02-16', 'S108'),
(1010, 'ALTA_3G_Calle Canfranc, 15', '2015-03-03', '2015-03-06', 'S107'),
(1011, 'ALTA_IVPN_PTO_Calle Parras. 46', '2015-03-16', '2015-03-19', 'S109'),
(1012, 'ALTA_IVPN_PTO_Carrera del Darro, 5', '2015-03-23', '2015-03-26', 'S110');
```

Figura 54. Comandos de inserción de datos en una tabla indicando el valor del campo autoincremental

Además, aunque el siguiente valor del *PROID* fuera 1013 por el hecho de ser auto incremental, podría añadirse un registro cuyo identificador fuese, por ejemplo, 1080. Como el incremento se da sobre el mayor número almacenado, el siguiente proyecto registrado en el que no se indicara de manera explícita el valor del campo *PROID* tendría como identificador el 1081. La única posibilidad en la que la inserción puede fallar es si se trata de añadir un registro cuyo *PROID* ya esté almacenado en la tabla.

Como se mencionaba anteriormente, como el resto de tablas se crea de la misma manera, no se va a redundar en la información facilitada al lector.

6.3.Programa PHP

El programa que interactúa con la base de datos se ha desarrollado en PHP. Se necesitaba un lenguaje de programación que fuera capaz de ejecutarse en el servidor y que pudiera realizar consultas a una base de datos de manera dinámica y sencilla. A continuación se mostrarán fragmentos de código que permitirán entender el funcionamiento de los puntos más críticos del *script*.

Este programa recibe como parámetro el dato introducido por el usuario en la caja de texto de la página web, que será un identificador de tarea. Tras hacer la consulta en base a este número, los datos obtenidos se almacenan en una matriz, cuyo primer *array* contiene el nombre de los campos consultados y el segundo las parejas clave-valor que asocian el nombre del campo con su contenido. Esta matriz se devuelve al cliente en formato *JavaScript Object Notation* o Notación de Objetos de JavaScript (JSON). El objetivo de devolver los datos de la forma explicada anteriormente es ser fiel a la realidad.

Lo primero que hay que hacer es conectarse a la base de datos. Para ello es necesario saber la dirección del servidor donde se aloja esta base de datos, el usuario con el que se puede acceder, el cual debe haber sido creado previamente por el administrador de la misma, la contraseña utilizada por dicho usuario y el nombre de la propia base de datos. En caso de que la conexión se realice de forma satisfactoria, el *script* seguirá adelante realizando las consultas establecidas. En caso contrario, se le notificará al usuario el código del error así como el tipo del mismo. En la figura 55 se observa el código para la conexión:

```php
$servidor = "localhost"; //Dirección del servidor.En este caso localhost
$usuario = "Alberto"; //Usuario creado en la base de datos
$pass = "bbddpfg"; //Contraseña del usuario
$BD = "bbdd_proyecto"; //El nombre de la base de datos

//Conectamos con la base de datos
$conexion = new mysqli($servidor, $usuario, $pass, $BD);
if ($conexion->connect_error) {
    switch($conexion->connect_errno){
        case 2002:
            $tipoError = "Direccion de servidor desconocida.";
        break;
        case 1045:
            $tipoError = "Error de autenticacion. Revise el user o la password.";
        break;
        case 1049:
            $tipoError = "Base de datos desconocida.";
        break;
        default:
            $tipoError = "Error desconocido.";
        break;
    }
    die('Error de conexion: (' . $conexion->connect_errno . ') <br>Descripcion: ' . $tipoError);
}
```

Figura 55. Conexión a la base de datos

Una vez que se ha establecido la conexión, se extrae el valor que se ha pasado desde el cliente y se realizan una serie de comprobaciones para evitar posibles ataques de inyección de código SQL. Este tipo de ataque se define, según la propia web de PHP, como "*técnica donde un atacante crea o altera comandos SQL existentes para exponer datos ocultos, sobrescribir los valiosos, o peor aún, ejecutar comandos peligrosos a nivel de sistema en el equipo que*

hospeda la base de datos. Esto se logra a través de la práctica de tomar la entrada del usuario y combinarla con parámetros estáticos para elaborar una consulta SQL" [73]. Para evitarlo, se han realizado las siguientes acciones: escapar las cadenas introducidas en la caja de texto de búsqueda y comprobar si el tipo de dato leído es el esperado. Además, destacar que el método utilizado para realizar las consultas no permite que se concatenen consultas, por lo que evita, en parte, el problema.

Para escapar la cadena, el código utilizado es el de la figura 56:

```
$numeroTarea = $_GET["numeroTarea"];
//Como medida de seguridad para evitar ataques, se escapa la cadena obtenida
$numeroTarea = mysqli_real_escape_string($conexion, $numeroTarea);
```

Figura 56. Escapar la cadena leída

De esta manera, si se añaden caracteres como las comillas simples o dobles, se añadirá el símbolo "\" y en la siguiente comprobación, ilustrada en la figura 57, fallará.

```
if(ctype_digit($numeroTarea)){
else{
    echo "El valor introducido contiene caracteres no numericos. Reviselo, gracias.";
}
```

Figura 57. Comprobación del tipo de dato leído

En caso de que el tipo de dato no sea un entero no se ejecutará más código y se devolverá una cadena de error al cliente. De esta manera se asegura que no se incluya ningún carácter como, por ejemplo, el punto y coma.

Tras este código para evitar los ataques por inyección de código, se comprueba, en primer lugar, si existe la tarea y, en segundo lugar, si esta es una tarea de configuración o por el contrario es otro tipo de tarea. Una vez que se determina que la tarea existe y que se trata de un alta de servicios, se realiza la primera consulta. Desde este momento se toma como ejemplo la tarea 10002, que se corresponde con la instalación de un servicio ADSL. En la consulta mostrada a continuación se pregunta por los datos genéricos del cliente, es decir, el tipo de servicio, el equipo instalado, el número identificador del cliente, el nombre del cliente, el código asignado a la localización donde está montado el servicio y el municipio al que pertenece. En caso de que el identificador de tarea introducido por el usuario no exista o bien la tarea no sea un alta de servicios, se le notificará al usuario la causa por la que no se ha hecho la consulta. El código utilizado para realizar la consulta es el de la figura 58:

```
//Se comprueba si la tarea buscada existe y, si existe, se trata de una baja o no
$consultaEsAlta = "SELECT tareas.NOMBRE FROM tareas WHERE tareas.TAREAID = '".$numeroTarea."'";
$resConsultaEsAlta = mysqli_query($conexion, $consultaEsAlta);
$esAlta = mysqli_fetch_assoc($resConsultaEsAlta)["NOMBRE"];
if(strlen($esAlta) > 0){
    if(strstr($esAlta, "ALTA")){
        $consultaDatosGenericosCliente =
            "SELECT tareas.SERVICIO, tareas.ELEMENTO, clientes.CODIGO_ORGANIZACION,
            clientes.CLIENTE, sites.SITEID, sites.MUNICIPIO
            FROM tareas INNER JOIN proyectos ON tareas.PROID = proyectos.PROID
            INNER JOIN sites ON proyectos.SITES = sites.SITEID
            INNER JOIN clientes ON sites.CLIENTE = clientes.CODIGO_ORGANIZACION
            WHERE tareas.TAREAID = '".$numeroTarea."'";
```

Figura 58. Consulta de los datos genéricos del cliente

El resultado de la consulta es un objeto *mysqli_result*, el cual contiene los registros devueltos por la base de datos. Para poder operar con dichos registros es necesario transformarlos en *arrays*, lo que se consigue utilizando la función *mysqli_fetch_array*, que devuelve un *array* por cada uno de los registros o filas que ha devuelto la consulta. Este *array* tendrá tantas posiciones como campos tiene el registro y, al añadirle la constante *MYSQLI_ASSOC*, se comportará como un *array* asociativo de manera que en cada una de ellas se almacena una pareja clave-valor. La clave se corresponde con el nombre del atributo mientras que el valor hace referencia al contenido del campo en cuestión.

El resultado obtenido de realizar la consulta es el mostrado en la figura 59:

SERVICIO	ELEMENTO	CODIGO_ORGANIZACION	CLIENTE	SITEID	MUNICIPIO
ADSL	ROUTER	1	BT	S101	MADRID

Figura 59. Registro obtenido tras realizar la consulta de los datos genéricos del cliente

El objeto obtenido se almacena en la variable *$arrayDatos*. En cada una de las posiciones de este *array* se guardarán los *arrays* asociativos obtenidos tras cada consulta realizada a la base de datos. Por tanto, contendrá los datos necesarios para generar el resultado final. El código que realiza esto es el de la figura 60:

```
$arrayDatos = array();
while($cadaRegistro = mysqli_fetch_array($datosConsulta, MYSQLI_ASSOC)) {
    $arrayDatos[$i] = $cadaRegistro;
    $i++;
}
```

Figura 60. Creamos el array de datos con cada uno de los registros de la consulta

Como se puede observar en la figura 59, esta consulta sólo ha generado un registro, por lo que en *$arrayDatos* sólo se guardará un *array*. Este contiene en cada una de sus posiciones las parejas clave-valor correspondientes al nombre del atributo y al contenido del campo, es decir, la primera posición del *array* contiene *SERVICIO>ADSL*, la segunda *ELEMENTO>ROUTER* y así sucesivamente.

Una vez que se han conseguido los datos genéricos del cliente, se debe obtener el resto de la información para configurar el encaminador. Para ello es necesario consultar todos los componentes técnicos asociados al tipo de servicio que se va a configurar. Estos componentes no siempre serán los mismos puesto que, por ejemplo, una línea ADSL no tiene asociados los mismos componentes que una línea 3G. El código para realizar dicha consulta es el mostrado en la figura 61:

```
$consultaCTs = "SELECT tareas_ct.CTID
FROM tareas INNER JOIN tareas_ct ON tareas.TAREAID = tareas_ct.TAREAID
    WHERE tareas.TAREAID = ".$numeroTarea;
$datosConsultaCTs = mysqli_query($conexion, $consultaCTs);
$numFilas = mysqli_num_rows($datosConsultaCTs);
if ($numFilas > 0) {
else {
    echo "La tarea que usted indica no existe. Disculpe mas molestias.";
}
```

Figura 61. Consulta de los CT de cada línea

Gracias a la tabla *tareas_ct* definida en el apartado 6.2.1, se evita realizar una consulta a cada una de las tablas de componentes para obtener aquellos que están asociados a la tarea que se está analizando.

El resultado que se obtiene de la consulta se observa en la figura 62:

CTID
201
401

Figura 62. Resultado de la consulta de los componentes técnicos

La consulta muestra que la línea tiene asociados dos componentes técnicos, el 201 y el 401. Aunque ya se conoce que hay que obtener los datos de dos tablas diferentes, no se sabe a qué dos tablas referencian cada uno de los códigos identificadores obtenidos. Por tanto, el programa debe de analizar la información y determinar qué tabla es la que hay que consultar.

Para ello, se analizará el primer dígito del identificador obtenido en la consulta anterior. Tal y como se estudió en el apartado 6.2.1, las tablas que hacen referencia a los distintos componentes técnicos se identifican por un número que, en función del tipo de componente, comienza por una cifra u otra. De esta manera, el identificador 201 hará referencia a la tabla *ct_router* mientras que el identificador 401 hará referencia a la tabla *ct_alta_generica.* Para conseguir identificar el nombre de las tablas por las que se preguntará a la base de datos se realizan las operaciones mostradas en la figura 63:

```
while($rowCT = mysqli_fetch_array($datosConsultaCTs, MYSQLI_NUM)) {
    $cadenaNombreColumnas = "";
    $tipoCT = substr($rowCT[0], 0, 1);//Buscamos el tipo de CT que vamos a tener que sacar
    switch($tipoCT){
        case "2"://CT_Router
            $consultaNombreColumnas = "SHOW COLUMNS FROM ct_router";
            $nombreTabla = "ct_router";
        break;
        case "3"://CT_3G
            $consultaNombreColumnas = "SHOW COLUMNS FROM ct_3g";
            $nombreTabla = "ct_3g";
        break;
        case "4"://CT_Alta_Generica
            $consultaNombreColumnas = "SHOW COLUMNS FROM ct_alta_generica";
            $nombreTabla = "ct_alta_generica";
        break;
        case "5"://CT_iVPN
            $consultaNombreColumnas = "SHOW COLUMNS FROM ct_ivpn";
            $nombreTabla = "ct_ivpn";
        break;
        case "6"://CT_Linea
            $consultaNombreColumnas = "SHOW COLUMNS FROM ct_linea";
            $nombreTabla = "ct_linea";
        break;
        case "7"://CT_SAP
            $consultaNombreColumnas = "SHOW COLUMNS FROM ct_sap";
            $nombreTabla = "ct_sap";
        break;
        case "8"://CT_QOSID
            $consultaNombreColumnas = "SHOW COLUMNS FROM ct_qosid";
            $nombreTabla = "ct_qosid";
        break;
```

Figura 63. Búsqueda del tipo de componente técnico

La variable *$nombreTabla* es en la que se almacenará a qué tabla hay que realizar la consulta. Ahora, si el objetivo fuese obtener los datos almacenados en dicha tabla para un *CTID* concreto, bastaría con preguntarle a la base de datos por todos los registros cuyo elemento identificador fuese igual al *CTID* analizado. Sin embargo, este tipo de consulta sería errónea puesto que en el resultado devuelto el identificador del componente es innecesario además de que, en algunos tipos de servicio, no se deben obtener todos los atributos de la tabla. Es decir, no sólo se necesita conocer el nombre de la tabla que hay que consultar sino que además es necesario saber el nombre específico de los elementos por los que se va a preguntar. Por este motivo se va a generar la consulta almacenada en la variable *$consultaNombreColumnas*, cuyo resultado para el *ct_router (201)*, que se puede ver en la figura 64, permite conocer el nombre de cada uno de los campos de la tabla.

Field	Type	Null	Key	Default	Extra
CTID	smallint(3) unsigned	NO	PRI	*NULL*	auto_increment
USERID	varchar(15)	NO		*NULL*	
IP_GESTION	varchar(18)	NO		*NULL*	
NEXT_HOP	varchar(18)	NO		*NULL*	
MASCARA_GESTION	varchar(18)	NO		*NULL*	
MARCA	varchar(6)	NO		*NULL*	
MODELO	varchar(15)	NO		*NULL*	
PLANTILLA	varchar(20)	NO		*NULL*	

Figura 64. Resultado de la consulta de las columnas de la tabla para el ct_router

Una vez obtenida la información anterior, se debe de comprobar qué campos hay que incluir en el resultado y qué campos deben ser omitidos. Para solventar este problema se ha implementado un *array* que funciona a modo de lista almacenando todos los nombres de los elementos que deben ser excluidos en función del tipo de servicio a configurar. En la figura 65 se puede observar que, de momento, sólo se deben ignorar datos en función de si el servicio es NEBA o no. En caso de que en futuras evoluciones de la herramienta se quiera prescindir de otros campos en servicios ADSL, 3G o IVPN, se deben incluir los arrays comentados en el código, cuyo contenido será el nombre de los campos a omitir, y eliminar la variable *$camposOmitidosDefecto* que ahora determina el elemento que hay que obviar en todos los casos excepto en NEBA.

```
$camposOmitidosNeba = array("ct_alta_generica.EF_VALUE", "ct_alta_generica.AF4_VALUE",
    "ct_alta_generica.AF3_VALUE", "ct_alta_generica.AF2_VALUE", "ct_alta_generica.AF1_VALUE",
    "ct_alta_generica.DE_VALUE", "ct_qosid.SERVICE_MGNT_PIR");
$camposOmitidosDefecto = array("ct_alta_generica.IP_FIJA_WAN");
//$camposOmitidosAdsl = array("ct_alta_generica.IP_FIJA_WAN");
//$camposOmitidosTresG = array("ct_alta_generica.IP_FIJA_WAN");
//$camposOmitidosIvpn = array("ct_alta_generica.IP_FIJA_WAN");
```

Figura 65. Listas con los campos a ignorar por servicio instalado

Como se puede observar en el código de la figura 66, tras comprobar que la consulta ha arrojado al menos una fila de datos, se vuelcan los registros de dicha consulta a un *array* numérico. Tras este paso y exceptuando el primer registro que hace referencia al *CTID*, se recorren todos los elementos del *array* numérico obteniendo el valor de la primera posición de cada uno de ellos, que se corresponde con el nombre del campo. Finalmente, se comprueba si dicho campo se encuentra o no en la lista de omitidos y, en caso negativo, se añade a la variable *$cadenaNombreColumnas:*

```
$datosNombreColumnas = mysqli_query($conexion, $consultaNombreColumnas);
$numFilas = mysqli_num_rows($datosNombreColumnas);
$rowsNombreColumnas = mysqli_fetch_all($datosNombreColumnas, MYSQLI_NUM);
if ($numFilas > 0) {
    for($j=1; $j<$numFilas; $j++){
        $nombreCampo = $rowsNombreColumnas[$j][0];
        //En función al tipo de servicio comprobaremos un listado u otro de campos omitidos
        switch($tipoServicio){
            case "NEBA":
                if (!in_array($nombreTabla.".".$nombreCampo, $camposOmitidosNeba)){
                    $cadenaNombreColumnas .= $nombreTabla.".".$nombreCampo.", ";
                }
                break;
            default:
                if (!in_array($nombreTabla.".".$nombreCampo, $camposOmitidosDefecto)){
                    $cadenaNombreColumnas .= $nombreTabla.".".$nombreCampo.", ";
                }
                break;
        }
    }
    //Eliminamos el ", " del final de la cadena para quedarnos con la parte de la consulta
    //SQL bien formada.
    $cadenaNombreColumnas = substr($cadenaNombreColumnas, 0, -2);
}
else{
    echo "Fallo en la consulta de los nombres de columna del CT";
}
```

Figura 66. Comprobación de los campos a consultar

La cadena generada por el código de la figura 66 es la mostrada en la figura 67:

```
ct_router.USERID, ct_router.IP_GESTION, ct_router.NEXT_HOP,
ct_router.MASCARA_GESTION, ct_router.MARCA, ct_router.MODELO,
ct_router.PLANTILLAct_alta_generica.TELEFONO, ct_alta_generica.ADM_ADSL,
ct_alta_generica.DIRECCION_GESTION, ct_alta_generica.TIPO_CONEXION,
ct_alta_generica.TIPO_BACKUP, ct_alta_generica.VPI,
ct_alta_generica.VCI, ct_alta_generica.TECNOLOGIA,
ct_alta_generica.PROVEEDOR, ct_alta_generica.QOSID,
ct_alta_generica.USUARIO_PPP, ct_alta_generica.PASSWORD_PPP,
ct_alta_generica.DIRECCION_IP_LAN, ct_alta_generica.MASCARA_IP_LAN,
ct_alta_generica.PRIORIDAD, ct_alta_generica.EF_VALUE,
ct_alta_generica.AF4_VALUE, ct_alta_generica.AF3_VALUE,
ct_alta_generica.AF2_VALUE, ct_alta_generica.AF1_VALUE,
ct_alta_generica.DE_VALUE
```

Figura 67. Cadena que contiene los campos por los que se va a preguntar a la base de datos

Una vez que se conocen los atributos por los que se debe preguntar, se genera la consulta de la figura 68:

```
$consultaDatosCT ="SELECT ".$cadenaNombreColumnas."
                    FROM tareas INNER JOIN tareas_ct ON tareas.TAREAID = tareas_ct.TAREAID
                    INNER JOIN ".$nombreTabla." ON tareas_ct.CTID = ".$nombreTabla.".CTID
                    WHERE tareas.TAREAID = ".$numeroTarea;
```

Figura 68. Código de la consulta de los atributos del componente técnico

Los datos obtenidos tras realizar la consulta, tal y como se hizo con los datos genéricos del cliente, son añadidos a la variable *$arrayDatos*, como se puede observar en la figura 69:

```
$datosConsulta = mysqli_query($conexion, $consultaDatosCT);
while($cadaRegistro = mysqli_fetch_array($datosConsulta, MYSQLI_ASSOC)) {
    $arrayDatos[$i] = $cadaRegistro;
    $i++;
}
```

Figura 69. Código que añade el resultado de la consulta al array de datos

Estas operaciones se repiten para cada *CTID* obtenido en la consulta de la figura 62 por lo que la variable *$arrayDatos* contendrá toda la información necesaria para la configuración. Sin embargo, para poder simular el resultado devuelto en el entorno laboral, es necesario realizar un último tratamiento de la información. En este punto del *script*, la variable *$arrayDatos* contiene tres *arrays:*

- El primero contiene el *array* asociativo de la información obtenida en la consulta de los datos genéricos del cliente.
- El segundo contiene el *array* asociativo de la información obtenida de la consulta del componente técnico router (201).
- El tercero contiene el *array* asociativo de la información obtenida de la consulta del componente técnico alta genérica (401).

En este paso se unifican estos tres *arrays* en uno solo. Para ello se realiza lo mostrado en la figura 70:

```
$arrayNombreCols = array();
$datosAtributos = array();
foreach($arrayDatos as $arrayD)
{
    foreach($arrayD as $nombre=>$valor)
    {
        $arrayNombreCols[] = $nombre;
        if($nombre == "PROVEEDOR" || $nombre == "OPERADORA"){
            $consultaOperadora = "SELECT operadoras.NOMBRE
                FROM operadoras WHERE operadoras.OPERADORAID = '".$valor."'";
            $resConsultaOperadora = mysqli_query($conexion, $consultaOperadora);
            $operadora = mysqli_fetch_assoc($resConsultaOperadora)["NOMBRE"];
            $datosAtributos[$nombre] = $operadora;
        }
        else
            $datosAtributos[$nombre] = $valor;
    }
}
```

Figura 70. Creación de los arrays que serán devueltos al cliente

Por un lado se crea un *array* que contiene el nombre de los campos consultados en la base de datos, *$arrayNombreCols*. Por otro, se genera un *array* asociativo que almacena las parejas compuestas por el nombre del campo y su valor, *$datosAtributos*.

Para cada elemento del *$arrayDatos* se analiza su clave y su valor de tal manera que su clave se almacena en *$arrayNombreCols* mientras que en *$datosAtributos* se unifican todos los elementos obtenidos a lo largo de las distintas consultas realizadas.

Además, los elementos cuyo nombre sea *PROVEEDOR* u *OPERADORA* tienen un tratamiento especial. El motivo es que se ha creado una tabla de operadoras para evitar posibles referencias a la misma con distintos nombres, es decir, que no se guarden dos registros en los cuales la operadora sea "Movistar" y Movistar España". Además, gracias a esta tabla, que relaciona una operadora con un código identificativo, existe la posibilidad de, por ejemplo, realizar consultas de líneas por operadoras, lo que puede ser interesante en un futuro. Por lo tanto, el código de la figura 70 también se encarga de obtener el nombre de la operadora para que en el resultado veamos este y no un número que no aporte información.

Finalmente, el resultado devuelto en formato JSON al cliente que realizó la petición es un *array* asociativo cuyas claves son "columnas" y "datos" y sus valores los *arrays* creados mediante el código de la figura 71:

```
$resultado = array();
$resultado["columnas"] = $arrayNombreCols;
$resultado["datos"] = $datosAtributos;
echo json_encode($resultado);
```

Figura 71. Código que devuelve el resultado al usuario en formato JSON

Lo que se recibe en el lado del cliente se recoge en la figura 72:


```
"columnas":
    [
        "SERVICIO","ELEMENTO","CODIGO_ORGANIZACION","CLIENTE","SITEID","MUNICIPIO","USERID","IP_GESTION",
        "NEXT_HOP","MASCARA_GESTION","MARCA","MODELO","PLANTILLA","TELEFONO","ADM_ADSL","DIRECCION_GESTION",
        "TIPO_CONEXION","TIPO_BACKUP","VPI","VCI","TECNOLOGIA","PROVEEDOR","QOSID","USUARIO_PPP","PASSWORD_PPP",
        "DIRECCION_IP_LAN","MASCARA_IP_LAN","PRIORIDAD","EF_VALUE","AF4_VALUE","AF3_VALUE","AF2_VALUE",
        "AF1_VALUE","DE_VALUE"
    ],
"datos":
    {
        "SERVICIO":"ADSL","ELEMENTO":"ROUTER","CODIGO_ORGANIZACION":"1","CLIENTE":"BT","SITEID":"S101",
        "MUNICIPIO":"MADRID","USERID":"BTTTEL_ATM_S101","IP_GESTION":"10.4.113.194","NEXT_HOP":"10.4.113.199",
        "MASCARA_GESTION":"255.255.255.0","MARCA":"CISCO","MODELO":"CISCO1941\/K9","PLANTILLA":"",
        "TELEFONO":"915689897","ADM_ADSL":"214748364712","DIRECCION_GESTION":"10.4.137.196",
        "TIPO_CONEXION":"PPPoE","TIPO_BACKUP":"MAIN","VPI":"0","VCI":"35","TECNOLOGIA":"ADSL2+",
        "PROVEEDOR":"TELEFONICA","QOSID":"10678","USUARIO_PPP":"101_A_3P","PASSWORD_PPP":"P101",
        "DIRECCION_IP_LAN":"10.228.21.0","MASCARA_IP_LAN":"255.255.255.0","PRIORIDAD":"1","EF_VALUE":"0",
        "AF4_VALUE":"500","AF3_VALUE":"0","AF2_VALUE":"0","AF1_VALUE":"250","DE_VALUE":"0"
    }
```


Figura 72. Resultado devuelto al usuario

Este resultado se tratará en el lado del cliente mediante un programa desarrollado en lenguaje JavaScript que se describe en el apartado siguiente.

6.4.Herramienta Web: HTML, JavaScript y CSS

La herramienta web es la interfaz de usuario a través de la cual este puede realizar consultas a la base de datos, generar los ficheros CSV que contienen dichos datos o generar el fichero de configuración. Esta herramienta está constituida por tres archivos que serán descritos a continuación. En primer lugar se mostrará el fichero HTML que contiene la estructura básica de la página web, en segundo lugar se explicará la hoja de estilos diseñada para dar formato a dicha web y, finalmente, se estudiará el fichero desarrollado en lenguaje JavaScript con el que se han realizado funciones para la interacción con el usuario.

HerramientaWeb.html

En este fichero se determina el aspecto visual que tendrá la página web. La estructura está dividida en dos bloques bien diferenciados. El primer bloque, denominado *capaSuperior*, es el que contiene la imagen corporativa y el título de la herramienta, mientras que el segundo, *contenidoPrincipal*, es en el que se definen los elementos necesarios para la búsqueda de datos y se presenta el resultado. El código utilizado para generar la página es el de la figura 73:

```
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Generador Automatico de Ficheros de Configuracion</title>
    <link rel="stylesheet" type="text/css" href="./HojaEstilos.css"/>
    <script type="text/javascript" src="./FuncionesWeb.js"></script>
</head>

<body>
    <div class="contenedorTodo">
        <div class="capaSuperior">
            <img alt="" class="logoBt" src="imagenes/LogoBT.png" />
            <p>Generador Automatico de Ficheros de Configuracion</p>
        </div>
        <div id="contenidoPrincipal" class="contenidoPrincipal">
            <div class="busqueda">
                <input type="text" id="textBox" name="textBox"/>
                <input type="submit" value="Buscar" onclick="javascript:consultar(textBox.value)"/>
            </div>
            <div id="resultado"></div>
        </div>
    </div>
</body>
```

Figura 73. HTML: código HTML de la herramienta web

Como se puede observar en la figura, este fichero está enlazado tanto con la hoja de estilos que se encarga del formato de la web, *HojaEstilos.css*, como con el fichero desarrollado en JavaScript, *FuncionesWeb.js*.

El motivo por el que se han definido los atributos *class* e *id* es porque ambos son utilizados por los ficheros enlazados. Los primeros se utilizan en la hoja de estilos para dar formato a elementos concretos de la web mientras que los segundos se utilizan en JavaScript para obtener la referencia de aquellos objetos con los que se desea interactuar.

HojaEstilos.css

El formato de la página web que va a ver el usuario se define en este documento. Concretamente, los selectores utilizados para definir el estilo son:

- *contenedorTodo*: se definen tanto el tamaño por defecto como el tamaño mínimo del elemento que contiene a todos los demás. El tamaño por defecto es el 100% de la pantalla del usuario mientras que, en caso de minimizar la ventana, se define un tamaño mínimo para evitar que los elementos se descoloquen y sea imposible ver el contenido.
- *capaSuperior*: se define el tamaño, borde y color del encabezado de la página en el que se encuentra la imagen y el título.
- *capaSuperior p*: se define el formato del texto situado en el encabezado de la página: tamaño, posición, tipo de letra y color.
- *img.logoBT*: se fija el tamaño de la imagen del logo de la empresa y su posición.
- *contenidoPrincipal*: se fija tanto el tamaño por defecto como el mínimo que tendrá el elemento que contiene la caja de texto, los botones y la tabla generada.
- *búsqueda*: se fija la posición del elemento o sección que contiene la caja de texto y el botón *Buscar*.
- *nombres*: se define la posición, tamaño y la fuente del texto de las celdas que contienen los nombres de los campos.
- *datos*: se define la posición, tamaño, tipo de fuente y color del texto de las cajas contenidas en las celdas de datos.
- *table*: se define la posición, el tipo de borde y el tamaño de la tabla presentada al usuario.
- *botones*: se define el posicionamiento de los botones de acción *GenerarCSV* y *Generar fichero de configuración*.

FuncionesWeb.js

Este fichero contiene el código JavaScript desarrollado que trata los datos devueltos tras realizar una petición al servidor además de ofrecer al usuario la posibilidad de interactuar con la propia web. A continuación, a través de los métodos definidos, se detalla la funcionalidad del código generado. Los métodos se presentan en orden de ejecución:

- *consultar*: esta es la función que se ejecuta cuando el usuario decide realizar la búsqueda de los datos relacionados con un identificador de tarea. Lo primero que hace esta función es dejar limpia la pantalla de posibles búsquedas anteriores. Esto es así porque, cuando un usuario realiza una búsqueda, se muestra una tabla resultado además de dos botones de acción. Por eso, cuando se realiza una nueva búsqueda, es necesario que la pantalla mostrada al usuario sólo tenga los datos nuevos. Para ello, si se detecta que en el contenido de la web existe un elemento del tipo *table*, es decir, que ya se ha hecho alguna búsqueda con anterioridad, se borran todos los elementos desde este. El código se puede observar en la figura 74:

```
for(i=0; i<cP.childNodes.length; i++){
    if(cP.childNodes[i].nodeName == "TABLE"){
        while(cP.childNodes[i] !== undefined){
            cP.removeChild(cP.childNodes[i]);
        }
    }
}
```

Figura 74. JavaScript: eliminar elementos de búsquedas anteriores

Tras esto, se hace la petición al servidor. Para ello, se crea el objeto que permitirá la interacción con el mismo y que se explica en el punto siguiente. Una vez obtenido este objeto, se especifica el tipo de petición, GET en este caso, indicando la dirección *Uniform Resource Locator* o Localizador Uniforme de Recursos (URL) donde se aloja el fichero PHP y pasándole como parámetro el valor del identificador de tarea que el usuario desea buscar. Lo siguiente es indicar qué función queremos que se ejecute cuando se obtiene la respuesta del servidor y, finalmente, enviar la petición y esperar dicha respuesta. El código que realiza estas operaciones es el de la figura 75:

```
conexion=crearXMLHttpRequest();
conexion.open("GET", "http://localhost/ProcesadorConsultas.php?numeroTarea="+valor, true);
conexion.onreadystatechange = function(){procesarEventos(valor);};
conexion.setRequestHeader("If-Modified-Since", new Date().toUTCString());
conexion.send(null);
```

Figura 75. JavaScript: petición al servidor

- *crearXMLHttpRequest*: mediante esta función se crea el objeto que permitirá conectarse con el servidor donde está alojado el código PHP que se encarga de realizar la consulta a la base de datos. Como la versión del navegador oficial utilizado en la empresa, Internet Explorer, es superior a la 6, bastará con utilizar el método nativo de JavaScript. Si la versión de este navegador fuera inferior a la 7, se necesitaría crear la conexión con el servidor mediante ActiveX.
- *procesarEventos*: este es el método principal de este fichero puesto que es el encargado de tratar los datos devueltos por el servidor y generar la tabla resultado mostrada al usuario además de crear dos botones que, cuando se accionen, utilizarán estos datos para llevar a cabo distintas operaciones.

  La respuesta del servidor es una cadena que contiene los datos obtenidos de la consulta representados en formato JSON, o una cadena con un mensaje de error en caso de que la tarea por la que ha buscado el usuario no haya arrojado datos. En el primero de los casos se analiza la sintaxis (se *parsea)* de la cadena con el objetivo de obtener un *array* asociativo con el que se pueda acceder a los datos de manera sencilla para crear la tabla mostrada al usuario, mientras que en el segundo se le notifica al usuario el error producido.

  Tal y como se mencionaba anteriormente, el *array* asociativo obtenido se trata con el objetivo de crear una tabla que le facilite al usuario la manera de ver los resultados obtenidos.

Para ello, y tomando como ejemplo una pantalla de 15 pulgadas, se ha decidido que los datos se presenten de la siguiente manera: cada fila representada en la tabla contendrá siete columnas, de manera que todo el contenido quepa en la misma pantalla y no sea necesario moverse por la web para ver los datos.

Además, Por cada fila que contenga los nombres de los campos representados existirá otra que contendrá los datos. Por tanto, todas las filas impares de la tabla representarán nombres de campos mientras que las pares representarán datos. A continuación, en la figura 76, se muestra un fragmento de código en el que se puede observar cómo se crea la tabla en función de los datos:

```
arrayJSON = JSON.parse(respuesta);
for (i=0; i<arrayJSON.columnas.length; i++){
    if((i%COLUMNAS) == 0){//En este caso se crea una nueva serie de filas de nombres y de datos
        if(Math.floor(i/COLUMNAS) != 0){
            tBody.appendChild(filaNombres);
            tBody.appendChild(filaDatos);
        }
        filaNombres = document.createElement("tr");
        filaDatos = document.createElement("tr");
    }
    celdaNombres = document.createElement("td");
    celdaNombres.setAttribute("class", "nombres");
    valorCeldaNombres = document.createTextNode(arrayJSON.columnas[i]);
    celdaNombres.appendChild(valorCeldaNombres);
    filaNombres.appendChild(celdaNombres);

    celdaDatos = document.createElement("td");
    valorCeldaDatos = document.createElement("input");
    valorCeldaDatos.setAttribute("class", "datos");
    valorCeldaDatos.setAttribute("type", "textbox");
    valorCeldaDatos.setAttribute("value", arrayJSON.datos[arrayJSON.columnas[i]]);
    celdaDatos.appendChild(valorCeldaDatos);
    filaDatos.appendChild(celdaDatos);
}
//Introducimos las ultimas filas creadas en la tabla.
tBody.appendChild(filaNombres);
tBody.appendChild(filaDatos);
tabla.appendChild(tBody);
cPrincipal.appendChild(tabla);
```

Figura 76. JavaScript: creación de la tabla

Lo primero que se hace es obtener el *array* asociativo que contiene la matriz de datos devuelta por el servidor. Para cada elemento obtenido se va a crear una columna con un máximo de siete columnas por fila, de manera que siempre que el elemento tratado sea múltiplo de siete se generarán dos nuevas filas, una de nombre y otra de datos, y se insertarán la completas al cuerpo de la tabla. Cada celda de la fila de nombres se completa con el nombre del elemento analizado, el cual se obtiene del *array columnas* mientras que en las celdas de las filas de datos se inserta una caja de texto que a su vez contiene el valor del campo analizado, que se obtiene del *array datos.* Esto se hace así para ofrecer al usuario la posibilidad de modificar los datos ya que si sólo se introdujera el valor en la celda el contenido de esta no podría ser modificado.

Una vez creada la tabla e insertada al cuerpo de la web, se crean dos botones: *Generar CSV*, que servirá para generar el fichero CSV a partir de los datos de la tabla, y *Generar fichero de configuración*, que generará el fichero de texto con la configuración que debe ser cargada en el encaminador. El código para ello es el de la figura 77:

```
btnCSV = document.createElement("input");
btnCSV.setAttribute("type", "submit");
btnCSV.setAttribute("value", "Generar CSV");
btnCSV.setAttribute("class", "botones");
btnCSV.onclick = function(){generarCSV(tabla, idTarea, true);};
btnRun = document.createElement("input");
btnRun.setAttribute("type", "submit");
btnRun.setAttribute("value", "Generar fichero de configuración");
btnRun.setAttribute("class", "botones");
btnRun.onclick = function(){generarFicheroConf(tabla, idTarea);};
cPrincipal.appendChild(btnCSV);
cPrincipal.appendChild(btnRun);
document.body.appendChild(cPrincipal);
```

Figura 77. JavaScript: creación de los botones de acción

- *generarCSV*: mediante este método se crea el fichero con extensión CSV. El contenido de este fichero estará definido por los datos leídos de la tabla, los cuales pueden ser los nativos de la base de datos o bien pueden haber sido modificados por el usuario. Para ello, hay que recorrer tanto las filas de la tabla como cada columna de cada fila, puesto que el orden con el que se guardan los datos es importante.

  Tras obtener el objeto que representa las filas, se recorre de manera que sólo tengamos en cuenta las filas impares de la misma que hacen referencia a los nombres de los campos, tal y como se ha explicado en el método *procesarEventos*. Esto se hace así porque, en caso de que sólo se necesitaran almacenar los datos en el fichero CSV, simplemente bastaría con recorrer las filas pares, las de datos, y guardar el contenido de cada columna. Sin embargo, es necesario comprobar el nombre del campo ya que el nombre del fichero estará formado por el identificador de tarea buscado y por el contenido del campo *SITEID*.

  Además de realizar esta comprobación y formar el nombre del fichero, se irán añadiendo los datos leídos en cada fila de datos, separados por el símbolo “;”, al fichero. Este método es necesario ya que por un lado, es petición expresa del personal del departamento que la aplicación pueda lanzarse de manera independiente a la web por línea de comandos, por otro, hasta que no se consiga acceso a la base de datos en el entorno laboral no podrá sacarse todo el partido a la web. En la figura 78 se muestra el código que realiza las acciones mencionadas anteriormente:

```
filas = tabla.firstChild.childNodes;
for(i=0; i<filas.length; i+=2){
    columnasNombres = filas[i].childNodes;
    columnasDatos = filas[i+1].childNodes;
    for(j=0; j<columnasNombres.length; j++){
        if(columnasNombres[j].childNodes[0].nodeValue == "SITEID")
            nombreCSV += columnasDatos[j].childNodes[0].value + "_" + idTarea + ".csv";
        if(i<((filas.length)-2) || j<((columnasNombres.length)-1)){
            contenidoCSV += columnasDatos[j].childNodes[0].value + ";";
        }
        else{
            contenidoCSV += columnasDatos[j].childNodes[0].value;
        }
    }
}
```

Figura 78. JavaScript: generación del fichero CSV

Una vez generado el contenido del fichero, este se guarda en el directorio FILE_IN y se esperan dos segundos para comprobar que ha sido generado satisfactoriamente. Para ello se utiliza el método explicado a continuación.

- *comprobarGeneracionCSV*: este método comprueba si existe el fichero que se debería haber creado a partir de los datos de la tabla. En caso de que lo encuentre le notifica al usuario que la operación se ha realizado con éxito y abre el directorio que contiene los ficheros con extensión CSV mediante el objeto *Shell*, que será explicado al final del apartado. En caso de que no se encuentre el fichero esperado se notifica al usuario que se ha producido un error.

- *generarFicheroConf*: esta función es la encargada de llamar a la aplicación Java encargada de generar el fichero de texto que contiene la configuración del encaminador. Para ello, se invoca a la función *generarCSV*, que como se ha visto genera el fichero con los datos obtenidos de la tabla. Este paso previo es necesario porque para ejecutar la aplicación Java es necesario pasarle por parámetro el nombre del fichero con extensión CSV en el que se almacenan los datos necesarios para generar la configuración del encaminador. En un futuro este paso intermedio será eliminado. Una vez obtenido dicho nombre y, de nuevo, mediante el objeto *Shell*, se invoca a la aplicación. Al igual que se hacía con la generación del fichero CSV, se esperan dos segundos tras los cuales se comprueba si se ha creado el fichero resultado esperado o por el contrario se crea un fichero de error. Para ello se utiliza la función explicada a continuación.

- *comprobarGeneracionFichConf*: el procedimiento es prácticamente similar al explicado para el método *comprobarGeneracionCSV*, con la salvedad de que en caso encontrar el fichero con extensión *.txt* esperado, además de notificar al usuario de que la operación se ha desarrollado con éxito, se abre el fichero creado, mientras que si no lo encuentra, indica al usuario que se ha producido un error y abre el fichero de error producido por la aplicación Java.

- *Objeto Shell*: este objeto ofrece funciones que permiten realizar, entre otras, operaciones como modificar variables de registro, ejecutar aplicaciones o acceder a las diferentes carpetas del sistema.

En este caso se ha utilizado este objeto tanto para ejecutar aplicaciones como para abrir determinados directorios de manera automática y sin que el usuario tenga que llevar a cabo ninguna acción.

El motivo por el cual se ha decidido hacer esto es por comodidad del usuario. En primer lugar, porque se le evita tener que lanzar la aplicación Java manualmente. En segundo lugar, porque cada vez que se lleva a cabo alguna operación que suponga la creación de algún archivo, o bien se le presenta directamente el directorio en el que se ha creado dicho archivo o bien se le abre, de manera que el usuario puede comprobar los resultados. En la figura 79 se muestran algunos de los comandos mediante los cuales se han ejecutado las acciones mencionadas:

```
shell = new ActiveXObject("WScript.Shell");
shell.run('cmd.exe /c cd  C:/xampp/htdocs/ACTProyecto/bin & java ACTProyecto ' + nombreCSV);
shell.run('cmd.exe /c explorer ' + ruta + nombreCSV.substring(0, nombreCSV.length-4) + '.txt');
```

Figura 79. JavaScript: objeto Shell

En este fragmento se muestra el código necesario tanto para ejecutar la aplicación Java desde la web como para abrir el fichero resultado en caso de que todo haya funcionado satisfactoriamente. Como se puede observar, se ejecuta la línea de comandos de Windows y se añaden las líneas de comando necesarias para llevar a cabo las acciones deseadas. En cuanto a la sintaxis [74], el símbolo "/c" sirve para que la ventana de la línea de comandos desaparezca una vez llevada a cabo la acción propuesta. En el primer comando ejecutado, hay que situarse en el directorio en el que se encuentra el fichero con extensión .class de la clase principal de la aplicación y lanzarla. Como se puede observar, *nombreCSV* hace referencia al nombre del archivo que contiene los datos con los que se va a realizar la configuración del encaminador. En el segundo comando, se abre el fichero de texto generado por la aplicación con el objetivo de que el usuario sepa qué se ha generado y pueda revisarlo. Es importante destacar que en caso de que la ejecución del programa sea errónea, se abrirá el fichero con extensión *.error* creado. En caso de que lo que se decida sea simplemente generar el fichero CSV, se abrirá la carpeta en la que se alojan este tipo de ficheros.

6.5. Aplicación Java

La aplicación desarrollada en Java será la encargada de crear el fichero de texto que contendrá la configuración a cargar en el encaminador, el cual puede ser Cisco o Teldat.

Para generar dicha configuración, la aplicación necesita obtener información de un fichero con extensión CSV generado de manera previa a la ejecución de esta. El objetivo, una vez obtenidos los permisos para poder operar en el entorno laboral con la copia de la base de datos, será eliminar este fichero previo y obtener los datos directamente. Sin embargo, hasta que esto sea posible los instaladores sólo podrán lanzar la aplicación si los datos se obtienen del fichero CSV. En este PFG, el fichero se crea a partir de los datos obtenidos de la base de datos tras realizar la consulta en función del identificador de tarea.

Una vez que se ha obtenido la información del fichero fuente y se han realizado las comprobaciones necesarias para asegurar un correcto funcionamiento, se pasa a generar la configuración del encaminador. Esta se realizará basándose en un conjunto de plantillas que servirán de modelo. Cada una de estas plantillas, generadas por los compañeros del departamento en el que se va a hacer uso de esta aplicación, contiene sentencias que permiten realizar dicha configuración. Estas sentencias, a su vez, están compuestas por un comando y una o varias variables, como se puede observar en la figura 80:

ip address <*ip_lan_router*> <*mascara subred lan*>

Figura 80. Ejemplo de sentencia de configuración

Ahora bien, en cada ejecución no deben utilizarse todas las plantillas en las que puede apoyarse la aplicación, sino que esta debe estar dotada de la inteligencia suficiente como para determinar, en función de una serie de condicionantes, qué plantillas debe utilizar para generar el fichero de configuración. Además, debe diferenciar lo que es comando de lo que son variables para poder sustituir estas por los valores apropiados obtenidos a partir del fichero fuente.

Finalmente, con las sentencias modificadas de cada una de las plantillas utilizadas se genera el fichero resultado. Este fichero contendrá la configuración final del encaminador y el usuario del programa podrá cargarla cuando desee en el mismo.

6.5.1. Distribución de los ficheros

Tras haber estudiado la función de la aplicación de forma resumida, se presenta la distribución de los ficheros con los que interactúa la misma. El objetivo de este apartado es que el lector conozca cómo está estructurado sistema desde un punto de vista gráfico y esto le sirva para comprender mejor el desarrollo explicado en el siguiente apartado.

La estructura es la de la figura 81:

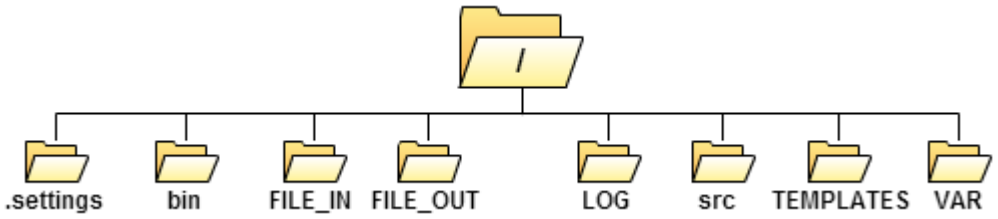

Figura 81. Estructura de directorios de la aplicación

- **bin**: directorio que contiene los ficheros *.class* de la aplicación. Contienen el código compilado interpretado por la máquina virtual de Java.
- **FILE_IN**: directorio que contiene los ficheros generados con la información obtenida de la consulta a la base de datos. Estos son fundamentales puesto que si no existen no se puede generar ningún fichero de configuración. Su estructura interna debe ser fiel a la propuesta en los ficheros que contiene el directorio VAR, el cual se explica más adelante. En caso contrario, no podrá generarse el fichero de configuración. La nomenclatura de los mismos está formada por el *SITEID* y el número de tarea. En la figura 82 se muestra el contenido gráficamente:

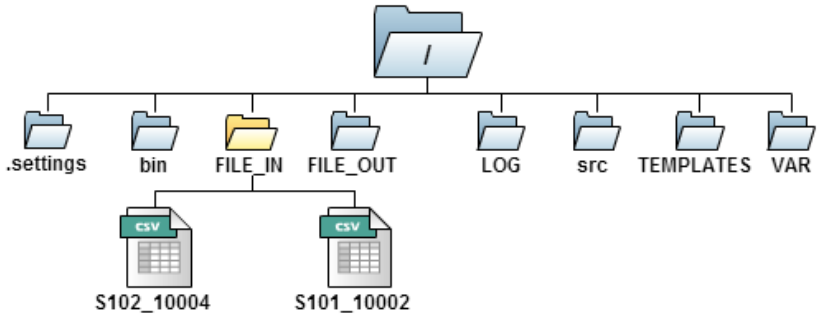

Figura 82. Estructura interna de FILE_IN

- **FILE_OUT**: directorio que contiene el resultado final de la ejecución del programa. Este directorio puede contener dos tipos de fichero:
  - *.txt*: los ficheros con esta extensión contendrán la configuración final del encaminador.
  - *.error*: los ficheros con esta extensión contendrán el tipo de fallo que ha provocado que no se haya podido generar el fichero con la configuración del encaminador.

Su estructura se puede observar en la figura 83:

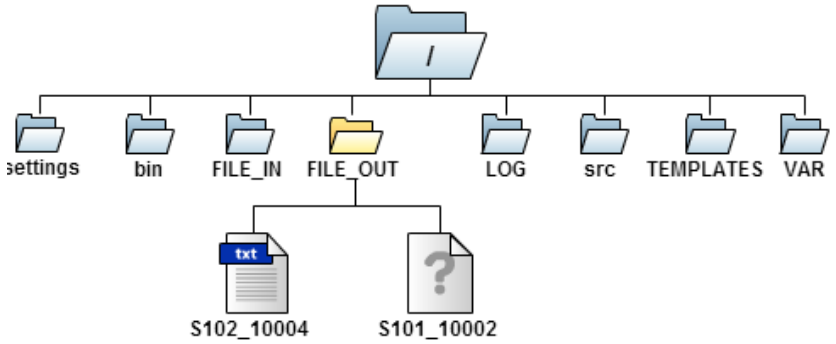

Figura 83. Estructura del directorio FILE_OUT

- **LOG**: directorio que contiene los ficheros de texto que actúan como registros de ejecución. Estos ficheros se generan una vez cada día que la aplicación es ejecutada y en ellos se registran los eventos ocurridos durante la ejecución. Su estructura es la de la figura 84:

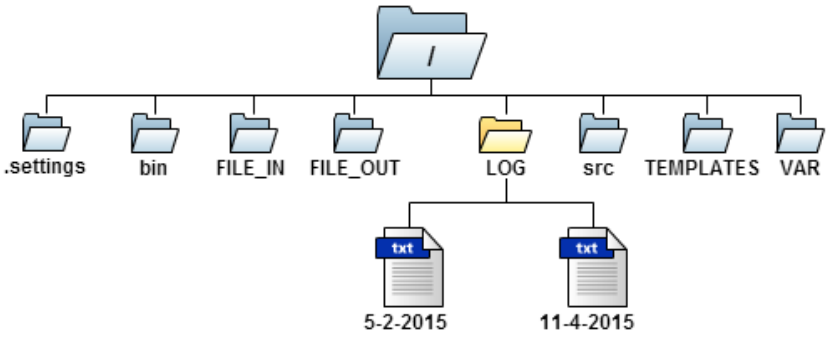

Figura 84. Estructura del directorio LOG

- **src**: directorio que contiene los ficheros *.java* de la aplicación. Contienen el código fuente generado por el programador.
- **TEMPLATES**: directorio que contiene todas las plantillas sobre las que podrá apoyarse la aplicación para generar la configuración final del encaminador. Este directorio, a su vez, se divide en otros directorios en función de la marca del encaminador o del tipo de servicio configurado En la figura 85 se puede ver el árbol de directorios y ficheros:

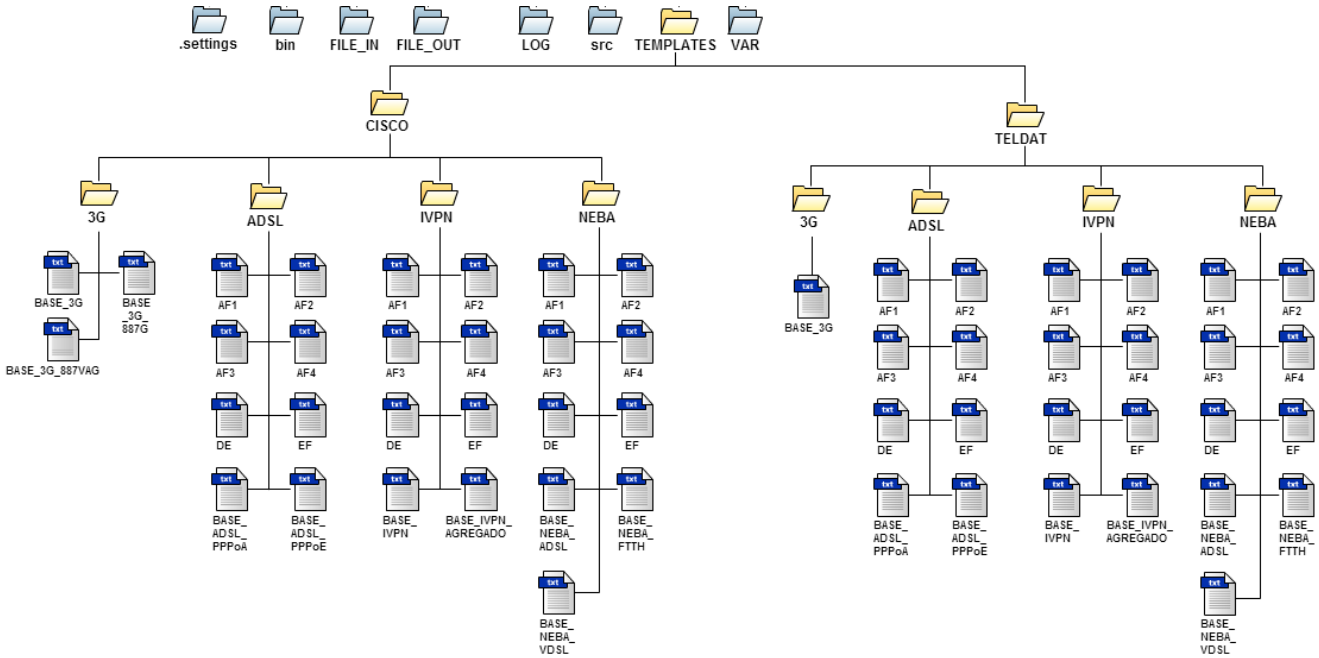

Figura 85. Estructura del directorio TEMPLATES

- **VAR**: directorio que contiene los ficheros que establecen la estructura que tienen que cumplir los archivos generados al realizar la consulta a la base de datos y un fichero denominado "formato_interface.txt" el cual contiene, por orden, la marca del encaminador, el modelo del mismo, el servicio a configurar y el valor de la interfaz. Para cada tipo de servicio existe un fichero modelo. Además, dado que los parámetros de cada uno de los ficheros vienen separados por el símbolo ";", la extensión de estos es CSV. Su estructura se muestra en la figura 86:

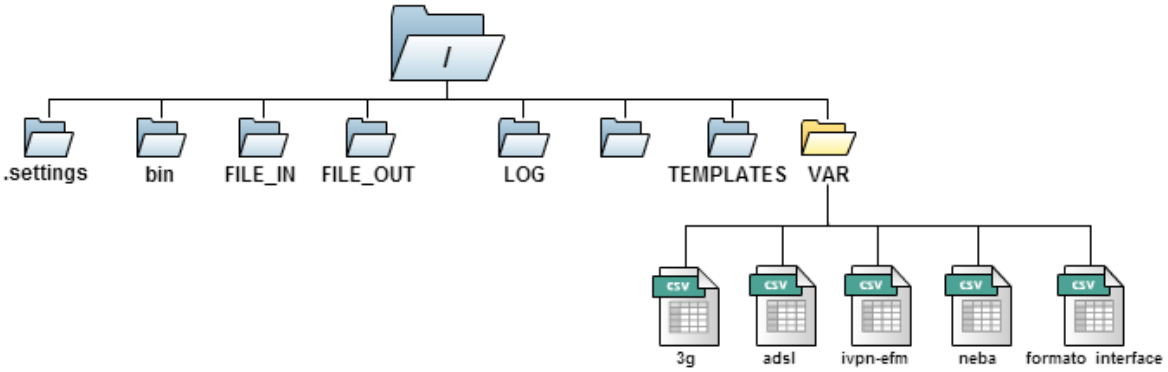

Figura 86. Estructura del directorio VAR

6.5.2. Implementación

La aplicación se ha divido en cinco módulos según las características de las clases que los componen. El diagrama de paquetes mostrado en la figura 87 ayuda a entender la estructura del sistema:

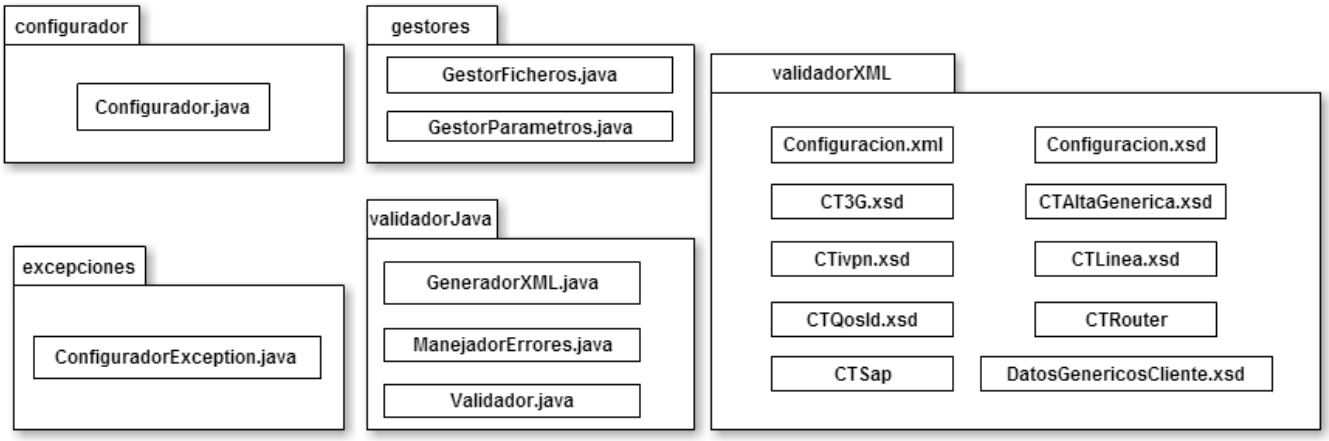

Figura 87. Diagrama de paquetes de la aplicación

A continuación se va a realizar un estudio detallado de cada uno de los módulos. En este estudio se incluyen diagramas de clase así como una explicación de la funcionalidad de cada clase.

Módulo gestores

Este módulo se caracteriza por contener las clases que permiten realizar operaciones con los parámetros de los ficheros fuente y modelo así como escribir registros, errores y resultados en distintos ficheros. El diagrama de clases de este módulo es el de la figura 88:

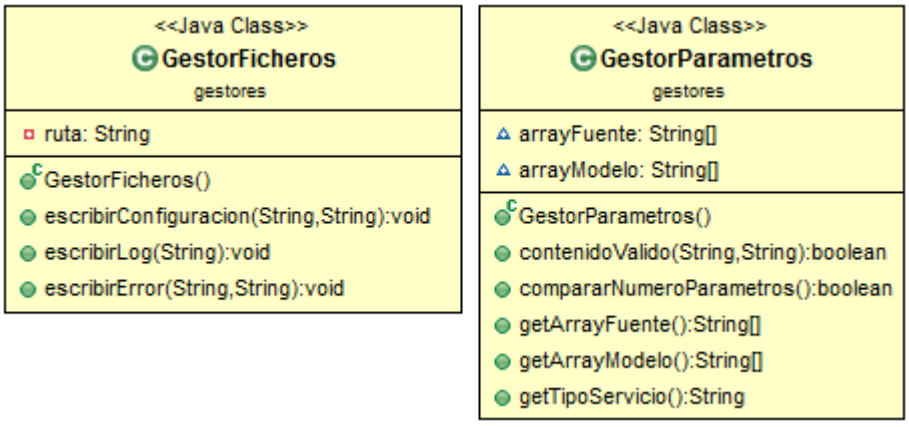

Figura 88. Diagrama de clases del módulo gestores

GestorFicheros.java

Esta clase es la encargada de escribir en los distintos ficheros que genera la aplicación tras su ejecución. Mediante sus métodos, esta clase permite la escritura en:

- **Fichero de configuración**: este fichero almacena, en caso de que la ejecución se realice con éxito, las sentencias necesarias para configurar el encaminador.
- **Fichero de registro**: en este fichero se almacena la información de los eventos ocurridos durante la ejecución del programa y se ubica en el directorio LOG. Se genera un único fichero por cada día que se ejecuta la aplicación de manera que, en caso que en un mismo día se realice más de una configuración, los registros de cada ejecución serán almacenados en el mismo fichero. De manera adicional, se añade información relativa a la fecha y la hora de cada ejecución. El aspecto de uno de estos ficheros es el observado en la figura 89:

```
Thu Mar 26 18:15:58 CET 2015: Validación satisfactoria

Thu Mar 26 18:15:58 CET 2015: Programa lanzado. El fichero procesado es: S104_10008_70003

Thu Mar 26 18:15:58 CET 2015: El fichero S104_10008_70003 ha sido configurado satisfactoriamente


Thu Mar 26 18:16:43 CET 2015: Ejecución detenida. Se ha producido un error y el programa no puede continuar.
Revise el error, el cual ha sido guardado en el siguiente fichero:
C:/Users/Alberto/workspace/ACTProyecto/FILE_OUT/S104_10008_70003.error


Thu Mar 26 18:17:19 CET 2015: Validación satisfactoria

Thu Mar 26 18:17:19 CET 2015: Programa lanzado. El fichero procesado es: S102_10004_70003

Thu Mar 26 18:17:19 CET 2015: El fichero S102_10004_70003 ha sido configurado satisfactoriamente
```

Figura 89. Ejemplo del fichero de registro

- **Fichero de error**: en este fichero se almacena la información que indica el tipo de error producido que ha imposibilitado la creación del fichero de configuración. En este caso, el fichero de error se mostrará en la figura 96.

El parámetro denominado *ruta* almacenará la dirección del directorio raíz donde se ubica el proyecto para que, en caso que se cambie la ubicación del mismo, sólo sea necesario cambiar el valor de esta variable.

Los métodos *escribirConfiguracion* y *escribirError* reciben como parámetros tanto la información que debe ser almacenada como el nombre que hay que darle al fichero en cuestión. Sin embargo, el método *escribirLog* sólo necesita que se le pase por parámetro la información puesto que el nombre del fichero se determinará internamente en función del día de ejecución.

GestorParametros.java

Esta clase se encarga de la interacción con los parámetros de los ficheros CSV del sistema, es decir, de los ficheros fuente y modelo. Los métodos de la clase permiten realizar una validación a más alto nivel de la que veremos en el módulo *validadorJava*, ya que se comprueba si los ficheros están bien formados respetando que los parámetros del mismo estén separados entre símbolos “;”, si estando bien formados tienen contenido o por el contrario vienen vacíos, y si ambos ficheros

coinciden en el número de parámetros o no, es decir, si, por ejemplo, se está configurando una línea ADSL y su fichero modelo consta de 34 parámetros, el fichero fuente del que leerá la aplicación debe estar compuesto igualmente por 34 parámetros. En caso de que las validaciones sean satisfactorias, las variables *arrayFuente* y *arrayModelo* almacenarán los *arrays* formados por los parámetros analizados de cada uno de los ficheros, que serán devueltos por los métodos *getArrayFuente* y *getArrayModelo* respectivamente. Con el método *getTipoServicio* se obtendrá el tipo de servicio que se está configurando.

A continuación se muestra un fragmento del contenido de un fichero fuente, en la figura 90, y un fichero modelo, en la figura 91:

```
servicio;elemento;codigo_organizacion;cliente;siteid;municipio;
```

Figura 90. Fragmento del contenido de un fichero modelo

```
ADSL;ROUTER;1;BT;S101;MADRID;
```

Figura 91. Fragmento del contenido de un fichero fuente

Módulo validadorJava

Este módulo se caracteriza por contener las clases encargadas de realizar la validación de los datos obtenidos en el fichero fuente. El diagrama de clases de este módulo es el de la figura 92:

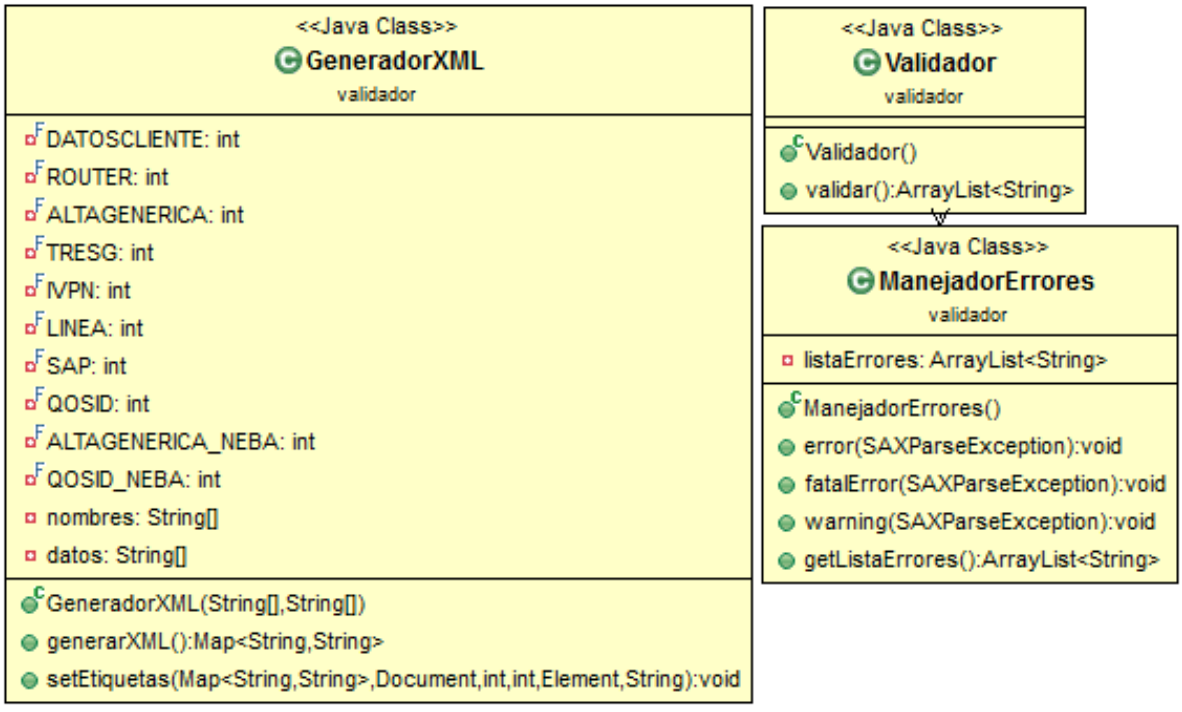

Figura 92. Diagrama de clases del módulo validadorJava

GeneradorXML.java

Una vez que se han llevado a cabo las validaciones explicadas en la clase *GestorParametros.java*, hay que realizar una validación más estricta de los datos del fichero fuente. Para ello, la aplicación genera un archivo XML que se valida contra un *XML Schema* que describe su estructura. Por lo tanto, esta clase es la encargada de generar el documento XML a partir de la información de los parámetros del fichero fuente y del fichero modelo, siendo los primeros los valores de los elementos y los segundos los nombres de los mismos. Las variables de los métodos de la clase observadas en la figura 92 se explican en detalle a continuación:

- *DATOSCLIENTE*: entero que determina el número de parámetros que forman los datos genéricos del cliente.
- *ROUTER*: entero que determina el número de parámetros que forman el componente técnico *router.*
- *ALTAGENERICA*: entero que determina el número de parámetros que forman el componente técnico *alta genérica.*
- *TRESG*: entero que determina el número de parámetros que forman el componente técnico *3G.*
- *IVPN*: entero que determina el número de parámetros que forman el componente técnico *ivpn.*
- *LINEA*: entero que determina el número de parámetros que forman el componente técnico *línea.*
- *SAP*: entero que determina el número de parámetros que forman el componente técnico *SAP.*
- *QOSID*: entero que determina el número de parámetros que forman el componente técnico *QOSID.*
- *ALTAGENERICA*: entero que determina el número de parámetros que forman el componente técnico *alta genérica* para el tipo de servicio NEBA.
- *QOSID_NEBA*: entero que determina el número de parámetros que forman el componente técnico *QOSID* para el tipo de servicio NEBA.
- *nombres: array* pasado por parámetro desde la clase principal que contiene el nombre de cada uno de los parámetros del tipo de servicio a configurar. Estos nombres se obtienen del fichero modelo.
- *datos: array* pasado desde la clase principal que contiene los datos referentes al fichero fuente leído.

Con el constructor de la clase se generan los *arrays* tanto de nombres como de datos cuyo contenido servirá para conformar el documento XML.

El método *generarXML* devuelve un objeto *map* que guarda una relación entre una clave y un valor y en el que no puede haber claves duplicadas. En este caso se almacena como clave el nombre del parámetro obtenido del *array* de nombres y como

valor el dato obtenido del *array* de datos. Su uso será fundamental a la hora de realizar la sustitución de las variables en las sentencias que conformarán el fichero de texto de configuración puesto que la clave hará referencia al nombre de la variable a sustituir y el valor al dato por el que se sustituye. Además, mediante esta función también se genera el documento XML que debe ser validado contra el esquema con el objetivo de producir una configuración válida.

En base al tipo de servicio la estructura del documento puede variar. Por tanto, lo primero que se tendrá que determinar es para qué servicio se quiere generar la configuración del encaminador. Tras esto, se deben incluir los espacios de nombres de los distintos esquemas utilizados. Esto es así porque el esquema contra el que se valida el XML creado está compuesto a su vez de pequeños esquemas de cada uno de los componentes técnicos con el objetivo de hacerlo más versátil. El último paso que hay que dar para completar el documento es incluir los elementos al mismo. Las etiquetas de cada uno de estos elementos deben estar constituidas por el prefijo indicado en el espacio de nombres y por el nombre del parámetro, obtenido del *array* de nombres.

Es importante destacar que todos aquellos parámetros cuyo contenido esté vacío no serán creados en el XML, lo que ayudará a la validación de los datos ya que, a excepción de *plantilla*, todos los demás deben estar presentes.

En la figura 93 se muestra un fragmento de código que ayuda a entender cómo se genera el documento:

```
/*Creamos el elemento raíz*/
configuracion = documento.createElement("tns:configuracion");
configuracion.setAttribute("xmlns:tns", "http://www.example.org/Configuracion");

//Comprobamos que tipo de servicio vamos a configurar, para decidir la estructura del documento XML
switch(datos[0].toLowerCase()){
case "adsl":
    tipoEstructura = documento.createElement("tns:ADSL");
    datosCliente = documento.createElement("tns:datosGenericosCliente");
    ctRouter = documento.createElement("tns:ctRouter");
    altaGenerica = documento.createElement("tns:ctAltaGenerica");
    configuracion.setAttribute("xmlns:dgc", "http://www.example.org/DatosGenericosCliente");
    configuracion.setAttribute("xmlns:ctr", "http://www.example.org/CTRouter");
    configuracion.setAttribute("xmlns:ctag", "http://www.example.org/CTAltaGenerica");
    configuracion.setAttribute("xmlns:xsi", "http://www.w3.org/2001/XMLSchema-instance");
    configuracion.setAttribute("xsi:schemaLocation", "http://www.example.org/Configuracion " +
            "Configuracion.xsd http://www.example.org/DatosGenericosCliente " +
            "DatosGenericosCliente.xsd http://www.example.org/CTRouter " +
            "CTRouter.xsd http://www.example.org/CTAltaGenerica CTAltaGenerica.xsd");

    /*Creamos los elementos para los datos del cliente*/
    setEtiquetas(mapa, documento, 0, DATOSCLIENTE, datosCliente, "dgc");
    /*Creamos los elementos para los datos del router*/
    setEtiquetas(mapa, documento, DATOSCLIENTE, DATOSCLIENTE+ROUTER, ctRouter, "ctr");
    /*Creamos los elementos para los datos del alta generica*/
    setEtiquetas(mapa, documento, DATOSCLIENTE+ROUTER, DATOSCLIENTE+ROUTER+ALTAGENERICA, altaGenerica, "ctag");

    tipoEstructura.appendChild(datosCliente);
    tipoEstructura.appendChild(ctRouter);
    tipoEstructura.appendChild(altaGenerica);
    configuracion.appendChild(tipoEstructura);
    documento.appendChild(configuracion);
    break;
```

Figura 93. Java: fragmento de código que genera el XML para un servicio ADSL

Tras instanciar el objeto que hace referencia al documento, se crea el elemento raíz y se fija el prefijo para el espacio de nombres del mismo. Una vez creado el elemento raíz, se ha de determinar qué tipo de servicio se está configurando puesto que, en función de este, la estructura del elemento *configuración* puede variar.

En este caso, a modo de ejemplo, se ha decidido mostrar la estructura de un documento que hace referencia a una instalación ADSL. Para ello, se crean los elementos que representan cada uno de los componentes técnicos que han de estar presentes en este tipo de configuración, así como los espacios de nombres que los identifican.

Finalmente, por medio del método *setEtiquetas*, cuyo código se puede observar en la figura 94, se crean cada uno de los elementos de cada componente técnico. El último paso es añadir, de forma ordenada, todos los elementos a su correspondiente padre, de manera que quede un documento XML bien formado.

```
public void setEtiquetas(int ini, int fin, Element elemento, String prefijo){
    int i;
    for(i = ini; i<fin; i++){
        if(!datos[i].trim().equals("")){//En caso de que el valor esté vacío, no creamos el elemento
            etiqueta = document.createElement(prefijo+":"+nombres[i].toLowerCase());
            etiqueta.appendChild(document.createTextNode(datos[i].toUpperCase()));
            elemento.appendChild(etiqueta);
            //Introducimos una entrada al mapa
            mapa.put(nombres[i], datos[i].toUpperCase());
        }
    }
}
```

Figura 94. Java: código del método setEtiquetas

Este método crea los elementos de cada componente técnico con el prefijo adecuado. Los parámetros *ini* y *fin* sirven para determinar de qué a qué atributo de los leídos hay que añadir al elemento en cuestión.

Además, añade al objeto *mapa* la pareja compuesta por el nombre del parámetro y su valor. Se ha tomado la decisión de crear este tipo de objeto porque a la hora del tratamiento de datos es más sencillo, ordenado y rápido de utilizar que la tecnología *Document Object Model* o Modelo de Objeto de Documento (DOM) o que acceder a cada elemento del *array* de datos cuando se necesite. El hecho de crear este objeto a la misma vez que el XML es porque así se aprovecha el recorrido de los *arrays* de datos y de nombres.

En la figura 95 se puede ver un ejemplo del fichero XML formado:

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:configuracion xsi:schemaLocation="http://www.example.org/Configuracion Configuracion.xsd
  http://www.example.org/DatosGenericosCliente DatosGenericosCliente.xsd
  http://www.example.org/CTRouter CTRouter.xsd
  http://www.example.org/CTAltaGenerica CTAltaGenerica.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:dgc="http://www.example.org/DatosGenericosCliente"
  xmlns:ctr="http://www.example.org/CTRouter"
  xmlns:ctag="http://www.example.org/CTAltaGenerica"
  xmlns:tns="http://www.example.org/Configuracion">
  - <tns:ADSL>
      - <tns:datosGenericosCliente>
            <dgc:servicio>ADSL</dgc:servicio>
            <dgc:elemento>ROUTER</dgc:elemento>
            <dgc:codigo_organizacion>1</dgc:codigo_organizacion>
            <dgc:cliente>BT</dgc:cliente>
            <dgc:siteid>S101</dgc:siteid>
            <dgc:municipio>MADRID</dgc:municipio>
        </tns:datosGenericosCliente>
      - <tns:ctRouter>
            <ctr:userid>BTTTEL_ATM_S101</ctr:userid>
            <ctr:ip_gestion>10.4.113.194</ctr:ip_gestion>
            <ctr:next_hop>10.4.113.199</ctr:next_hop>
            <ctr:mascara_gestion>255.255.255.0</ctr:mascara_gestion>
            <ctr:marca>CISCO</ctr:marca>
            <ctr:modelo>CISCO1941/K9</ctr:modelo>
        </tns:ctRouter>
      - <tns:ctAltaGenerica>
            <ctag:telefono>915689897</ctag:telefono>
            <ctag:adm_adsl>214748364712</ctag:adm_adsl>
            <ctag:direccion_gestion>10.4.137.196</ctag:direccion_gestion>
            <ctag:tipo_conexion>PPPOE</ctag:tipo_conexion>
            <ctag:tipo_backup>MAIN</ctag:tipo_backup>
            <ctag:vpi>0</ctag:vpi>
            <ctag:vci>35</ctag:vci>
            <ctag:tecnologia>ADSL2+</ctag:tecnologia>
            <ctag:proveedor>TELEFONICA</ctag:proveedor>
            <ctag:qosid>10678</ctag:qosid>
            <ctag:usuario_ppp>101_A_3P</ctag:usuario_ppp>
            <ctag:password_ppp>P101</ctag:password_ppp>
            <ctag:direccion_ip_lan>10.228.21.0</ctag:direccion_ip_lan>
            <ctag:mascara_ip_lan>255.255.255.0</ctag:mascara_ip_lan>
            <ctag:prioridad>1</ctag:prioridad>
            <ctag:ef_value>128</ctag:ef_value>
            <ctag:af4_value>0</ctag:af4_value>
            <ctag:af3_value>0</ctag:af3_value>
            <ctag:af2_value>0</ctag:af2_value>
            <ctag:af1_value>128</ctag:af1_value>
            <ctag:de_value>256</ctag:de_value>
        </tns:ctAltaGenerica>
    </tns:ADSL>
</tns:configuracion>
```

Figura 95. XML: estructura del fichero XML de configuración generado

Validador.java

Esta clase es la encargada de validar el fichero XML de configuración generado a partir de los ficheros fuente y modelo contra el esquema creado para tal caso. Se instancian tanto el objeto que representa al documento XML final como el objeto que representa el esquema contra el que hay que validarlo, se especifica el *ErrorHandler* que se va a utilizar durante la validación y finalmente se obtiene la lista de errores que se hayan producido durante la misma.

ManejadorErrores.java

Para el tratamiento de los errores producidos durante la validación del documento XML se implementa la interfaz *ErrorHandler* mediante la cual podemos determinar el comportamiento de la aplicación en caso de que se produzcan fallos. La implementación llevada a cabo para el método *error* permitirá diferenciar entre los errores producidos por una malformación del documento y los errores producidos por datos que no cumplen con las reglas fijadas en el esquema.

Los fallos de malformación se producen cuando uno de los parámetros leídos, a excepción de *plantilla*, viene vacío ya que, tal y como se comentaba durante la explicación de la clase *GeneradorXML.java*, el programa no lo incluye en el documento XML. En caso de que falte más de un dato, se notificará sólo el primero que ha provocado la malformación de manera que a medida que se vayan resolviendo, se notificarán los siguientes.

En cuanto a los fallos correspondientes al incumplimiento de las reglas fijadas en el esquema, son aquellos en los que, por ejemplo, no se ha formado bien una dirección IP o la cadena que hace referencia al nombre de la provincia excede el tamaño máximo. Es importante destacar que para evitar este tipo de errores se han definido reglas para todos y cada uno de los elementos del documento, independientemente del servicio configurado.

Todos los errores serán almacenados en una lista y notificados al usuario en detalle a través del fichero de error generado en el directorio FILE_OUT, que tendrá un aspecto como el mostrado en la figura 96:

```
ERROR: El parámetro 'ELEMENTO' del bloque de información 'Datos Genéricos Cliente' es erróneo.
Motivo: valor erróneo: ROUTERA
Solución: el elemento configurado debe ser 'ROUTER'


ERROR: El parámetro 'codigo_organizacion' está vacío. Revíselo, gracias.


ERROR: El parámetro 'SITEID' del bloque de información 'Datos Genéricos Cliente' es erróneo.
Motivo: el valor 'S101S' no cumple la estructura correcta
Solución: revise de que el valor está compuesto por una S y tres números del rango 001 - 999. Ejemplo: S001


ERROR: El parámetro 'NEXT_HOP' del CT 'Router' es erróneo.
Motivo: la dirección IP '10.4.313.199' no cumple la estructura correcta
Solución: revise que la estructura de la dirección del siguiente salto sea correcta. Ejemplo: 10.4.113.194
```

Figura 96. Ejemplo del contenido de un fichero de error

Como se puede observar, en este caso se ha producido un error por malformación del XML a causa de que uno de los datos fundamentales no se ha incluido y tres errores por incumplimiento de reglas.

En el primer tipo de fallo se recomienda al usuario que revise el parámetro que ha dado error, el cual está vacío ya que no se ha incluido en el XML.

En el segundo tipo, se le indica al usuario el motivo por el cual se ha producido el error y se propone una solución al mismo.

Los métodos *fatalError* y *warning* se han definido de manera que añadan un error a la lista indicando que el usuario de la aplicación se ponga en contacto con el administrador de la misma, ya que son errores no contemplados que deben tratarse específicamente.

Módulo validadorXML

Este módulo está constituido por todos los ficheros necesarios para la validación de los datos. En la figura 97 se muestra un esquema de su estructura:

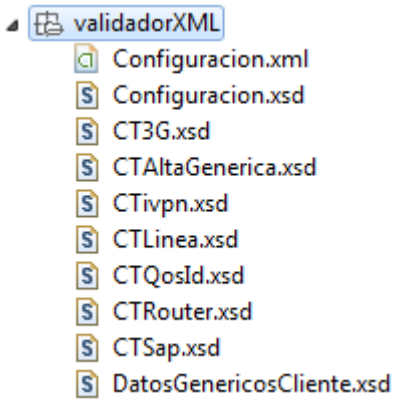

Figura 97. Estructura del módulo validadorXML

Configurador.xml

Este fichero es el generado en la clase *GeneradorXML.java*, cuyo resultado se ha mostrado en la figura 95.

Configurador.xsd

Este fichero, mostrado en la figura 98, contiene el esquema contra el que se valida el *Configurador.xml*:

```
<xs:annotation>
    <xs:documentation xml:lang="es">
    Elemento que contiene la estructura que debe tener el XML correspondiente a una instalación ADSL
    </xs:documentation>
</xs:annotation>
<xs:element name="ADSL">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="datosGenericosCliente" type="dgc:datosCliente"/>
            <xs:element name="ctRouter" type="ctr:ctRouter"/>
            <xs:element name="ctAltaGenerica" type="ctag:ctAltaGenerica"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:annotation>
    <xs:documentation xml:lang="es">
    Elemento que contiene la estructura que debe tener el XML correspondiente a una instalación IVPN
    </xs:documentation>
</xs:annotation>
<xs:element name="IVPN">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="datosGenericosCliente" type="dgc:datosCliente"/>
            <xs:element name="ctRouter" type="ctr:ctRouter"/>
            <xs:element name="ctIVPN" type="ctivp:ctIVPN"/>
            <xs:element name="ctLinea" type="ctl:ctLinea"/>
            <xs:element name="ctSap" type="sap:ctSap"/>
            <xs:element name="ctQosId" type="qos:ctQosId"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:annotation>
    <xs:documentation xml:lang="es">
    Elemento que contiene la estructura que debe tener el XML correspondiente a una instalación 3G
    </xs:documentation>
</xs:annotation>
<xs:element name="TRES_G">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="datosGenericosCliente" type="dgc:datosCliente"/>
            <xs:element name="ctRouter" type="ctr:ctRouter"/>
            <xs:element name="ctTresg" type="cttg:ctTresg"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:annotation>
    <xs:documentation xml:lang="es">
    Elemento que contiene la estructura que debe tener el XML correspondiente a una instalación NEBA
    </xs:documentation>
</xs:annotation>
<xs:element name="NEBA">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="datosGenericosCliente" type="dgc:datosCliente"/>
            <xs:element name="ctRouter" type="ctr:ctRouter"/>
            <xs:element name="ctAltaGenericaNeba" type="ctag:ctAltaGenericaNeba"/>
            <xs:element name="ctQosIdNeba" type="qos:ctQosIdNeba"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:annotation>
    <xs:documentation xml:lang="es">
        Elemento que contiene toda la información acerca de la configuración encaminador.
    </xs:documentation>
</xs:annotation>
<xs:element name="configuracion">
    <xs:complexType>
        <xs:choice minOccurs="1" maxOccurs="1">
            <xs:element ref="tns:ADSL"/>
            <xs:element ref="tns:IVPN"/>
            <xs:element ref="tns:TRES_G"/>
            <xs:element ref="tns:NEBA"/>
        </xs:choice>
    </xs:complexType>
</xs:element>
```

Figura 98. XSD: esquema para validar los documentos de configuración

En la figura 98 no se han incluido los espacios de nombres a fin de introducir sólo la información relevante acerca del esquema.

Como se puede observar, los tipos de dato de los elementos que configuran la estructura del esquema son complejos y, a su vez, estos están compuestos por elementos cuyo tipo se ha definido en otro fichero *XML Schema Definition* o Definición de Esquema XML (XSD).

El objetivo de dividir el esquema en ficheros es proporcionar al sistema de validación la mayor escalabilidad posible de manera que, si en un futuro es necesario añadir o eliminar algún elemento a la estructura principal, no se deban realizar grandes variaciones.

DatosGenericosCliente.xsd

Este fichero contiene el tipo de dato complejo al que hace referencia el elemento *datosGenericosCliente.* La estructura que debe cumplir este elemento es la observada en la figura 99:

```
<xs:complexType name="datosCliente">
    <xs:sequence>
            <xs:element name="servicio" type="tns:listaServicios"/>
            <xs:element name="elemento" type="tns:listaDispositivos"/>
            <xs:element name="codigo_organizacion" type="tns:valorCodOrg"/>
            <xs:element name="cliente" type="tns:longitudCliente_municipio"/>
            <xs:element name="siteid" type="tns:formatoSite"/>
            <xs:element name="municipio" type="tns:longitudCliente_municipio"/>
        </xs:sequence>
</xs:complexType>
```

Figura 99. XSD: estructura del elemento datosGenericosCliente

A su vez, algunos de los tipos de estos elementos tampoco son simples, por lo que se describen a continuación:

- *listaServicios*: establece una lista de posibles valores, los cuales son: ADSL, 3G, NEBA o IVPN.
- *listaDispositivos*: de momento, el contenido de este elemento sólo puede ser ROUTER.
- *valorCodOrg*: establece un rango numérico, el cual va desde 1 hasta 65535.
- *longitudCliente_municipio*: define la longitud máxima que puede tener la cadena del nombre del cliente o de la provincia, que son treinta caracteres en ambos casos.
- *formatoSite*: cadena de texto cuyo primer carácter es una S seguida de tres números. Como el mínimo valor debe ser 001, la regla establecida indica que las posibilidades son: 00X, 0XY o XYY siendo X un número del rango [1-9] e Y un número del rango [0-9]. Ejemplo: S101.

CTRouter.xsd

Este fichero contiene el tipo de dato complejo al que hace referencia el elemento *ctRouter*. La estructura que debe cumplir este elemento es la de la figura 100:

```
<xs:element name="userid" type="tns:formatoRouter"/>
<xs:element name="ip_gestion" type="tns:formatoDirIP"/>
<xs:element name="next_hop" type="tns:formatoDirIP"/>
<xs:element name="mascara_gestion" type="tns:formatoDirIP"/>
<xs:element name="marca" type="tns:marcaEncaminador"/>
<xs:element name="modelo" type="tns:modeloEncaminador"/>
<xs:element name="plantilla" type="xs:string"/>

<xs:complexType name="ctRouter">
    <xs:sequence>
        <xs:element ref="tns:userid"/>
        <xs:element ref="tns:ip_gestion"/>
        <xs:element ref="tns:next_hop"/>
        <xs:element ref="tns:mascara_gestion"/>
        <xs:element ref="tns:marca"/>
        <xs:element ref="tns:modelo"/>
        <xs:element ref="tns:plantilla" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
```

Figura 100. XSD: estructura del elemento ctRouter

A su vez, algunos de los tipos de estos elementos tampoco son simples, por lo que se describen a continuación:

- *formatoRouter*: cadena de texto que debe cumplir una de estas dos estructuras:
  - Tres letras del rango [A-Z], la cadena “TEL_ATM” o “TEL_IP” o “JAZ_IP” o “NB” o “BU”, la cadena “_S” y finalmente tres dígitos. Como el mínimo valor debe ser 001, la regla establecida indica que las posibilidades son: 00X, 0XY o XYY siendo X un número del rango [1-9] e Y un número del rango [0-9]. Ejemplo: AAATEL_IP_S111.
  - Letra R, tres letras del rango [A-Z], tres dígitos, letra S y tres dígitos. Los dos conjuntos de dígitos siguen la misma regla explicada anteriormente. Ejemplo: RAAA111S111.
- *formatoDirIP*: cadena de texto que debe contener una dirección IP válida, es decir, cuatro grupos de números separados por puntos, cada grupo puede contener entre uno y tres dígitos y el número formado por estos dígitos no debe ser nunca superior a 255. Además, es opcional que la máscara de subred acompañe a la dirección IP. La máscara vendrá representada por el carácter “/” y un número formado por uno o dos dígitos. El número formado por estos dígitos no debe ser nunca superior a 32. Ejemplo: 172.168.2.4/24.
- *marcaEncaminador*: establece una lista de posibles valores, los cuales son: CISCO o TELDAT.
- *modeloEncaminador*: establece una lista de posibles valores, los cuales son: CISCO1941, CISCO887, CISCO881, C1+, ATLAS, C1 y C8. En caso de incluir nuevos modelos habría que ampliar esta lista.

CTAltaGenerica.xsd

Este fichero contiene el tipo de dato complejo al que hace referencia el elemento *ctAltaGenerica*. Su estructura es la de la figura 101:

```
<xs:element name="telefono" type="tns:nTelefono"/>
<xs:element name="adm_adsl" type="tns:numAdm"/>
<xs:element name="direccion_gestion" type="tns:formatoDirIP"/>
<xs:element name="ip_fija_wan" type="tns:formatoDirIP"/>
<xs:element name="tipo_conexion" type="tns:tipoConexion"/>
<xs:element name="tipo_backup" type="tns:tipoBackup"/>
<xs:element name="vpi" type="tns:valorVCI_VPI"/>
<xs:element name="vci" type="tns:valorVCI_VPI"/>
<xs:element name="tecnologia" type="tns:tipoTecnologia"/>
<xs:element name="proveedor" type="tns:nombreProveedor"/>
<xs:element name="qosid" type="tns:valorQosid"/>
<xs:element name="usuario_ppp" type="tns:tipoUsuarioPPP"/>
<xs:element name="password_ppp" type="tns:tipoPassword"/>
<xs:element name="direccion_ip_lan" type="tns:formatoDirIP"/>
<xs:element name="mascara_ip_lan" type="tns:formatoDirIP"/>
<xs:element name="prioridad" type="tns:tipoPrioridad"/>
<xs:element name="ef_value" type="tns:valorKbps"/>
<xs:element name="af4_value" type="tns:valorKbps"/>
<xs:element name="af3_value" type="tns:valorKbps"/>
<xs:element name="af2_value" type="tns:valorKbps"/>
<xs:element name="af1_value" type="tns:valorKbps"/>
<xs:element name="de_value" type="tns:valorKbps"/>

<xs:complexType name="ctAltaGenerica">
    <xs:sequence>
        <xs:element ref="tns:telefono"/>
        <xs:element ref="tns:adm_adsl"/>
        <xs:element ref="tns:direccion_gestion"/>
        <xs:element ref="tns:tipo_conexion"/>
        <xs:element ref="tns:tipo_backup"/>
        <xs:element ref="tns:vpi"/>
        <xs:element ref="tns:vci"/>
        <xs:element ref="tns:tecnologia"/>
        <xs:element ref="tns:proveedor"/>
        <xs:element ref="tns:qosid"/>
        <xs:element ref="tns:usuario_ppp"/>
        <xs:element ref="tns:password_ppp"/>
        <xs:element ref="tns:direccion_ip_lan"/>
        <xs:element ref="tns:mascara_ip_lan"/>
        <xs:element ref="tns:prioridad"/>
        <xs:element ref="tns:ef_value"/>
        <xs:element ref="tns:af4_value"/>
        <xs:element ref="tns:af3_value"/>
        <xs:element ref="tns:af2_value"/>
        <xs:element ref="tns:af1_value"/>
        <xs:element ref="tns:de_value"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="ctAltaGenericaNeba">
    <xs:sequence>
        <xs:element ref="tns:telefono"/>
        <xs:element ref="tns:adm_adsl"/>
        <xs:element ref="tns:direccion_gestion"/>
        <xs:element ref="tns:ip_fija_wan"/>
        <xs:element ref="tns:tipo_conexion"/>
        <xs:element ref="tns:tipo_backup"/>
        <xs:element ref="tns:vpi"/>
        <xs:element ref="tns:vci"/>
        <xs:element ref="tns:tecnologia"/>
        <xs:element ref="tns:proveedor"/>
        <xs:element ref="tns:qosid"/>
        <xs:element ref="tns:usuario_ppp"/>
        <xs:element ref="tns:password_ppp"/>
        <xs:element ref="tns:direccion_ip_lan"/>
        <xs:element ref="tns:mascara_ip_lan"/>
        <xs:element ref="tns:prioridad"/>
    </xs:sequence>
</xs:complexType>
```

Figura 101. XSD: estructura del elemento altaGenerica

Como se puede observar, existen dos tipos de elemento que se diferencian en su estructura ya que NEBA tiene una serie de peculiaridades.

A su vez, algunos de los tipos de estos elementos tampoco son simples, por lo que se describen a continuación:

- *nTelefono*: cadena que aunque habitualmente sólo esté constituida de dígitos, puede contener en alguna ocasión un conjunto de caracteres. Su longitud máxima es de quince caracteres.
- *numAdm*: establece un rango numérico de doce dígitos, el cual va desde 000000000001 hasta 999999999999.
- *formatoDirIP*: explicado en el esquema *CTRouter.xsd* en la página 146.
- *tipoConexión*: establece una lista de posibles valores, los cuales son: PPPoA o PPPoE.
- *tipoBackup*: establece una lista de posibles valores, los cuales son: MAIN o BU.
- *valorVCI_VPI*: establece un rango numérico válido, el cual va desde 1 hasta 99.
- *tipoTecnologia*: establece una lista de posibles valores, los cuales son: ADSL, ADSL2+, VDSL y FTTH.
- *nombreProveedor*: establece una lista de posibles valores, los cuales son: TELEFONICA, JAZZTEL u ONO.
- *valorQosid*: establece un rango numérico válido, el cual va desde 1 hasta 99999.
- *tipoUsuarioPPP*: cadena de texto formada por tres dígitos y una de las siguientes cadenas: “_D_A_3P”, “_A_3P” o “_IP_T” o “_NB”. Como el mínimo valor válido del conjunto de dígitos debe ser 001, la regla establecida indica que las posibilidades son: 00X, 0XY o XYY siendo X un número del rango [1-9] e Y un número del rango [0-9]. Ejemplo: 111_D_A_3P.
- *tipoPassword*: cadena de texto formada por una P y tres números que cumplen la regla descrita en tipo *tipoUsuarioPPP*. Ejemplo: P123.
- *tipoPrioridad*: número entero del rango [1-9].
- *valorKbps*: establece un rango numérico válido, el cual va desde 1 hasta 99999999.

CT3G.xsd

Este fichero contiene el tipo de dato complejo al que hace referencia el elemento *ctTresg.* La estructura que debe cumplir este elemento es la observada en la figura 102:

```
<xs:element name="sim" type="tns:nTelefono"/>
<xs:element name="apn" type="tns:tipoAPN"/>
<xs:element name="nombre_vpn" type="tns:tipoNombreVPN"/>
<xs:element name="qosid" type="tns:valorQosid"/>
<xs:element name="nombre_usuario" type="tns:tipoNombreUsuario"/>
<xs:element name="direccion_ip_fija" type="tns:formatoDirIP"/>
<xs:element name="ip_gestion" type="tns:formatoDirIP"/>
<xs:element name="nemonico" type="tns:tipoNemonico"/>
<xs:element name="ip_lan" type="tns:formatoDirIP"/>
<xs:element name="mascara_ip_lan" type="tns:formatoDirIP"/>
<xs:element name="prioridad" type="tns:tipoPrioridad"/>

<xs:complexType name="ctTresg">
    <xs:sequence>
        <xs:element ref="tns:sim"/>
        <xs:element ref="tns:apn"/>
        <xs:element ref="tns:nombre_vpn"/>
        <xs:element ref="tns:qosid"/>
        <xs:element ref="tns:nombre_usuario"/>
        <xs:element ref="tns:direccion_ip_fija"/>
        <xs:element ref="tns:ip_gestion"/>
        <xs:element ref="tns:nemonico"/>
        <xs:element ref="tns:ip_lan"/>
        <xs:element ref="tns:mascara_ip_lan"/>
        <xs:element ref="tns:prioridad"/>
    </xs:sequence>
</xs:complexType>
```

Figura 102. XSD: estructura del elemento tresg

A su vez, algunos de los tipos de estos elementos tampoco son simples, por lo que se describen a continuación:

- *nTelefono*: entero de nueve dígitos. Está compuesto por una cifra cuyo valor es 6 o 7 seguida de ocho cifras cuyo valor puede variar entre 0 y 9.
- *tipoAPN*: establece una lista de posibles valores que, de momento, se limitan a INTERNET.BRTM.COM.
- *tipoNombreVPN*: establece una lista de posibles valores, los cuales son: INTERNA o EXTERNA.
- *valorQosid*: explicado en el esquema *CTAltaGenerica.xsd* en la página 148.
- *tipoNombreUsuario*: cadena de texto formada por tres letras del rango [A-Z], la cadena “BU_S” y tres dígitos cuyo mínimo valor puede ser 001, por lo que la regla establecida indica que las posibilidades son: 00X, 0XY o XYY siendo X un número del rango [1-9] e Y un número del rango [0-9]. Ejemplo: AAABU_S111.
- *formatoDirIP*: explicado en el esquema *CTRouter.xsd* en la página 146.
- *tipoNemonico*: establece una lista de posibles valores, los cuales son: @BI, @BU o @BM.
- *tipoPrioridad*: explicado en el esquema *CTAltaGenerica.xsd* en la página 148.

CTivpn.xsd

Este fichero contiene el tipo de dato complejo al que hace referencia el elemento *ctIVPN*. La estructura que debe cumplir este elemento es la de la figura 103:

```
<xs:element name="user_ivpn" type="tns:tipoUserIVPN"/>
<xs:element name="tipo_puerto" type="tns:tipoPuerto"/>
<xs:element name="routing_cpe" type="tns:tipoRouting"/>
<xs:element name="as_number" type="tns:valorASNumber"/>
<xs:element name="reflector" type="tns:reflec"/>
<xs:element name="puerto_conexion" type="tns:longitudPuerto"/>
<xs:element name="qosid" type="tns:valorQosid"/>
<xs:element name="line_vel" type="tns:velocidad"/>
<xs:element name="numero_servicios" type="tns:numServ"/>
<xs:element name="friendly_name" type="tns:tipoFriendlyName"/>
<xs:element name="ip_wan_vpls" type="tns:formatoDirIP"/>

<xs:complexType name="ctIVPN">
    <xs:sequence>
        <xs:element ref="tns:user_ivpn"/>
        <xs:element ref="tns:tipo_puerto"/>
        <xs:element ref="tns:routing_cpe"/>
        <xs:element ref="tns:as_number"/>
        <xs:element ref="tns:reflector"/>
        <xs:element ref="tns:line_vel"/>
        <xs:element ref="tns:puerto_conexion"/>
        <xs:element ref="tns:qosid"/>
        <xs:element ref="tns:numero_servicios"/>
        <xs:element ref="tns:friendly_name"/>
        <xs:element ref="tns:ip_wan_vpls"/>
    </xs:sequence>
</xs:complexType>
```

Figura 103. XSD: estructura del elemento ctIVPN

A su vez, algunos de los tipos de estos elementos tampoco son simples, por lo que se describen a continuación:

- *tipoUserIVPN*: cadena de texto formada por tres letras del rango [A-Z], tres dígitos, la letra S y otros tres dígitos. Como el mínimo valor válido del conjunto de dígitos debe ser 001 en ambos casos, la regla establecida indica que las posibilidades son: 00X, 0XY o XYY siendo X un número del rango [1-9] e Y un número del rango [0-9]. Ejemplo:AAA111S111.
- *tipoPuerto:* establece una lista de posibles valores, los cuales son: AGREGADO o PUNTO.
- *tipoRouting*: establece una lista de posibles valores que, de momento, se limitan a BGP.
- *valorASNumber*: establece un rango numérico válido, el cual va desde 1 hasta 65535.
- *reflec*: establece una lista de posibles valores, los cuales son: SI o NO.
- *longitudPuerto*: define la longitud máxima que puede tener la cadena del nombre que contiene el equipo y el puerto donde se conectará la línea del cliente, que son treinta caracteres.

- *valorQosid*: explicado en el esquema *CTAltaGenerica.xsd* en la página 148.
- *velocidad*: establece un rango numérico válido para la velocidad de la línea expresada en Mbps, el cual va desde 1 hasta 1000.
- *numServ:* establece un rango numérico válido, el cual va desde 1 hasta 4096.
- *tipoFriendlyName*: establece una lista de posibles valores que, de momento, se limitan a VPN_BT_EU.
- *formatoDirIP*: explicado en el esquema *CTRouter.xsd* en la página 146.

CTLinea.xsd

Este fichero contiene el tipo de dato complejo al que hace referencia el elemento *linea.* La estructura que debe cumplir este elemento se muestra en la figura 104:

```
<xs:element name="administrativo" type="tns:numAdm"/>
<xs:element name="tipo_linea" type="tns:tipoLinea"/>
<xs:element name="vlan_agregado" type="tns:valorVLAN"/>
<xs:element name="operadora" type="tns:nombreOperadora"/>

<xs:complexType name="ctLinea">
    <xs:sequence>
        <xs:element ref="tns:administrativo"/>
        <xs:element ref="tns:tipo_linea"/>
        <xs:element ref="tns:vlan_agregado"/>
        <xs:element ref="tns:operadora"/>
    </xs:sequence>
</xs:complexType>
```

Figura 104. XSD: estructura del elemento linea

A su vez, algunos de los tipos de estos elementos tampoco son simples, por lo que se describen a continuación:

- *numAdm:* establece un rango numérico de doce dígitos, el cual va desde 000000000001 hasta 999999999999.
- *tipoLinea*: el contenido de los elementos cuyo tipo de dato sea este puede ser: PUNTO A PUNTO, AGREGADO.
- *valorVLAN*: establece un rango numérico, el cual va desde 1 hasta 4096.
- *nombreOperadora*: establece una lista de posibles valores, los cuales son: TELEFÓNICA, JAZZTEL u ONO.

CTSap.xsd

Este fichero contiene el tipo de dato complejo al que hace referencia el elemento *ctSap*. La estructura que debe cumplir este elemento es la observada en la figura 105:

```
<xs:element name="service_id" type="tns:valorServiceID"/>
<xs:element name="service_vlan" type="tns:valorVLAN"/>
<xs:element name="service_ip" type="tns:formatoDirIP"/>
<xs:element name="service_qos" type="tns:valorQosid"/>

<xs:complexType name="ctSap">
    <xs:sequence>
        <xs:element ref="tns:service_id"/>
        <xs:element ref="tns:service_vlan"/>
        <xs:element ref="tns:service_ip"/>
        <xs:element ref="tns:service_qos"/>
    </xs:sequence>
</xs:complexType>
```

Figura 105. XSD: estructura del elemento ctSap

A su vez, algunos de los tipos de estos elementos tampoco son simples, por lo que se describen a continuación:

- *valorServiceID:* establece un rango numérico, el cual va desde 1 hasta 99999999.
- *valorVLAN:* explicado en el esquema *CT_Linea.xsd* en la página 151.
- *formatoDirIP*: explicado en el esquema *CTRouter.xsd* en la página 146.
- *valorQosid*: explicado en el esquema *CTAltaGenerica.xsd* en la página 148.

CTQosId.xsd

Este fichero contiene el tipo de dato complejo al que hace referencia el elemento *ctQosId*. La estructura que debe cumplir este elemento se puede ver en la figura 106:

```
<xs:element name="service_pir" type="tns:valorPIR"/>
<xs:element name="service_efpctj" type="tns:valorPCTJ"/>
<xs:element name="service_af4pctj" type="tns:valorPCTJ"/>
<xs:element name="service_af3pctj" type="tns:valorPCTJ"/>
<xs:element name="service_af2pctj" type="tns:valorPCTJ"/>
<xs:element name="service_af1pctj" type="tns:valorPCTJ"/>
<xs:element name="service_depctj" type="tns:valorPCTJ"/>
<xs:element name="ef_service_pir" type="tns:valorClases"/>
<xs:element name="af4_service_pir" type="tns:valorClases"/>
<xs:element name="af3_service_pir" type="tns:valorClases"/>
<xs:element name="af2_service_pir" type="tns:valorClases"/>
<xs:element name="af1_service_pir" type="tns:valorClases"/>
<xs:element name="de_service_pir" type="tns:valorClases"/>
<xs:element name="service_mgnt_pir" type="tns:valorGestion"/>

<xs:complexType name="ctQosId">
    <xs:sequence>
        <xs:element ref="tns:service_pir"/>
        <xs:element ref="tns:service_efpctj"/>
        <xs:element ref="tns:service_af4pctj"/>
        <xs:element ref="tns:service_af3pctj"/>
        <xs:element ref="tns:service_af2pctj"/>
        <xs:element ref="tns:service_af1pctj"/>
        <xs:element ref="tns:service_depctj"/>
        <xs:element ref="tns:ef_service_pir"/>
        <xs:element ref="tns:af4_service_pir"/>
        <xs:element ref="tns:af3_service_pir"/>
        <xs:element ref="tns:af2_service_pir"/>
        <xs:element ref="tns:af1_service_pir"/>
        <xs:element ref="tns:de_service_pir"/>
        <xs:element ref="tns:service_mgnt_pir"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="ctQosIdNeba">
    <xs:sequence>
        <xs:element ref="tns:service_pir"/>
        <xs:element ref="tns:service_efpctj"/>
        <xs:element ref="tns:service_af4pctj"/>
        <xs:element ref="tns:service_af3pctj"/>
        <xs:element ref="tns:service_af2pctj"/>
        <xs:element ref="tns:service_af1pctj"/>
        <xs:element ref="tns:service_depctj"/>
        <xs:element ref="tns:ef_service_pir"/>
        <xs:element ref="tns:af4_service_pir"/>
        <xs:element ref="tns:af3_service_pir"/>
        <xs:element ref="tns:af2_service_pir"/>
        <xs:element ref="tns:af1_service_pir"/>
        <xs:element ref="tns:de_service_pir"/>
    </xs:sequence>
</xs:complexType>
```

Figura 106. XSD: estructura del elemento ctQosId

Como se puede observar, existen dos tipos de elemento que se diferencian en su estructura ya que NEBA tiene una serie de peculiaridades.

A su vez, algunos de los tipos de estos elementos tampoco son simples, por lo que se describen a continuación:

- *valorPIR:* establece un rango numérico, el cual va desde 1 hasta 1000000.
- *valorPCTJ:* establece un rango numérico, el cual va desde 1 hasta 99.
- *valorClases:* establece un rango numérico, el cual va desde 1 hasta 999999.
- *valorGestion:* establece un rango numérico, el cual va desde 1 hasta 99999.

Módulo excepciones

Este módulo se caracteriza por contener las excepciones que permiten manejar fallos que pueden surgir durante la ejecución del programa. El diagrama de clases de este módulo es el mostrado en la figura 107:

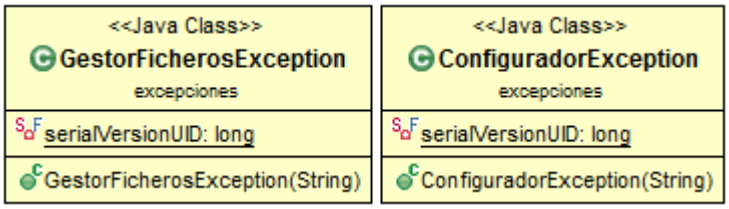

Figura 107. Diagrama de clases del módulo excepciones

GestoresException.java

Esta clase ha sido desarrollada para informar de los errores producidos en la clase GestorFicheros.java. Esta excepción se lanza en caso de detectar que se ha producido un error durante la escritura del fichero de configuración con el que se está interactuando y permite identificar el nombre del mismo así como la ruta en la que se encuentra.

ConfiguradorException.java

Esta clase ha sido desarrollada para informar de los errores producidos en la clase Configurador.java. Esta excepción se lanza en los siguientes casos:

- En caso de que se produzca un problema a la hora de fijar la interfaz WAN. Puede ser que la marca o el modelo a configurar no se encuentren en el fichero formato_interface.txt, que este mismo fichero este vacío, que no se encuentre el fichero en la ruta especificada o que se produzca un error durante la lectura de su contenido. En cada caso se identifica el motivo para que el usuario sea consciente de él.
- En caso de que se produzca un problema durante la obtención de los parámetros referentes a las calidades de servicio. Puede ser que la suma de porcentajes de cada clase de servicio supere el 100% o que la suma del ancho de banda de cada clase de servicio supere el ancho de banda total del puerto contratado.

- En caso de que se produzca un problema a la hora de generar el texto de la configuración. Puede ser que una de las plantillas utilizadas esté vacía o inaccesible, se produzca un error en la lectura de las mismas o que una de las variables a sustituir no se encuentre en el objeto *mapa*.

Módulo configurador

Este módulo se caracteriza por contener la clase encargada de realizar las operaciones necesarias para generar el fichero de texto de configuración. En la figura 108 se puede ver el diagrama de clases de este módulo:

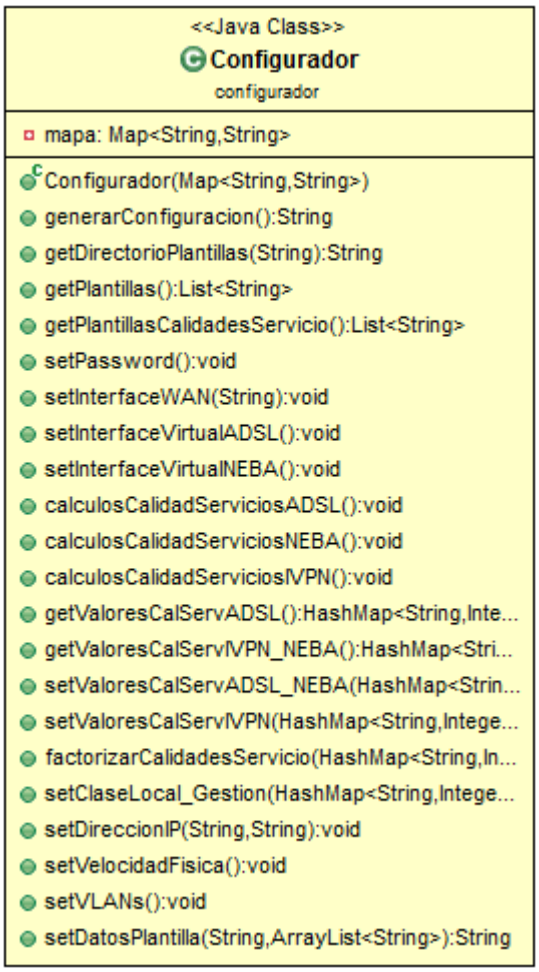

Figura 108- Diagrama de clases del módulo configurador

Configurador.java

Esta clase es la que realiza las acciones más importantes de la aplicación puesto que en ella se determinan qué plantillas se van a utilizar, se obtienen las diferentes calidades de servicio a configurar, se añaden elementos al objeto *mapa* que contiene las parejas nombre-valor y finalmente se genera el fichero de texto que contiene la configuración del encaminador sustituyendo las variables por los valores almacenados dicho objeto.

La variable *mapa* se recibe por parámetro a la hora de instanciar la clase *Configurador.java.* Almacena todos los valores leídos en el fichero fuente y validados mediante XML y XML Schema.

A continuación se ofrece un breve resumen de las acciones que se llevan a cabo en los métodos más importantes de la clase:

- *generarConfiguracion*: lo primero que hay que hacer es determinar el directorio en el que se encuentran las plantillas que debemos utilizar para confeccionar el fichero final. Esto es posible gracias a la función *getDirectorioPlantillas*, que en base a la marca del encaminador y al tipo de servicio obtiene la ruta de dicho directorio.

  El siguiente paso es determinar qué plantillas de las disponibles serán necesarias (puede ver gráficamente las plantillas disponibles consultando la figura 85), para lo que se desarrolla el método *getPlantillas.* Este método devuelve una lista constituida por los nombres de todas las plantillas que será necesario utilizar para generar la configuración. Primero se determina la plantilla base en función del tipo de servicio a configurar mientras que después, mediante el método *getPlantillasCalidadesServicio,* se obtienen las específicas de las calidades de servicio. Para ello, se comprueba que los valores *ef, af4, af3, af2, af1* y *de,* tomados del objeto *mapa* a partir de sus claves, sean relevantes, es decir, que el ancho de banda reservado para cada clase sea mayor a cero. Sólo en estos se añadirá a la lista la plantilla cuyo nombre es similar al valor evaluado.

  Una vez constituida la lista, se procede a completar la información almacenada en el *mapa* ya que los datos leídos y validados del fichero fuente no son suficientes para sustituir todas las variables de cada una de las plantillas. Por este motivo, se desarrolla un conjunto de funciones que, a partir de estos datos, añade nuevas parejas clave - valor al objeto. Es importante recordar que la clave se corresponderá con el nombre de la variable a sustituir mientras que el valor se corresponde con el dato por el que la sustituye. Estas funciones son las que se detallan a continuación.

- *setPassword*: función que genera la contraseña de acceso por gestión al encaminador a partir de las tres primeras letras del nombre de usuario y de la marca del encaminador y la añade al objeto *mapa.*

- *setInterfaceWAN*: método que determina la interfaz WAN configurada en función de la marca y del modelo del encaminador y del tipo de servicio que se quiere configurar y la añade al objeto *mapa.* El valor de esta interfaz se buscará en el fichero “formato_interface.csv”

- *setInterfaceVirtualADSL*: función que determina la interfaz virtual configurada para el servicio ADSL en base a la marca del encaminador y al tipo de conexión y la añade al objeto *mapa.*
- *setInterfaceVirtualNEBA*: función que determina la interfaz virtual configurada para el servicio NEBA en base a la marca del encaminador y del tipo de tecnología y la añade al objeto *mapa.*
- *calculosCalidadServiciosADSL*: mediante esta función se van a añadir al *mapa* todos los valores que tienen relación con las calidades de servicio contratadas para una línea ADSL. Para ello, utiliza los métodos *getValoresCalServADSL*, y *setValoresCalServADSL_NEBA setClaseLocal_*Gestion, los cuales son descritos a continuación.
- *getValoresCalServADSL*: se obtienen los valores del ancho de banda reservado para cada clase, se calcula el *rate* o velocidad total de puerto y se añaden a un objeto *map.* El motivo por el que se usa este objeto y no un array es que nos permite identificar cada valor por su clave, lo que facilita su uso en funciones posteriores.
- *setValoresCalServADSL_NEBA*: mediante esta función se calculan los valores de las variables que hacen referencia tanto al porcentaje del total reservado para cada calidad de servicio como al ancho de banda en Kbps y se añaden al *mapa*. Como su nombre indica, esta función puede aplicarse a líneas ADSL y NEBA. Particularmente para servicios ADSL, se aplica un factor de corrección puesto que hay que tener en cuenta las cabeceras de los paquetes ATM.
- *setClaseLocal_Gestion*: en los servicios ADSL y NEBA, esta función se encarga de destinar el 5% del ancho de banda reservado para la clase DE para la gestión y el mantenimiento de la línea. El 2% se reserva para el tráfico de mantenimiento del interfaz mientras que el 3% restante se usa para asegurar el acceso por gestión al encaminador.
- *calculosCalidadServiciosNEBA*: mediante esta función se van a añadir al *mapa* todos los valores que tienen relación con las calidades de servicio contratadas para una línea NEBA. Para ello, utiliza los métodos *getValoresCalServIVPN_NEBA*, la cual se describe a continuación, *setValoresCalServADSL_NEBA* y *setClaseLocal_*Gestion, ambos descritos anteriormente.
- *getValoresCalServIVPN_NEBA*: se obtiene tanto el porcentaje como el ancho de banda reservado para cada clase, se calcula el *rate* o velocidad total de puerto y se añaden a un objeto *map.* Además, se comprueba que la suma de los porcentajes es igual al 100% y que la suma de todos los anchos de banda de las calidades de servicio es igual a la velocidad total de puerto.
- *calculosCalidadServiciosIVPN*: mediante esta función se van a añadir al *mapa* todos los valores que tienen relación con las calidades de servicio contratadas para una línea IVPN. Para ello, utiliza los métodos *getValoresCalServIVPN_NEBA*, se ha descrito anteriormente, *factorizarCalidadesServicio* y *setValoresCalServIVPN*, las cuales se describen a continuación.

- *factorizarCalidadesServicio*: en esta clase se factorizan los valores que indican la velocidad asignada a cada clase de servicio. La factorización se produce porque para asegurar que todo el tráfico contratado por cliente sea efectivo es necesario que los valores configurados en el encaminador sean ligeramente superiores a los teóricos obtenidos del fichero fuente, puesto que hay que tener en cuenta los tamaños de las cabeceras. Los factores de corrección establecidos han sido fijados por el departamento de Ingeniería de Red.
- *setValoresCalServIVPN*: mediante esta función se añaden al *mapa* los valores de las variables que hacen referencia al ancho de banda contratado para cada clase. Además, en caso de que el encaminador configurado sea Cisco, hay que añadir esta velocidad en bits.
- *setDireccionIP*: función que, en base a la dirección IP y la máscara de subred expresada en notación CIDR pasadas por parámetro, determina la dirección, la máscara de subred, la dirección de red, la *wildcard mask* y la dirección parcial. Aunque la *wildcard mask* aún no se utiliza, se ha incluido a petición de los compañeros del departamento con vistas a utilizarlo en el futuro.
- *setVelocidadFisica*: función que determina la velocidad física de la línea, expresada en Mbps.
- *setVLANs*: función que determina la interfaz VLAN a configurar para los servicios IVPN y NEBA.
- *setDatosPlantilla*: función que genera el contenido del fichero de configuración resultado. Para cada una de las plantillas utilizadas para generar el resultado se realiza un análisis línea por línea. En este análisis, el programa busca el símbolo del porcentaje (%), el cual ha sido predefinido como comienzo y final de las variables que hay que sustituir por los datos del objeto *mapa*. De esta manera, una línea a la que hay que realizar modificaciones tiene el aspecto de la figura 109:

```
description CONEXION ADSL admin %adm_adsl% - telefono %telefono%
```

Figura 109. Línea de una de las plantillas que contiene variables que deben ser sustituidas

Una vez que identifica las posiciones de los dos símbolos, modifica la línea de manera que sustituye la variable por su valor correspondiente. Para ello utiliza el objeto *mapa* que, como se ha explicado durante el capítulo, identifica un valor a partir de una clave que, en este caso, es el nombre de la variable. Este proceso se realiza siempre que, tras haber realizado la sustitución, se encuentren los símbolos % que delimitan el comienzo y el fin de las variables. El resultado de aplicar este método, para la línea tomada como ejemplo en la figura 109 es el observado en la 110:

```
description CONEXION ADSL admin 214748364712 - telefono 915689897
```

Figura 110. Resultado tras modificar las variables por su valor

Cada línea, tras ser modificada, es incluida en el fichero de texto generado como resultado.

CAPÍTULO 7

MANUAL DE USUARIO

7. Manual de usuario

Este capítulo está enfocado a mostrar el funcionamiento del sistema. El objetivo es, una vez que se conoce el diseño del sistema y las funcionalidades del mismo, que el lector pueda ver la interacción entre la herramienta web y el usuario de la misma.

7.1.Modificación de las opciones de Internet Explorer

7.1.1. ActiveX

Para el correcto funcionamiento de la aplicación es necesario realizar una serie de modificaciones en la configuración de Internet Explorer para permitir la descarga de controles *ActiveX*. Estas modificaciones son las que se muestran, tras cada paso, en las figuras 111, 112 y 113:

1. Abrir **Opciones de Internet**

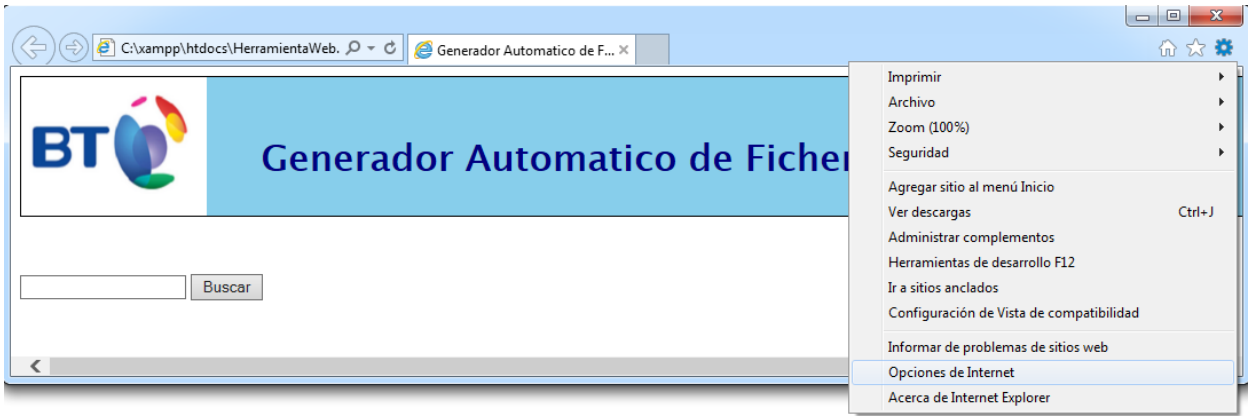

Figura 111. Opciones de Internet

2. En la ventana emergente, se selecciona la pestaña **Seguridad** y se abre **Nivel Personalizado…**

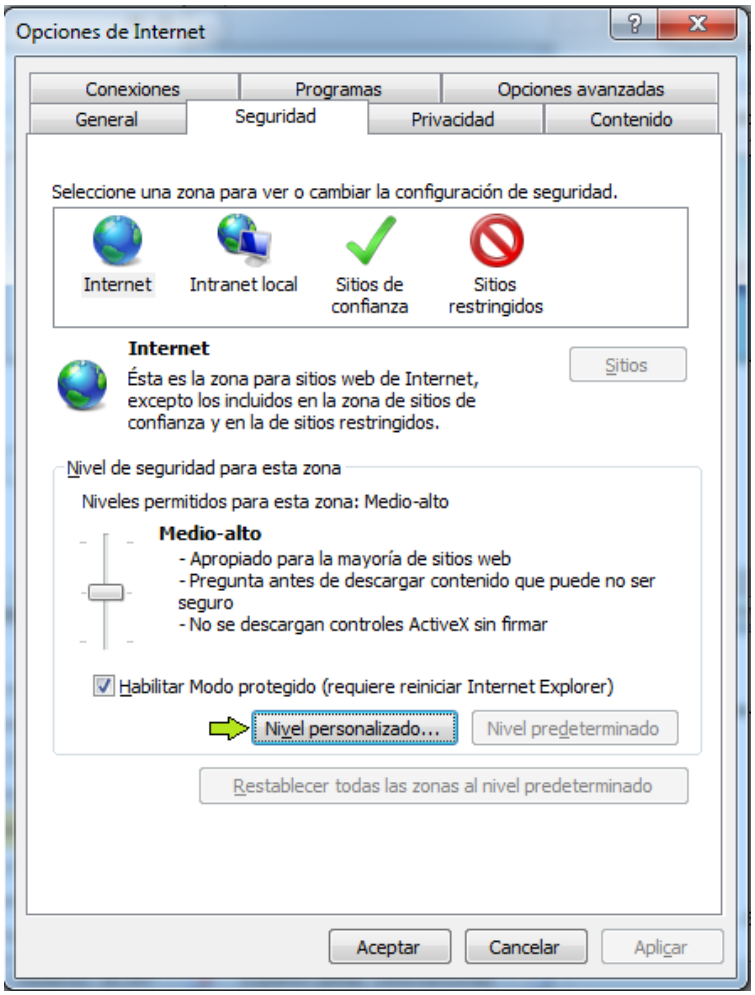

Figura 112. Seguridad > Nivel Personalizado

3. Una vez situados sobre la sección **Controles y complementos *ActiveX*** , configurar las opciones mostradas a continuación de la siguiente manera:

   - **Comportamiento de binarios y scripts**: Habilitar
   - **Descargar los controles *ActiveX* firmados**: Preguntar
   - **Ejecutar controles y complementos de *ActiveX***: Habilitar
   - **Generar *scripts* de los controles *ActiveX* marcados como seguros**: Habilitar

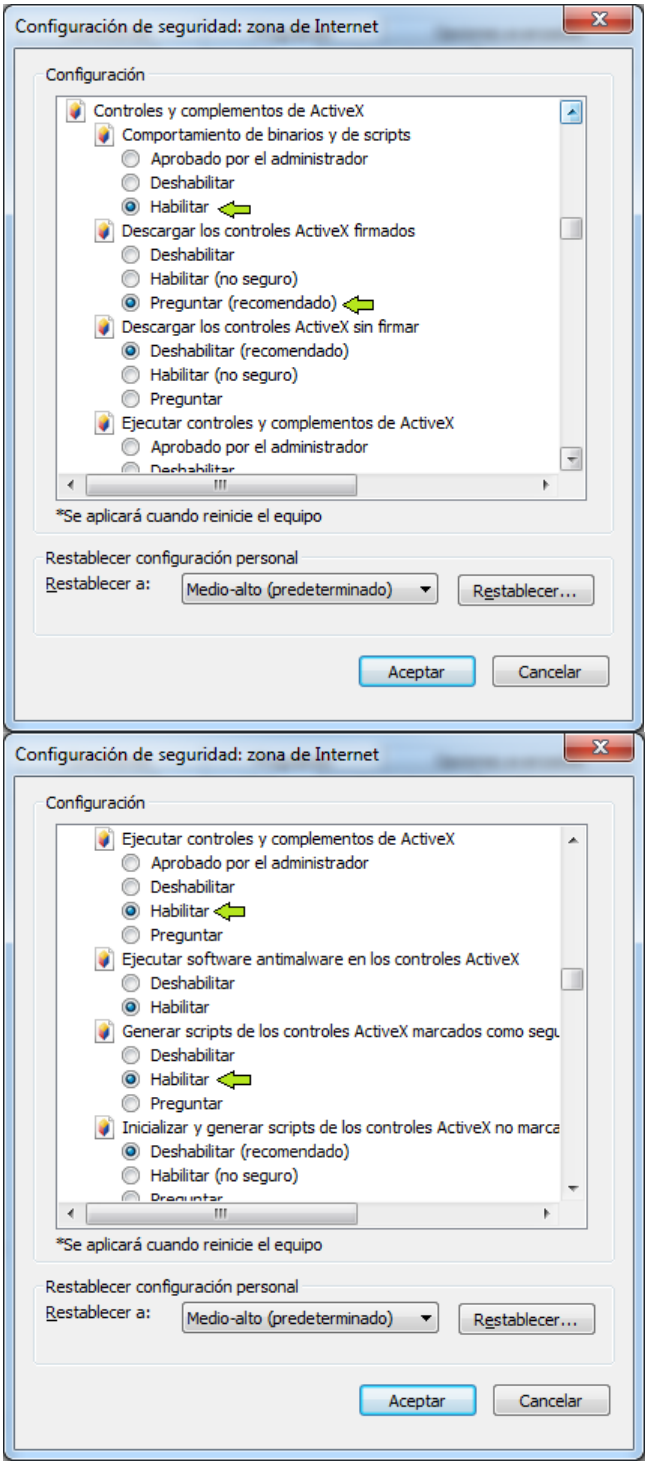

Figura 113. Configuración de las opciones de seguridad

4. Pulsar sobre el botón ***Aceptar*.**

7.1.2. JavaScript

El usuario de la aplicación tiene que habilitar la funcionalidad de JavaScript en la configuración del navegador.

1. Repetimos los pasos 1 y 2 explicados en el punto anterior.
2. Una vez situados sobre la sección **Automatización**, configurar las opciones mostradas en la figura 114:
    - ***Active Scripting***: Habilitar

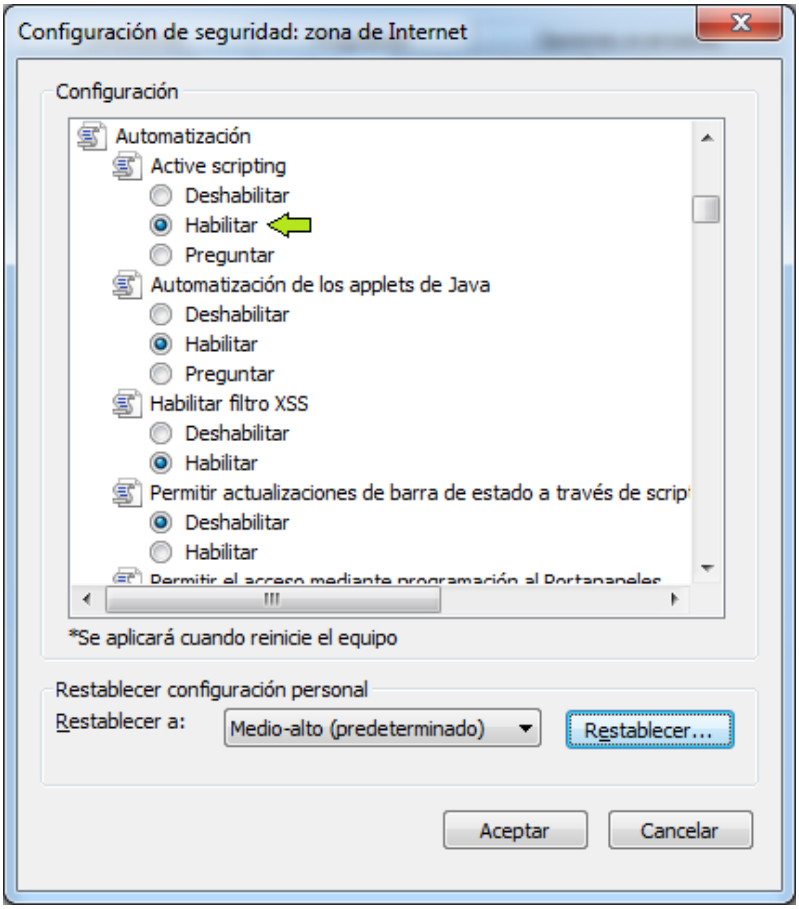

Figura 114. Configuración opciones JavaScript

3. Pulsar el botón ***Aceptar*.**

7.2.Consultas y posibles resultados

Lo primero que el usuario tiene que conocer es el número de tarea asociada al encaminador del que tiene que generar la configuración ya que en la empresa donde se va a implantar este sistema las instalaciones realizadas en los clientes están identificadas por un identificador. Este es el que se va a escribir en la caja de texto ofrecida por la web y, tras esto, se va a pulsar el botón *Buscar*. En la figura 115 se observa la pantalla inicial con el dato introducido:

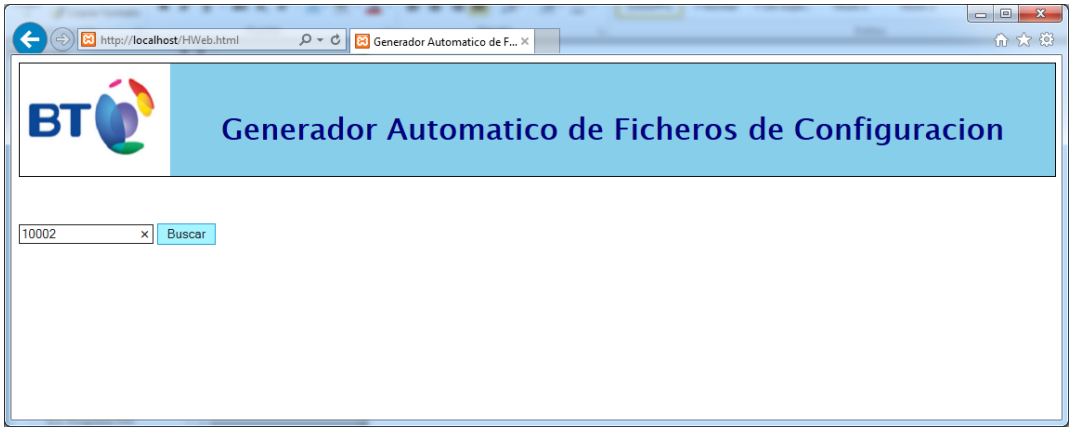

Figura 115. Consulta de datos a partir del identificador de tarea

Mientras la página espera la respuesta del servidor, se muestra una cadena de texto cuyo contenido es "*Cargando*", para que el usuario sepa que la página está funcionando y no existe ningún problema, lo cual se puede ver en la figura 116:

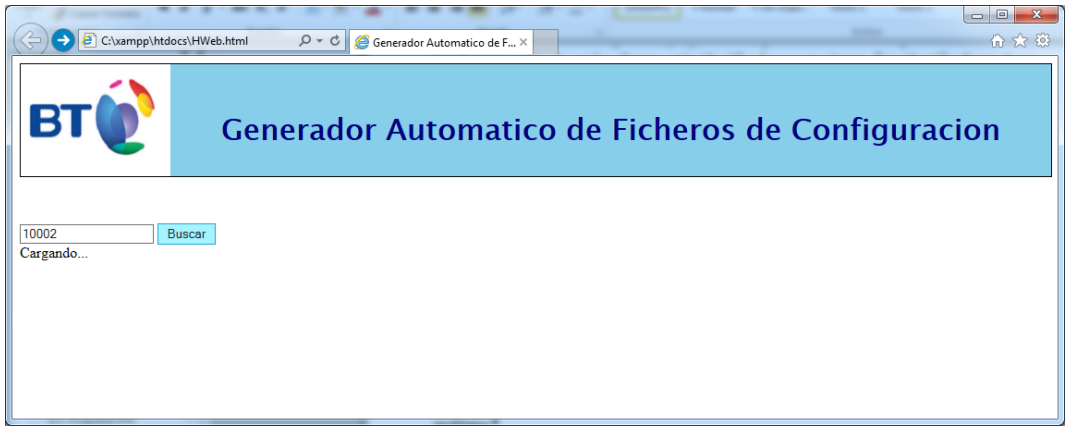

Figura 116. Mensaje de espera de la respuesta del servidor

En caso de que la conexión con la base de datos no pueda realizarse por un problema en la dirección del servidor, en el nombre de usuario, en la contraseña o en el propio nombre de la base de datos, se le notificará al usuario el fallo de la conexión, el código de error y el motivo, tal y como se ilustra en las figuras 117, 118 y 119:

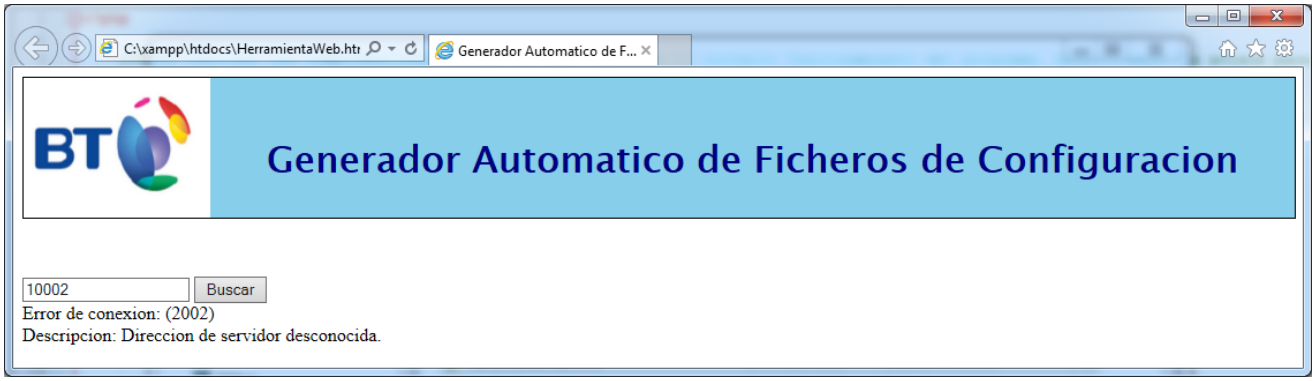

Figura 117. Error de conexión 2002. Dirección de servidor desconocida

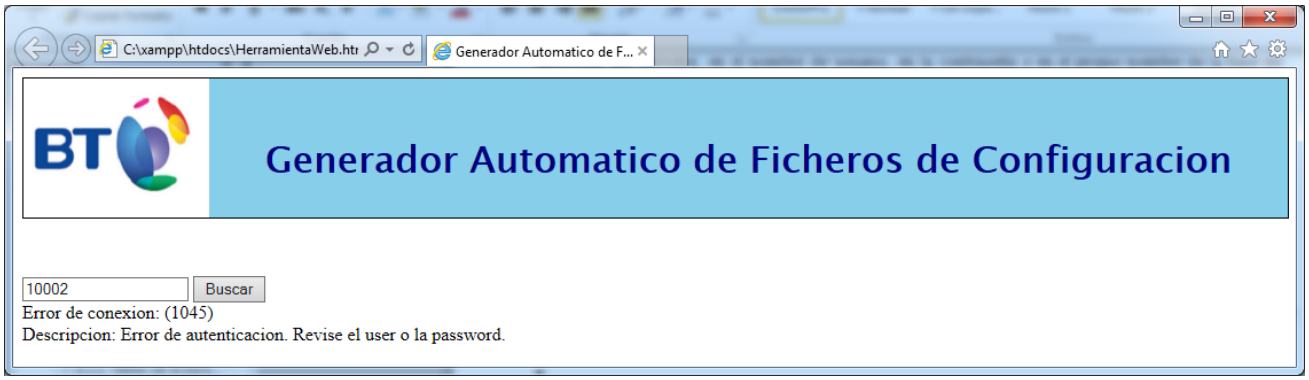

Figura 118. Error de conexión 1045. Error de autenticación por usuario o contraseña erróneos

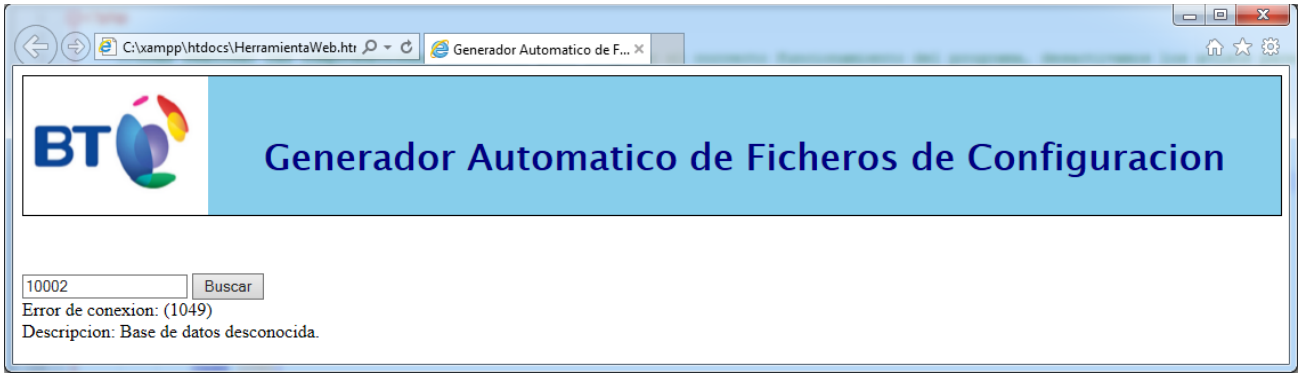

Figura 119. Error de conexión 1049. Base de datos desconocida

En caso de que la conexión sea satisfactoria y tras recibir la respuesta del servidor, pueden darse tres situaciones. Si la búsqueda arroja resultados, el usuario debe observar la tabla construida a partir de los mismos. Sin embargo, si el usuario está buscando un identificador de tarea que no existe en la base de datos o bien la tarea existe pero no se corresponde con un alta de servicios, entonces se debe notificar al usuario la causa por la que no se muestra la tabla de datos. Este comportamiento es el representado en las figuras 120, 121 y 122.

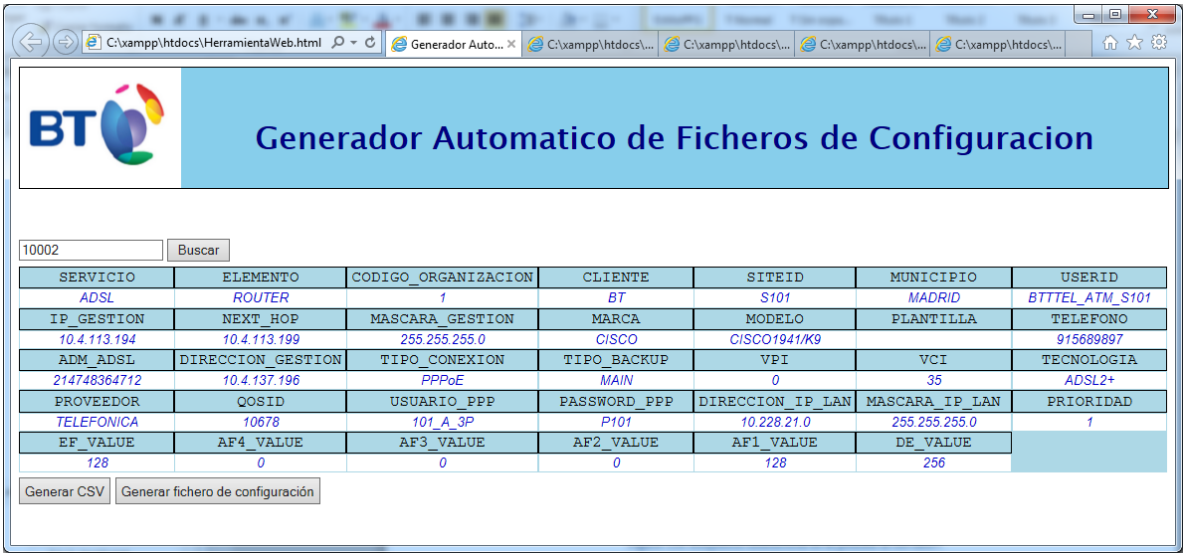

Figura 120. Respuesta satisfactoria de la petición al servidor

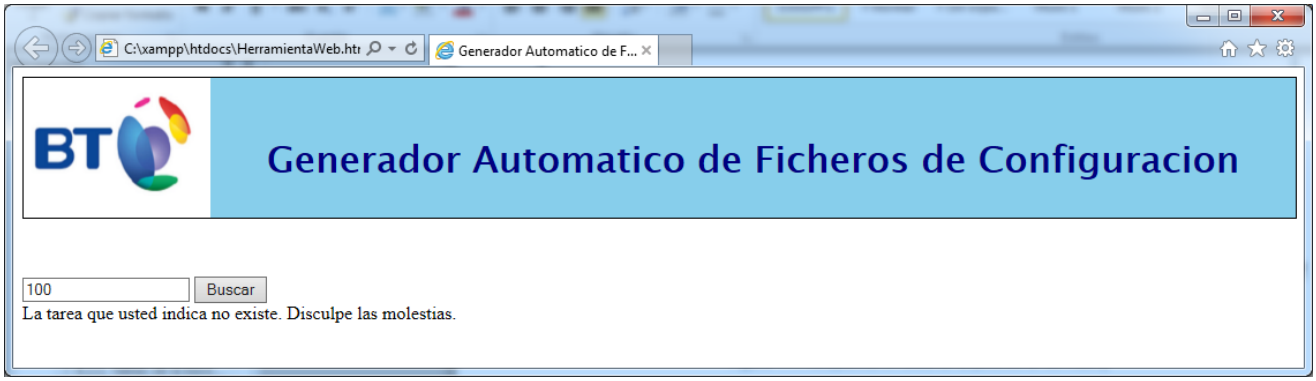

Figura 121. Respuesta ofrecida tras buscar una tarea inexistente

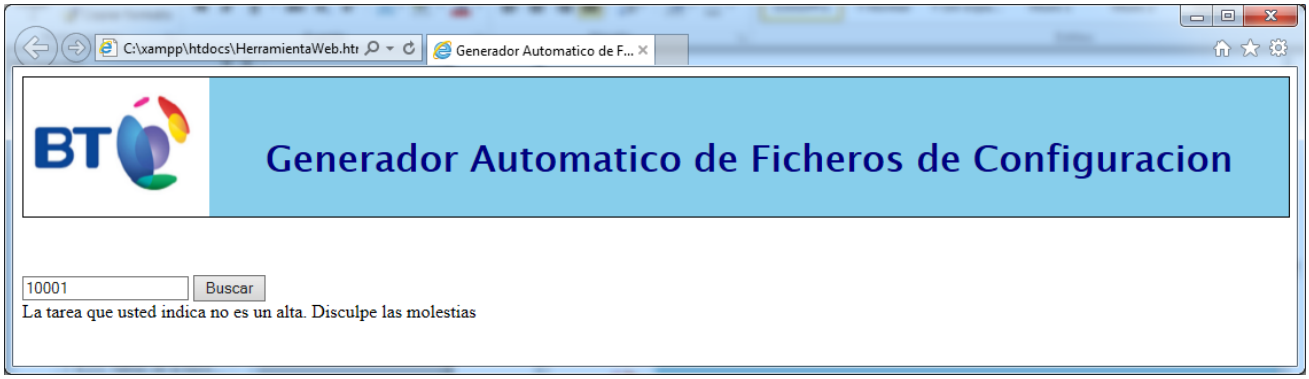

Figura 122. Respuesta ofrecida tras buscar una tarea distinta a un alta

En caso de que el usuario olvide introducir el identificador de la tarea, se mostrará un mensaje como el de la figura 123 indicando el error y no se realizará ninguna acción.

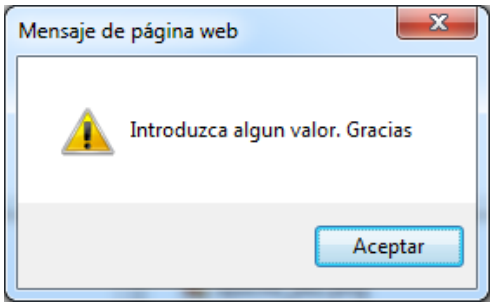

Figura 123. Ventana de notificación de error por búsqueda vacía

7.3.Modificación de los datos de la tabla

El motivo por el que, en lugar de ofrecer los datos obtenidos de la consulta en modo lectura, tal y como se hace con el nombre de los campos, se hace como contenido de cajas de texto insertadas en las celdas es porque se quiere ofrecer al usuario la opción de modificarlos. Esto es así porque muchos de los datos obtenidos en el entorno real pueden contener errores, los cuales, o son detectados previamente por el usuario o son detectados durante la validación de los datos realizada por la aplicación. Sea como sea, se deben modificar para generar un fichero de texto que contenga una configuración válida. Para modificarlos, bastará con situarse en la caja de texto que contiene el dato que se desea modificar y reescribirlo. En este caso, representado en la figura 124, se ha escrito la palabra "Modificación" en el campo "ELEMENTO".

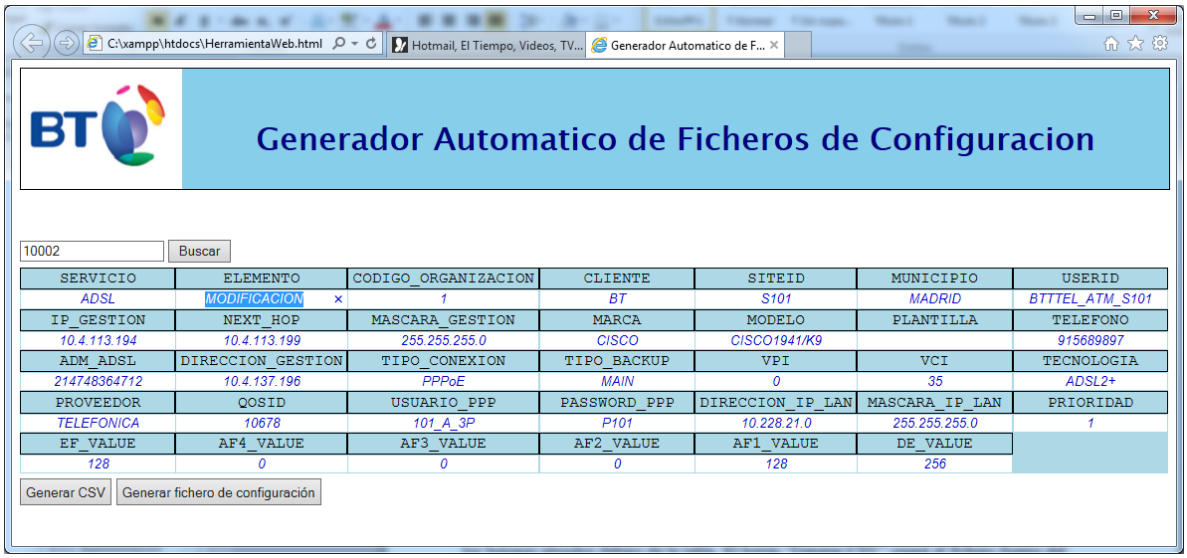

Figura 124. Modificación de los datos de la tabla

Tras asegurarse de que los datos vistos por pantalla son los que el usuario desea utilizar para la generación de la configuración del encaminador, se debe pulsar sobre uno de los botones situados debajo de la tabla. El botón *Generar CSV* creará el fichero fuente del que necesita leer la aplicación, mientras que el botón *Generar fichero de configuración* creará el fichero de texto que contiene la configuración del encaminador.

7.4.Generar CSV

Tras presionar este botón se genera el fichero con extensión CSV a partir de los datos leídos de la tabla. Tal y como ya se ha explicado anteriormente, este fichero es la fuente de la que la aplicación desarrollada en Java lee los datos, de manera que sin la existencia de este, no se puede generar el fichero que contiene la configuración. Tras dos segundos de espera en los que se da tiempo a que el fichero se cree en el directorio adecuado, se comprueba la existencia del mismo. Si la creación del fichero es satisfactoria, se comunica al usuario que el fichero se ha generado correctamente y se abre el directorio que contiene este tipo de ficheros. Estas operaciones se pueden observar en las figuras 125 y 126:

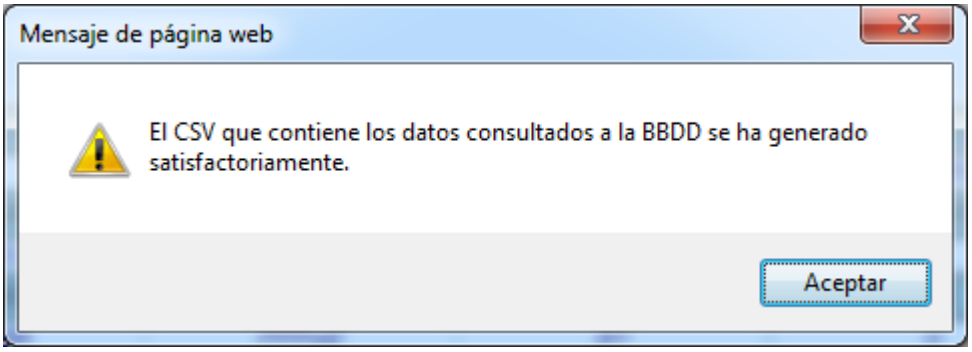

Figura 125. Mensaje satisfactorio de creación del fichero CSV

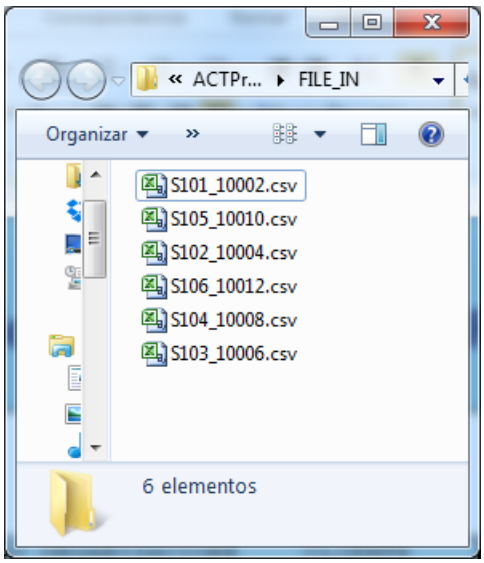

Figura 126. Directorio en el que se ubican los ficheros CSV

Por el contrario, en caso que se produzca algún error, se notifica al usuario la imposibilidad de crear el fichero mediante la ventana de la figura 127:

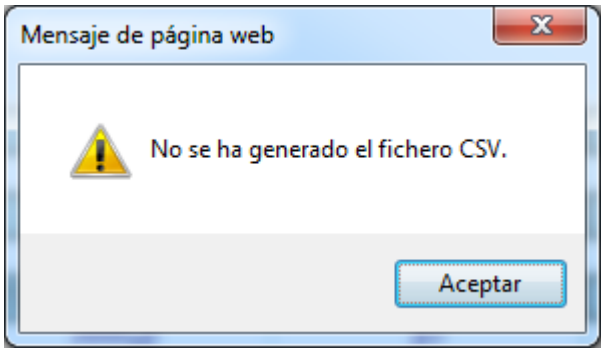

Figura 127. Mensaje de error en la creación del fichero CSV

7.5.Generar fichero de configuración

La generación del fichero de configuración se lleva a cabo mediante la aplicación Java. Por eso, una vez que se selecciona esta opción lo que se hace es, en primer lugar, generar el fichero fuente con los datos obtenidos de la tabla presentada en la web. En segundo lugar, se invoca la el programa desarrollado en Java, el cual se encarga de realizar todas las operaciones necesarias para generar el fichero de texto con la configuración que debe ser volcada al encaminador correspondiente. En caso de que todo el proceso se desarrolle con normalidad, el usuario recibirá un mensaje por pantalla que le indicará que el fichero se ha generado satisfactoriamente, el cual se puede observar en la figura 128, y además se abrirá dicho fichero, ilustrado en la figura 129:

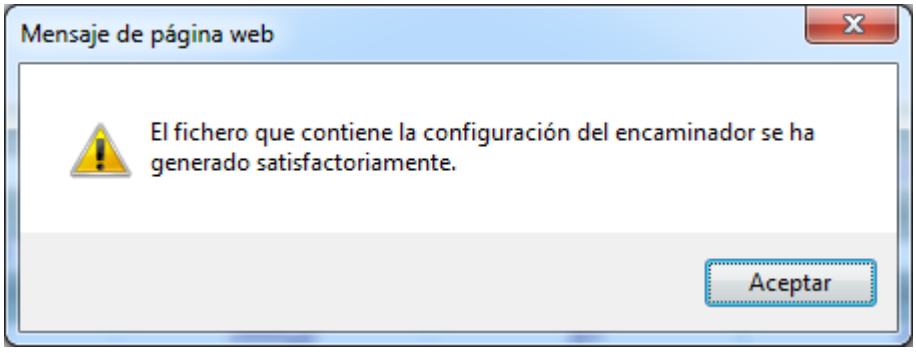

Figura 128. Mensaje de éxito en la generación del fichero de configuración

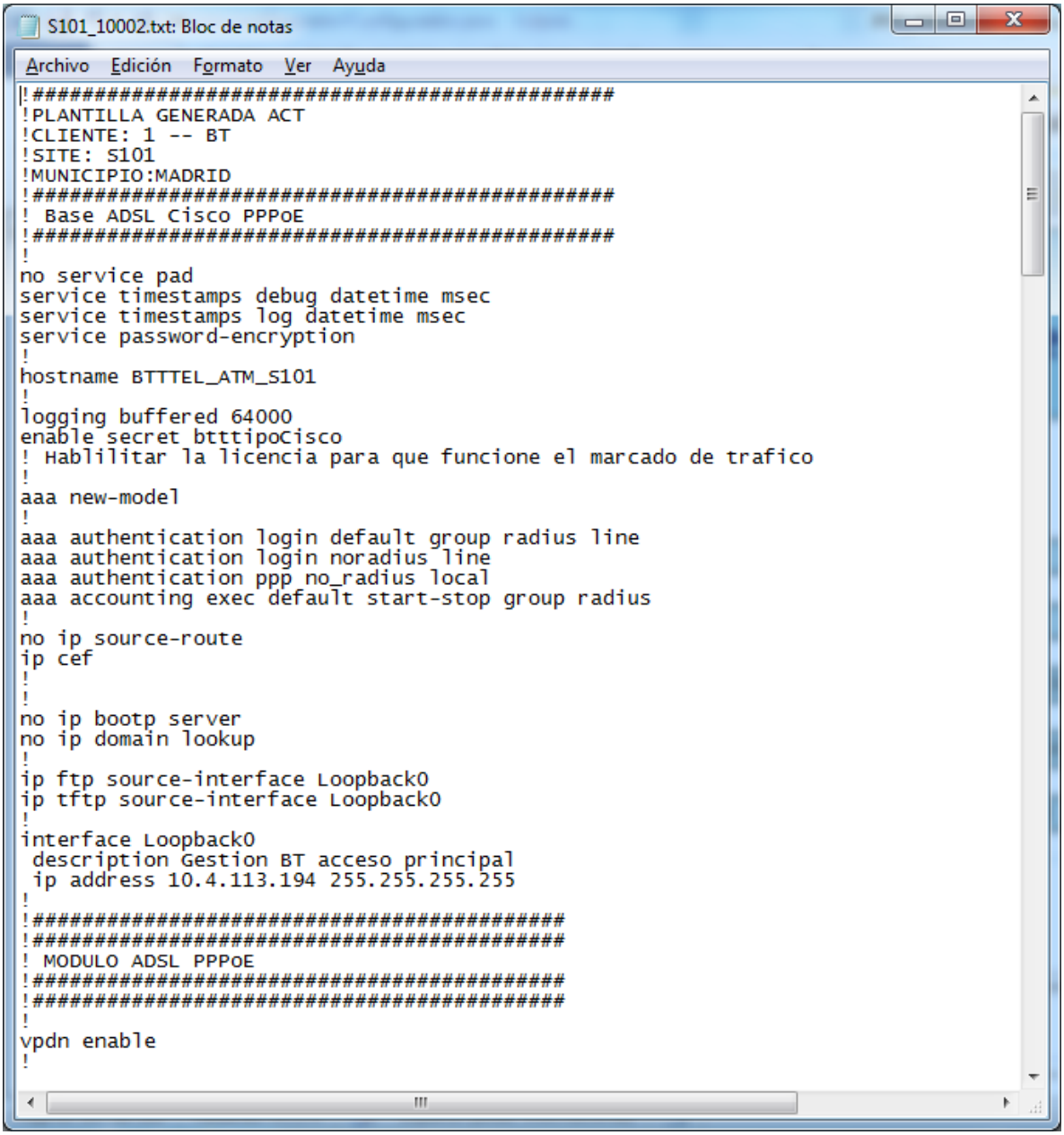

Figura 129. Fichero de texto generado

En caso de que se produzca algún problema durante la ejecución de la aplicación, se notificará al usuario no sólo de que se ha producido un error, sino que además se le abrirá el fichero creado que contiene qué error/errores se han producido. Para mostrar el funcionamiento mencionado, se han modificado en la tabla dos parámetros: *SITEID* y *NEXT_HOP*. La modificación puede observarse en la figura 130:

Figura 130. Modificación de los campos SITEID y NEXT_HOP

El resultado obtenido tras las modificaciones hechas se muestra en las figuras 131 y 132: en la primera se puede observar la ventana emergente informando del error al usuario, mientras que en la segunda se puede ver un análisis del error o errores producidos durante la ejecución:

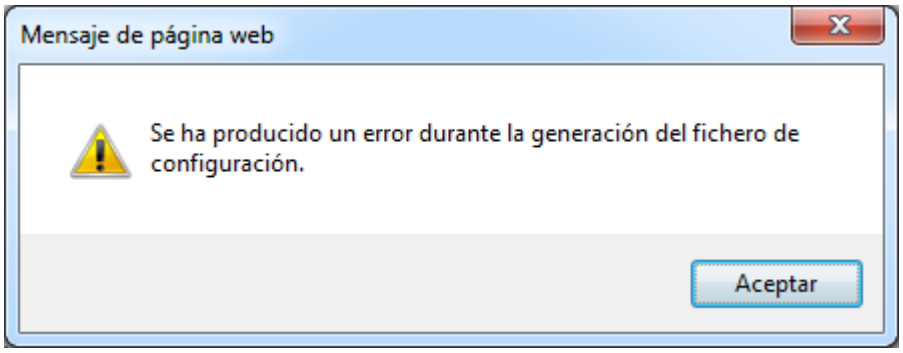

Figura 131. Mensaje de error en la creación del fichero de configuración

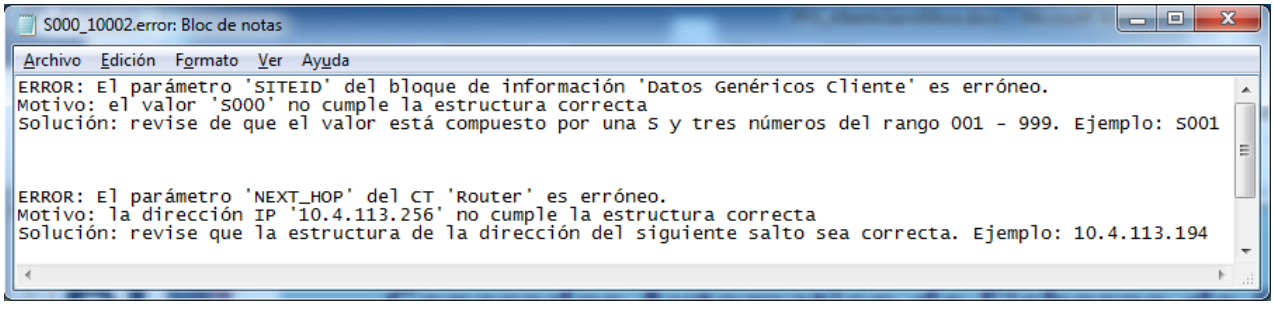

Figura 132. Contenido del fichero de error creado por la aplicación

CAPÍTULO 8

CONCLUSIONES Y TRABAJOS FUTUROS

8. Conclusiones y trabajos futuros

8.1.Conclusiones

El desarrollo de este PFG tenía como objetivo principal la implantación de un sistema que fuera capaz de resolver uno de los problemas con el que nos enfrentamos día a día en el departamento de entrega de la empresa: la configuración básica de encaminadores. Para ello, era necesario desarrollar una aplicación que fuera capaz de crear ficheros de texto que contuvieran dicha configuración para que el instalador sólo tuviera que volcarla sobre el equipo. Además, era necesario crear una interfaz gráfica sencilla para que el usuario consiguiera rápidamente los resultados deseados.

Tras realizar un estudio de la situación tanto con el tutor académico como con varios compañeros del departamento se decidió dividir el sistema en tres grandes bloques. Por un lado, tendríamos la herramienta web a la que accederían los miembros del departamento, por otro la aplicación desarrollada en Java que tuviera toda la lógica necesaria para generar el fichero de configuración y, por último, un servidor en el que se crearía una base de datos y un programa en lenguaje PHP que sirvieran para simular la situación real.

Para la herramienta web, el objetivo era buscar la máxima simplicidad posible no sólo desde el punto de vista del formato de presentación de la misma, sino también desde el punto de vista de diseño. Como consecuencia de esto, el diseño se dividió a su vez en tres módulos: el esqueleto, realizado en lenguaje HTML; el formato de los elementos, en CSS; y las funciones para ofrecer interacción entre el usuario y la web, en JavaScript. Este último módulo es que el que más problemas podría plantear puesto que debía tener tanto la lógica necesaria para tratar los datos devueltos tras realizar la petición de datos al servidor como las funcionalidades necesarias para ofrecer una buena experiencia de usuario. Además, utilizando este lenguaje se asume el agujero de seguridad que supone que cualquier persona que tenga acceso a la web pueda ver el código fuente. Pese a esto, JavaScript era la mejor opción por dos motivos. En primer lugar, en el código utilizado no se revela información confidencial, además de que nadie ajeno al departamento podría acceder a él. En segundo lugar, era un lenguaje conocido por algunos de los miembros del departamento, que serán los encargados de mantener la herramienta.

En cuanto a la aplicación Java, el objetivo era que fuese lo más sencilla posible puesto que, al ser una herramienta cuyo desarrollo acaba de comenzar y su funcionalidad va a crecer en base a los objetivos fijados por los miembros del departamento, es ideal que cualquiera pueda realizar cambios sin hacer grandes modificaciones en el código de la misma. La colaboración de algunos miembros del departamento ha sido necesaria para comprender aspectos técnicos que se debían incluir en el código con el objetivo de generar una configuración válida. Para la validación sintáctica de los datos se ha decidido utilizar la tecnología XML. Para ello, se ha definido el esquema que debe tener el documento creado a partir de los datos obtenidos de la tabla mostrada al usuario en la página web. La capacidad que nos ofrecen los esquemas XML para la definición de tipos de datos, tanto simples como complejos, hacen de esta una herramienta muy potente.

Además tiene la propiedad de ser escalable puesto que, en caso de incluir nuevos elementos, los cambios a realizar son puntuales y sencillos. Para la sustitución de los parámetros de las plantillas de configuración se ha decidido utilizar la interfaz *map* porque nos ofrece la funcionalidad exacta que se necesita para llevar a cabo la modificación dinámica de cada línea, que no es otra que la identificación del valor de un parámetro mediante una clave. En este caso, el nombre de la variable a modificar ha actuado a modo de clave. De esta manera, no importa el número de parámetros a modificar, el orden de los mismos o si pertenecen a uno u otro componente técnico. Sólo importa que se identifiquen con el mismo nombre tanto en el fichero modelo como en la plantilla de configuración. La consecuencia de esto es que se podrán añadir o quitar tantos parámetros como se deseen fácilmente.

Por último, era necesario simular la parte servidora del sistema puesto que en el entorno real, tanto la base de datos como el PHP utilizado para realizar las consultas lo gestiona otro departamento. Para ello, se ha utilizado la aplicación XAMPP, que permite una sencilla instalación de Apache, *MySQL* y PHP entre otros módulos. Para el desarrollo de la base de datos se ha utilizado la herramienta *phpMyAdmin*, proporcionada por dicha aplicación, que ofrece un interfaz sencillo para la creación y la inserción de datos en las tablas de la misma. Además, el objetivo ha sido representar lo más fielmente posible la estructura de la base de datos real, obviando, evidentemente, mucha información irrelevante para este PFG. En cuanto al programa que gestiona las peticiones realizadas desde el cliente, la lógica del mismo se ha desarrollado en PHP mientras que las consultas a la base de datos se han escrito en código SQL. La respuesta del servidor se ha diseñado de manera que la información obtenida se muestre de la misma manera que en el entorno laboral.

El desarrollo de este proyecto ha sido de gran utilidad puesto que no sólo se han puesto en práctica conceptos aprendidos durante el grado, como por ejemplo la programación en Java, el diseño de un documento XML o la validación de este contra un esquema, sino que se han adquirido nuevos conocimientos como la programación web en lenguaje tanto del lado del cliente como del servidor o lenguaje SQL para realizar consultas sobre administradores de bases de datos relacionales, como puede ser MySQL. El aprendizaje de estas nuevas tecnologías ha supuesto, al mismo tiempo, un desafío y una satisfacción. Desafío, porque no se tenían conocimientos previos, lo que ha hecho más duro el desarrollo del proyecto, y satisfacción, porque se ha comprobado que los conceptos teóricos y prácticos aprendidos durante el grado son útiles a la hora de afrontar nuevos retos, es decir, tras estos años de formación se han adquirido una serie de herramientas que permiten analizar, comprender y finalmente resolver problemas.

Personalmente, el aprendizaje de PHP y de SQL me ha permitido colaborar activamente en varios proyectos realizados en paralelo a este en la empresa.

Además, gracias al proceso de generación de los ficheros que contienen la configuración de los encaminadores se han adquirido algunas nociones que pueden ser la base de un nuevo aprendizaje.

Una vez expuestas las conclusiones, se puede pueden dar por cumplidos tanto los objetivos técnicos propuestos en este documento como los objetivos de formación y aplicación de conceptos teórico-prácticos asociados a la realización de un PFG.

8.2.Trabajos futuros

En este apartado se proponen diferentes líneas de trabajo con el objetivo de mejorar y aumentar las funcionalidades del mismo:

- Validar semánticamente determinados datos del fichero de texto que contiene la configuración del encaminador. Ejemplo: si el parámetro *siteid* que identifica la localización del cliente es "S101", se debe comprobar que este valor esté contenido en el parámetro *userid.*
- Desarrollar las funcionalidades de la página web en lenguaje PHP con el objetivo de que nadie tenga acceso al código que se encarga del tratamiento de los datos obtenidos tras la petición.
- Modificar el código de la web de manera que esta esté soportada por cualquier navegador, como por ejemplo Firefox o Chrome.
- Desarrollar un módulo que sea capaz de volcar directamente la configuración generada sobre el encaminador en cuestión, sin la necesidad de que un instalador tenga que realizar este trabajo manualmente.
- Eliminar la opción que permite la generación del fichero CSV, puesto que cuando se consiga que el departamento responsable nos ofrezca la respuesta de la petición, el CSV será innecesario. En este caso, habría que modificar el código de la aplicación, concretamente lo relacionado con las validaciones de los datos.
- Ampliar la funcionalidad de la herramienta web, de manera que se puedan realizar configuraciones más complejas permitiendo al usuario elegir qué configurar.
- En caso de detectar algún parámetro erróneo de la base de datos, sería ideal poder realizar el cambio desde la propia web. Sin embargo, para el desarrollo de esta funcionalidad sería necesario negociar aspectos de seguridad puesto que cualquier usuario podría realizar modificaciones.

REFERENCIAS

Referencias

[1] M. Á. Álvarez, «desarrolloweb.com,» 4 diciembre 2002. [En línea]. Disponible: http://www.desarrolloweb.com/articulos/993.php. [Último acceso: 18 marzo 2015].

[2] Real Academia de Ingeniería, «Diccionario Español de Ingeniería,» [En línea]. Disponible: http://diccionario.raing.es. [Último acceso: marzo 2015].

[3] Real Academia de Ingeniería, «Diccionario Español de Ingeniería,» [En línea]. Disponible: http://diccionario.raing.es/. [Último acceso: marzo 2015].

[4] Real Academia de Ingeniería, «Diccionario Español de Ingeniería,» [En línea]. Disponible: http://diccionario.raing.es/. [Último acceso: marzo 2015].

[5] Real Academia de Ingeniería, «Diccionario Español de Ingeniería,» [En línea]. Disponible: http://diccionario.raing.es. [Último acceso: marzo 2015].

[6] Real Academia Española, «Real Academia Española,» [En línea]. Disponible: http://lema.rae.es/drae/?val=software. [Último acceso: abril 2015].

[7] Instituto Nacional de Estadística, «Notas de Prensa,» 19 septiembre 2014. [En línea]. Disponible: http://www.ine.es/prensa/np859.pdf. [Último acceso: 7 enero 2015].

[8] INE. *Tipo de conexión utilizada por las empresas españolas según número de empleados* [imagen en línea]. Disponible: http://www.ine.es/prensa/np859.pdf. [Último acceso: 7 enero 2015].

[9] «ADSLZONE,» [En línea]. Disponible: http://www.adslzone.net/adsl-faq.html. [Último acceso: 7 enero 2015].

[10] M. Á. Pérez, «Think Big,» 9 junio 2014. [En línea]. Disponible: http://blogthinkbig.com/en-que-se-diferencia-el-vdsl-del-adsl/. [Último acceso: 8 enero 2015].

[11] ADSL Internet, *Tabla comparativa de las velocidades ofrecidas por ADSL y VDSL* [imagen en línea. Disponible: http://adsl-internet.com/conexiones-a-internet/vdsl/que-es-el-vdsl/ [Último acceso: 8 enero 2015].

[12] F. Huari, «SISBIB,» abril 2001. [En línea]. Disponible: http://sisbib.unmsm.edu.pe/bibvirtual/publicaciones/indata/v04_n1/tecnologia.htm. [Último acceso: 10 enero 2015].

[13] F. R. Vivanco, «FelipeReyesVivanco,» [En línea]. Disponible: http://www.felipereyesvivanco.com/redes-cableadas/redes-hfc/. [Último acceso: 12 enero 2015].

[14] VIVANCO, F. *Topología de red HFC* [imagen en línea]. Disponible: http://www.felipereyesvivanco.com/redes-cableadas/redes-hfc/ [Último acceso: 12 enero 2015].

[15] Tech Target, «SearchNetworking,» [En línea]. Disponible: http://searchnetworking.techtarget.com/definition/DOCSIS. [Último acceso: 12 enero 2015].

[16] R. Cambre, «ElOtroLado,» 25 octubre 2013. [En línea]. Disponible: http://www.elotrolado.net/noticia_el-cable-acelera-hasta-los-5-000-megas-de-bajada-con-docsis-3-1_22627. [Último acceso: 13 enero 2015].

[17] «Telecomunicaciones - conocimientos.com.ve,» 11 diciembre 2009. [En línea]. Disponible: http://telecomunicaciones.conocimientos.com.ve/2009_12_11_archive.html. [Último acceso: 13 enero 2015].

[18] «YIO,» [En línea]. Disponible: http://www.yio.com.ar/fo/indiceref.html. [Último acceso: 14 febrero 2015].

[19] YIO. *Fórmula del índice de refracción* [imagen en línea]. Disponible: http://www.yio.com.ar/fo/indiceref.html [Último acceso: 14 febrero 2015].

[20] YIO. *Fórmula de la ley de Snell* [imagen en línea]. Disponible: http://www.yio.com.ar/fo/indiceref.html [Último acceso: 14 febrero 2015].

[21] Textos científicos, TEORÍA DE PROPAGACIÓN, 2005. *Representación de la ley de Snell* [imagen en línea]. Disponible: http://www.textoscientificos.com/redes/fibraoptica/propagacion., [Último acceso: 16 febrero 2015]..

[22] Textos científicos, TEORÍA DE PROPAGACIÓN, 2005. *Ángulo crítico* [imagen en línea]. Disponible: http://www.textoscientificos.com/redes/fibraoptica/propagacion, [Último acceso: 16 febrero 2015].

[23] Textos científicos, TEORÍA DE PROPAGACIÓN, 2005. *Morfología de la fibra óptica* [imagen en línea]. Disponible: http://www.textoscientificos.com/redes/fibraoptica/propagacion, [Último acceso: 16 febrero 2015].

[24] Gycom, «Gycom,» [En línea]. Disponible: http://www.fibraoptica.com/informacion-tecnica/vistazo-tecnologia. [Último acceso: 15 enero 2015].

[25] Gycom. *Propagación de la luz en el interior de la fibra* [imagen en línea]. Disponible: http://www.fibraoptica.com/informacion-tecnica/vistazo-tecnologia [Último acceso: 15 enero 2015]..

[26] D. Valero, «ADSLZONE,» 31 julio 2014. [En línea]. Disponible: http://www.adslzone.net/2014/07/31/asi-esta-la-cobertura-de-fibra-optica-hasta-el-hogar-ftth-en-espana/. [Último acceso: 15 enero 2015].

[27] C. Valero, «ADSLZONE,» 14 enero 2015. [En línea]. Disponible: http://www.adslzone.net/2015/01/14/cobertura-y-despliegue-de-fibra-optica-en-2015-estos-son-los-planes-de-las-operadoras/. [Último acceso: 19 enero 2015].

[28] A. Crespo, «RedesZone,» 16 agosto 2011. [En línea]. Disponible: http://www.redeszone.net/2011/08/16/topologia-de-las-redes-de-fibra-optica-fttx/. [Último acceso: 19 enero 2015].

[29] CRESPO, A., 2001. *FTTH: fibra desde la central hasta el domicilio o negocio* [imagen en línea]. Disponible: http://www.redeszone.net/2011/08/16/topologia-de-las-redes-de-fibra-optica-fttx/ [Último acceso: 19 enero 2015].

[30] CRESPO, A., 2001. *FTTB: fibra desde la central hasta un nodo de distribución situado en el edificio o en las inmediaciones* [imagen en línea]. Disponible: http://www.redeszone.net/2011/08/16/topologia-de-las-redes-de-fibra-optica-fttx/ [Último acceso: 19 enero 2015].

[31] CRESPO, A., 2001. *FTTN: fibra desde la central hasta un nodo situado en las inmediaciones de un conjunto de edificios* [imagen en línea]. Disponible: http://www.redeszone.net/2011/08/16/topologia-de-las-redes-de-fibra-optica-fttx/ [Último acceso: 19 enero 2015].

[32] T. Martínez, «Telequismo,» 28 febrero 2013. [En línea]. Disponible: http://www.telequismo.com/2013/02/gpon-operador.html. [Último acceso: 21 enero 2015].

[33] MARTÍNEZ, T., 2013. *Estructura de una red GPON* [imagen en línea]. Disponible: http://www.telequismo.com/2013/02/gpon-operador.html [Último acceso: 21 enero 2015].

[34] GREGORIK, I. *Tipos de redes inalámbricas* [imagen en línea]. Disponible: http://chimera.labs.oreilly.com/books/1230000000545/ch05.html [Último acceso: 22 enero 2015].

[35] J.E AIRHEADS Community, 2014. *Características del estándar 802.11 (WiFi)* [imagen en línea]. Disponible: http://community.arubanetworks.com/t5/Unified-Wired-Wireless-Access/802-11-Standard-and-data-rate-increasing-Amendments/td-p/159172 [Último acceso: 25 enero 2015].

[36] M. González, «Redes Telemáticas,» 30 abril 2014. [En línea]. Disponible: http://redestelematicas.com/modos-de-funcionamiento-de-las-redes-wi-fi/. [Último acceso: 26 enero 2015].

[37] Kioskea, «Kioskea,» [En línea]. Disponible: http://es.kioskea.net/contents/791-modos-de-funcionamiento-wifi-802-11-o-wi-fi#infrastructure. [Último acceso: 2 febrero 2015].

[38] Kioskea. *ESS formada por la conexión entre dos o más celdas (BSS)* [imagen en línea]. Disponible: http://es.kioskea.net/contents/791-modos-de-funcionamiento-wifi-802-11-o-wi-fi#infrastructure [Último acceso: 2 febrero 2015].

[39] GONZALEZ, M., 2014. *Conexión del AP con el encaminador para ofrecer acceso a Internet* [imagen en línea]. Disponible: http://redestelematicas.com/modos-de-funcionamiento-de-las-redes-wi-fi/ [Último acceso: 26 enero 2015].

[40] M. Brain y E. Grabianowski, «HowStuffWorks,» [En línea]. Disponible: http://computer.howstuffworks.com/wimax1.htm. [Último acceso: 3 febrero 2015].

[41] M. Brain y E. Grabianowski. *Funcionamiento de la red WiMAX* [imagen en línea]. Disponible: http://computer.howstuffworks.com/wimax1.htm [Último acceso: 3 febrero 2015].

[42] C. R. Nespereira, *Redes de Comunicaciones Móviles*, Universidad Politécnica de Madrid, 2013.

[43] Windows, «¿Qué es la banda ancha móvil?,» [En línea]. Disponible: http://windows.microsoft.com/es-es/windows7/what-is-mobile-broadband. [Último acceso: 5 marzo 2015].

[44] «Internet-Satélite,» 16 octubre 2014. [En línea]. Disponible: http://www.internet-satelite.eu/noticias/espanoles-sin-cobertura-internet-banda-de-ancha/. [Último acceso: 4 febrero 2015].

[45] Computer Networking Simplified, «Computer Networking Simplified,» [En línea]. Disponible: http://computernetworkingsimplified.com/physical-layer/relationship-bandwidth-data-rate-channel-capacity/. [Último acceso: 4 febrero 2015].

[46] Computer Networking Simplified. *Velocidad máxima de transmisión para un canal sin ruido* [imagen en línea]. Disponible: http://computernetworkingsimplified.com/physical-layer/relationship-bandwidth-data-rate-channel-capacity/ [Último acceso: 4 febrero 2015].

[47] SHAH, E. *Fórmula del ruido en la comunicación* [imagen en línea]. Disponible: http://www.engineersblogsite.com/bursted-data-rate-limits.html [Último acceso: 5 febrero 2015].

[48] Computer Networking Simplified. *Velocidad máxima de transmisión para un canal sin ruido* [imagen en línea]. Disponible: http://computernetworkingsimplified.com/physical-layer/relationship-bandwidth-data-rate-channel-capacity/, [Último acceso: 4 febrero 2015].

[49] Colegio Oficial de Ingenieros de Telecomunicación, «COIT,» [En línea]. Disponible: https://www.coit.es/descargar.php?idfichero=974. [Último acceso: 7 febrero 2015].

[50] Universitat Politécnica de Catalunya, «http://ocw.upc.edu/,» [En línea]. Disponible: http://ocw.upc.edu/sites/default/files/materials/15011812/19918-2647.pdf. [Último acceso: 8 febrero 2015].

[51] Universidad Politécnica de Catalunya. *Acceso satelital unidireccional* [imagen en línea]. Disponible: http://ocw.upc.edu/sites/default/files/materials/15011812/19918-2647.pdf, [Último acceso: 8 febrero 2015].

[52] Universidad Politécnica de Catalunya. *Acceso satelital bidireccional* [imagen en línea]. Disponible: http://ocw.upc.edu/sites/default/files/materials/15011812/19918-2647.pdf [Último acceso: 8 febrero 2015].

[53] PCActual, «PCActual,» [En línea]. Disponible: http://www.pcactual.com/articulo/laboratorio/especiales/7415/dispositivos_plc_los_bits_viajan_por_red_electrica.html. [Último acceso: 8 febrero 2015].

[54] HomePlug-Alliance. *Funcionamiento del PLC* [imagen en línea]. Disponible: http://www.homeplug.org/explore-homeplug/overview/ [Último acceso: 8 febrero 2015].

[55] A. d. l. Fuente, «Blyx,» 2 octubre 2012. [En línea]. Disponible: http://blyx.com/public/docs/pila_OSI.pdf. [Último acceso: 11 febrero 2015].

[56] MikrotikXperts. *Niveles de red del modelo OSI* [imagen en línea]. Disponible: http://mikrotikxperts.com/index.php/2013-03-28-19-49-36/conocimientos-basicos/159-modelo-osi-y-tcp-ip [Último acceso: 10 febrero 2015].

[57] McGraw-Hill, «McGraw-Hill» [En línea]. Disponible: http://www.mcgraw-hill.es/bcv/guide/capitulo/8448199766.pdf. [Último acceso: 16 febrero 2015].

[58] Textos Científicos, 2006. *Relación entre capas del modelo OSI y del modelo TCP-IP* [imagen en línea]. Disponible: http://www.textoscientificos.com/redes/tcp-ip/comparacion-modelo-osi [Último acceso: 16 febrero 2015].

[59] Escuela Politécnica Superior, UAM. *Formato de un datagrama IP* [imagen en línea]. Disponible: http://arantxa.ii.uam.es/~rc2lab/practicas.html/prac2/index.html [Último acceso: 17 febrero 2015].

[60] G4 Communications, «G4 Communications,» [En línea]. Disponible: http://www.g4.net/docs/G4_RoutingBasics_1.pdf. [Último acceso: 18 febrero 2015].

[61] PALACIOS, A., 2009. *Ejemplo de tabla de enrutamiento* [imagen en línea]. Disponible: https://apalacioss.wordpress.com/page/4 [Último acceso: 12 marzo 2015].

[62] «Redes y Seguridad,» 27 enero 2009. [En línea]. Disponible: http://www.redesyseguridad.es/enrutamiento/. [Último acceso: 19 febrero 2015].

[63] M. González, «RedesTelemáticas,» 8 noviembre 2013. [En línea]. Disponible: http://redestelematicas.com/el-switch-como-funciona-y-sus-principales-caracteristicas/. [Último acceso: 23 febrero 2015].

[64] Cisco. *Trama Ethernet* [imagen en línea]. Disponible: http://www.cisco.com/en/US/docs/internetworking/troubleshooting/guide/tr1904.html#wp1020861 [Último acceso: 23 febrero 2015].

[65] Apache Friends, «XAMPP Apache + MySQL + PHP + Perl,» [En línea]. Disponible: https://www.apachefriends.org/es/index.html. [Último acceso: marzo 2015].

[66] M. Rouse, «TechTarget,» [En línea]. Disponible: http://searchsoa.techtarget.com/definition/HTML. [Último acceso: 4 marzo 2015].

[67] Libros Web, «LibrosWeb,» [En línea]. Disponible: http://librosweb.es/libro/css/. [Último acceso: marzo 2015].

[68] PHP, «PHP,» [En línea]. Disponible: http://php.net/manual/en/preface.php. [Último acceso: 5 marzo 2015].

[69] H. Sulbarán, «Sistemas de Información,» 7 mayo 2014. [En línea]. Disponible: http://helisulbaransistemas.blogspot.com.es/2014/05/motores-de-almacenamiento-de-mysql.html. [Último acceso: 6 marzo 2015].

[70] Definición De, «Definición De,» [En línea]. Disponible: http://definicion.de/java/. [Último acceso: 8 marzo 2015].

[71] Object Management Group, «Unified Modeling Language™ (UML®) Resource Page,» 22 febrero 2015. [En línea]. Disponible: http://www.uml.org/. [Último acceso: 10 marzo 2015].

[72] Real Academia Española, «Real Academia Española,» 2015. [En línea]. Disponible: http://lema.rae.es/drae/?val=codigo+postal. [Último acceso: 6 mayo 2015].

[73] PHP, «PHP,» 2015. [En línea]. Disponible: http://php.net/manual/es/security.database.sql-injection.php. [Último acceso: 5 mayo 2015].

[74] ss4, «ss4,» [En línea]. Disponible: http://ss64.com/nt/cmd.html. [Último acceso: marzo 2015].