Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Análisis de tráfico DNS mediante herramienta visual

Proyecto Final de Carrera

Ingeniería Técnica en Informática de Gestión

Autor: Jorge Manuel Pascual Llop

Director: Joaquín Gracia Moran

23 de septiembre de 2012

Resumen

Diseño e implementación en lenguaje C++ de una herramienta, que de forma gráfica, permita al usuario realizar consultas DNS, tanto de direcciones URL como de direcciones IP. La herramienta, una vez obtenga los datos necesarios para la consulta, podrá ejecutar las funciones necesarias del sistema para el tratamiento de los resultados.

La interfaz mostrará un resumen con los registros recibidos tras realizar la consulta, así como las respuestas obtenidas, indicando qué servidor nos las proporcionó, con lo cual se apreciará las partes que tiene una consulta DNS.

El entorno elegido para el desarrollo del proyecto es el Visual Studio 2005 bajo soporte de Windows XP.

Palabras clave: DNS, consulta, interfaz gráfico.

Tabla de contenidos

Contenido

1. Introducción 7
2. Justificación del problema 8
   2.1. Introducción 8
   2.2. Herramientas disponibles 9
   2.3. Otras opciones alternativas a nslookup 12
3. Protocolo DNS 13
   3.1. Definición 13
   3.2. Componentes principales 14
   3.3. Nombre de Dominio 15
   3.4. Tipos de servidores DNS 17
   3.5. Consultas DNS 18
   3.6. Principales tipos de registro 19
   3.7. Funcionamiento 20
4. Aplicación desarrollada 22
   4.1. Entorno de desarrollo 23
   4.2. Diagrama de ejecución 24
   4.3. Interfaz de la aplicación 26
   4.4. Ejecución de la aplicación 29
   4.5. Posibles ampliaciones 30
5. Bibliografía 31

Anexo I – Código Fuente 33

1. Introducción

En la actualidad Internet es la mayor herramienta de comunicación del mundo. Es una red global a la cual se conectan millones de dispositivos en todo el mundo. Dichos dispositivos pueden ser tanto PC's como PDAs, estaciones de trabajo, servidores, dispositivos móviles, etc. Para poder identificar todos estos dispositivos, se utiliza el protocolo de Sistema de Nombres de Dominios (DNS).

Cada dispositivo se identifica unívocamente por su dirección IP. Por otra parte, los usuarios les asignan a todos ellos un nombre, el cual es más sencillo de recordar. El protocolo DNS se encarga de traducir estos nombres de dominio en direcciones IP, las cuales son utilizadas por los protocolos de comunicación y por las aplicaciones de internet.

El protocolo DNS es un servicio indispensable para el acceso a internet de los usuarios, puesto que si no funciona correctamente sería imposible conocer las direcciones IP que hay detrás de los nombres de dominio de los servicios, dejando éstos de ser accesibles.

En caso de error o incidencia en la conectividad del sistema de información, una de las primeras comprobaciones a realizar es la del sistema DNS.

Las herramientas disponibles para realizar dichas comprobaciones suelen ser herramientas de línea de comandos. Esto, combinado con la variedad de parámetros que contienen dichas herramientas, provoca para los usuarios no avanzados cierta confusión y dificultad para su manejo.

Durante el desarrollo de este proyecto, se ha implementado una aplicación basada en dichas herramientas para realizar consultas DNS, que ofrece una interfaz más sencilla para el usuario y mucho más amigable a la hora de interpretar los resultados de las consultas.

2. Justificación del problema

2.1. Introducción

Desde que Internet fue creada y utilizada de una forma más significativa, allá por el año 1990, se ha ido convirtiendo cada vez más en una herramienta indispensable para infinidad de objetivos.

Así pues esta indispensabilidad nos ha llevado a, no solo querer saber que hay detrás de este gran universo, sino también a querer entender y explicar cómo es su funcionamiento.

Cada vez que entramos en una página web, ésta se nos muestra bajo un nombre y una dirección específica tales como "www.google.es" o "www.yahoo.com", todas ellas bajo el protocolo HTTP[1]. Sin embargo para resolver esa dirección nuestro navegador ha realizado varias consultas, entre ellas las del protocolo DNS[2], el cual ha sido la base del desarrollo de este proyecto.

DNS es un protocolo básico en internet, el cual nos permite identificar cualquier Nodo o Host[3] conectado a la red. DNS ofrece el servicio de traducción de nombres, devolviendo la dirección IP[4] del servidor en cuestión. Para ello se basa en una estructuración de servidores jerarquizados en los cuales se almacenan y registran, tanto a nivel local, como a nivel global, los datos necesarios para poder localizar el servidor deseado. Gracias a ello la navegación por internet es mucho más sencilla para el usuario final.

Su funcionamiento se puede comprobar mediante la herramienta en línea de comandos **nslookup**, la cual funciona tanto en UNIX como en Windows. Permite realizar dos tipos de consultas, una sencilla o no iterativa que nos muestra simplemente la información principal sobre un dominio, o un consulta iterativa, la cual nos detalla muchos más datos sobre cualquier consulta, como pueden ser el tipo de registros, o las distintas direcciones para un mismo dominio.

La herramienta **nslookup** ha sido la base para desarrollar este proyecto, puesto que, como más adelante detallaré, las alternativas a ésta en el lenguaje C++, por diversos motivos, no han sido posible adaptarlas a la aplicación.

[1] HTTP es un protocolo de internet que significa HiperText Transfer Protocol (protocolo de transferencia de hipertexto).

[2] DNS son las siglas de Domain Name System (sistema de nombres de dominio), uno de los sistemas en los que se basa internet.

[3] Nodo o Host: son dos términos que hacen referencia a cada uno de los dispositivos conectados a internet, ya sea Servidor, portátil, teléfono móvil, etc.

[4] Dirección IP: etiqueta numérica, compuesta por cuatro partes separadas por un punto, que identifican a cada miembro de una red.

2.2. Herramientas disponibles

Para la resolución de consultas DNS, actualmente a través de internet se pueden encontrar varias páginas con diversas herramientas de análisis de tráfico de red y de direcciones IP, gracias a las cuales podemos saciar nuestra curiosidad, y recabar mucha más información sobre ciertos campos, normalmente desconocidos para la mayoría de usuarios de la red.

Sin embargo las opciones que nos proporcionan las herramientas gratuitas son limitadas, cosa que sí permiten las herramientas de pago, aunque la gran mayoría de usuarios no está dispuesta a pagar por un servicio que puede obtener de manera gratuita aunque sea de menor nivel.

NMONITORING

http://www.nmonitoring.com

Esta web nos da la posibilidad de utilizar varias herramientas, entre ellas un consultor DNS, que nos indica la dirección o direcciones IP que pertenecen a cierto dominio web.

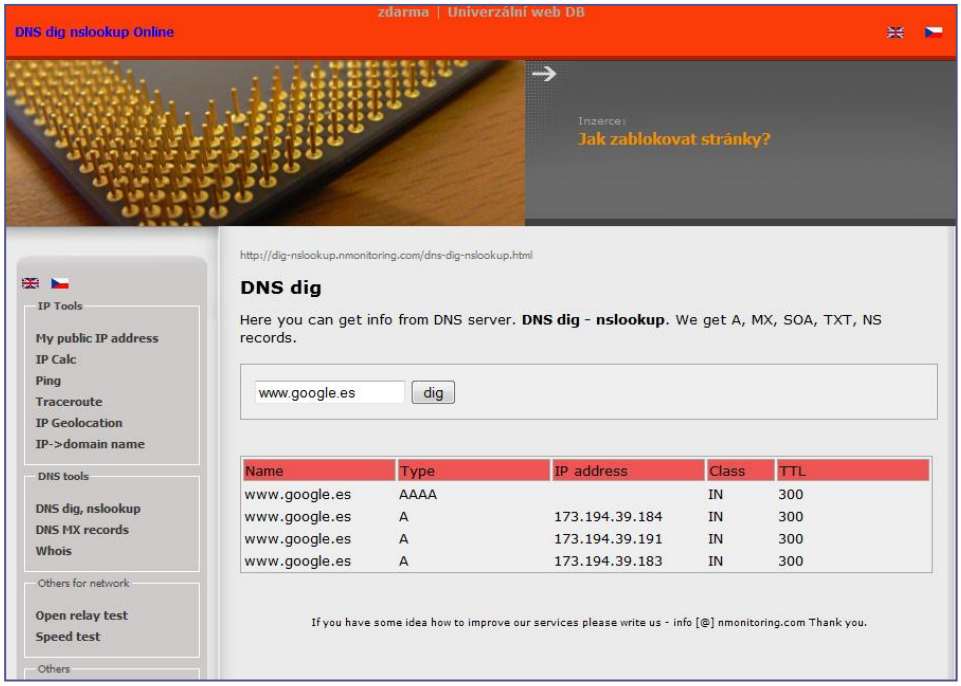


Figura 1. Resultado de consulta DNS en www.nmonitoring.com.

Aquí se pueden observar los resultados de la consulta sobre el dominio "www.google.es" (la cual será la que utilizaremos para el análisis de todas las herramientas), en los cuales nos indica el nombre, tipo, dirección IP, clase, y TTL[5] de cada respuesta.

Sin embargo no permite elegir el servidor para la consulta, ni el tipo de registro sobre el que hacer la consulta.

[5] TTL: Tiempo de vida o Time To Live en sus siglas en inglés, indica el momento en el que el registro debe ser borrado de la caché.

DNSQUERIES

http://www.dnsqueries.com/es/consulta_servidor_dns.php

Página web disponible en varios idiomas que proporciona diferentes herramientas además de las consultas DNS, tales como un convertidor a IPv4 o un escáner de puertos. Si nos movemos por su menú de herramientas hasta la opción "DQ tools/DNS tools/Hacer peticiones DNS", podemos utilizarla para resolver la dirección DNS que queramos.

Figura 2. Resultados de la búsqueda en www.dnsqueries.com.

Esta página web nos proporciona información muy escueta por cada registro. Sin embargo, nos permite elegir el tipo de registro de la consulta, pero no así el servidor para la consulta. Contiene gran cantidad de publicidad, aunque la imagen haya sido recortada para no mostrarla.

DNSSTUFF

http://www.dnsstuff.com/bc-lookups-tools

Herramienta de pago que nos permite elegir más datos para la consulta, como el servidor, el tipo de registro y la duración de la consulta. No contiene publicidad.

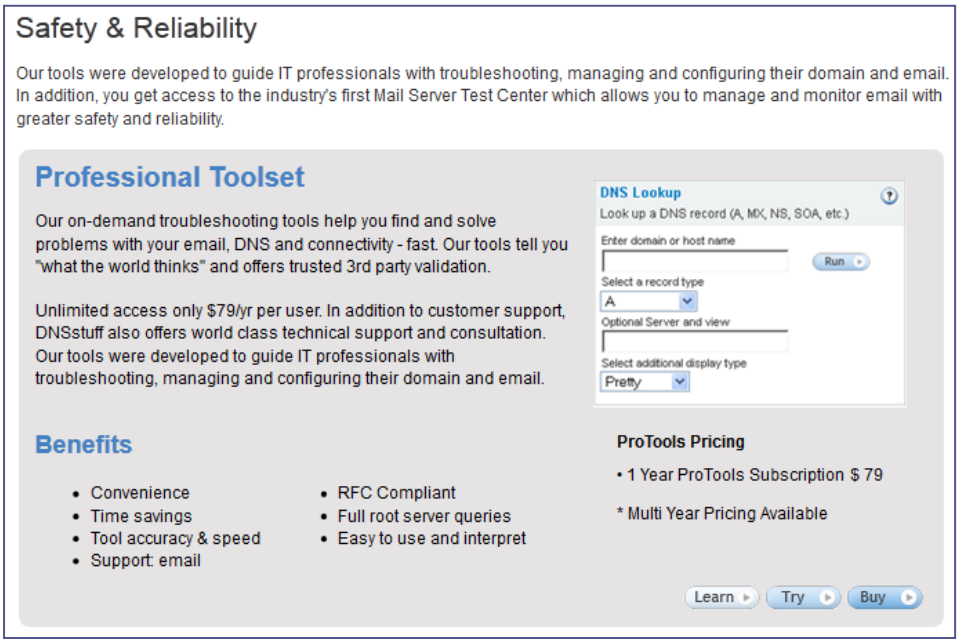


Figura 3. Interfaz de la herramienta DNSStuff.

2.3. Otras opciones alternativas a nslookup

Durante la elaboración de este proyecto se valoraron dos opciones de realización:

1. Realizarlo mediante Sockets.

2. Utilizar la herramienta *nslookup* como base.

Los sockets son mecanismos de comunicación en la red que se utilizan durante la ejecución del programa. Una vez el programa haya finalizado, el socket se desconecta, si bien, durante la ejecución el programa puede realizar dicha desconexión si fuese necesario. Permiten ejecutar programas en una arquitectura cliente-servidor, donde como es habitual el cliente realiza las peticiones al servidor, el cual le devuelve la información solicitada.

En el lenguaje C++ era necesario utilizar las librerías: *winsock.h, Ws2_32.lib, User32.lib,* y *Shell32.lib*. Una vez incluidas en el programa se empezaron a dar bastantes problemas de compatibilidad, y errores de conexión simplemente al compilar el código.

Una vez investigada la naturaleza de los errores, las soluciones posibles aún a pesar de probarlas todas e incluso algunas más, no se consiguieron solucionar dichos errores, por lo que se decidió pasar a la opción de utilizar la aplicación *nslookup*.

Debido a la naturaleza de *nslookup*, puesto que es una herramienta que funciona en modo consola de ejecución, con múltiples comandos adicionales, esta aplicación tiende a ser utilizada por usuarios avanzados. Así pues, la información mostrada por dicha herramienta dista bastante de ser de fácil inteligibilidad, debido a que se muestra en un entorno poco amigable hacia el usuario. Basándose en este motivo, el presente proyecto ha desarrollado esta aplicación con un entorno gráfico más sencillo y agradable a la vista y a la legibilidad de cara al usuario final, así como una utilización mucho más sencilla, sin necesidad de ejecutar ningún comando enrevesado ni tener que desarrollarlo en la consola del sistema, puesto que simplemente el usuario introducirá su consulta y se le devolverá la información referenciada a ella.

3. Protocolo DNS

3.1. Definición

Las siglas DNS significan Sistema de Nombres de Dominio, por sus siglas en inglés (Domain Name System). Es uno de los protocolos básicos de internet, y su principal función es la de transformar un nombre de dominio, fácilmente asociable y recordable a un tipo de servicio en cuestión, en una dirección IP.

Para ello utiliza una base de datos distribuida y jerárquica, almacenada en su estructura de servidores. Dichos servidores utilizan el protocolo UDP[6] por el puerto 53 para responder las consultas de las unidades clientes.

Los RFCs 1034 y 1035 son los correspondientes a este sistema, En ellos se desarrollan y explican todas sus especificaciones.

[6] UDP: User Datagram Protocol. Protocolo utilizado para enviar paquetes de información a través de la red, sin necesidad de que haya previamente una conexión.

3.2. Componentes principales

El protocolo DNS está formado por tres componentes fundamentales:

- **Espacio de nombres** (Domain Name Space). Junto al registro de recursos (Resource Records) constituyen las especificaciones de un árbol estructurado de datos asociado con los dominios. En teoría cada hoja o nodo del árbol da nombre a un tipo de información asociada al dominio, y guarda los datos correspondientes. Para obtener dicha información se utilizan las consultas, las cuales llaman al servidor del nombre del dominio del que requieren información.

- **Servidores de nombres** (Name Servers). Programas que responden a las peticiones de los clientes, los cuales tienen la información de la estructura del árbol de dominios, y guardan la información referente a los nombres de los dominios. Un servidor es capaz de guardar cualquier parte de la estructura de árbol, pero generalmente contiene la información de la parte del árbol en la que se encuentra. Además almacena punteros a otros servidores en los cuales se encuentra la información de los nombres de dominio de otras partes del árbol.

- **Resolvers**. Comúnmente llamados clientes DNS, son los programas que se encuentran en los ordenadores de los usuarios y generan las peticiones para resolver las direcciones que necesitan las aplicaciones. Dichas peticiones se envían a los servidores de nombres.

3.3. Nombre de Dominio

Los nombres de dominio son el método que contiene internet para organizar todos sus contenidos y servicios, los cuales se almacenan de una manera jerárquica, mediante una estructura de árbol, como se ha mencionado anteriormente.

Gracias a este árbol se puede componer un nombre de dominio de una manera bastante intuitiva. En la figura que se observa a continuación podemos ver una parte de la estructura de almacenamiento de los dominios en el protocolo DNS, en el cual se remarca la ruta a seguir para poder componer un nombre de dominio.

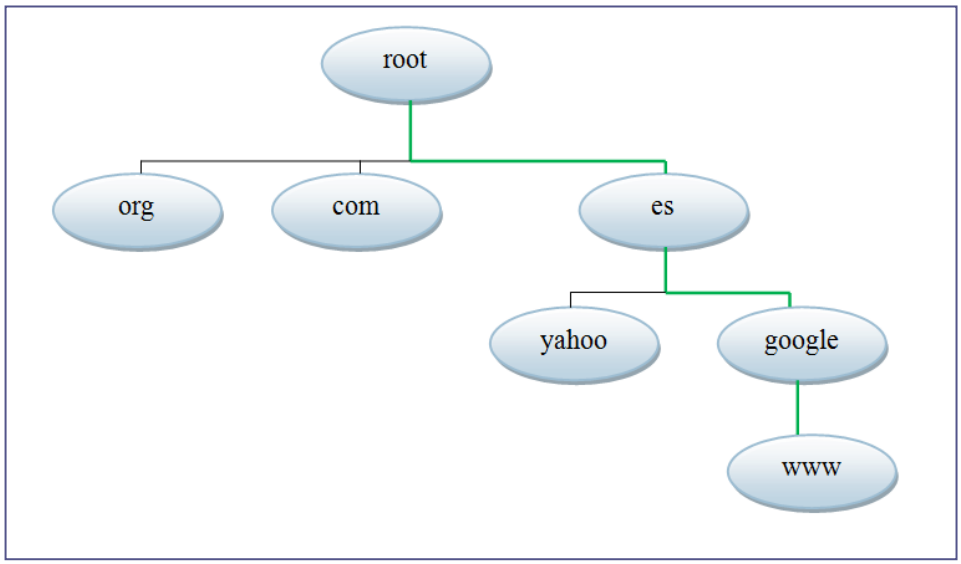


Figura 4. Estructura jerarquizada del protocolo DNS

El primer nivel o raíz (root) es el que sostiene al resto de nodos que conforman esta estructura. El siguiente nivel es el conocido como los dominios del nivel superior, o TLD (Top Level Domains), los cuales conforman la parte final del nombre de dominio y pueden indicarnos a qué tipo de servicio estamos accediendo. A partir de ellos, y en orden descendente, se encuentran los subdominios, estructura que puede constar de varios subniveles hasta llegar al último nivel, o nivel de host.

Una vez se sabe la ruta, la forma de componer el dominio es inversa al sentido del desplazamiento, por lo que la parte superior es el final del nombre del dominio y la inferior conforma el principio. Hay que tener en cuenta que se introduce el carácter punto (“.”) como separación entre cada nivel, por lo que en este caso el nombre de dominio solicitado sería “www.google.es.”. Esta ruta completa que incluye al nodo raíz, el punto final, es denominada dirección FQDN (Fully Qualified Domain Name) y permite identificar de manera unívoca un servicio de internet. Hay que indicar que el punto final correspondiente al nodo raíz se suele omitir, sin afectar esto a la identificación del servicio.

En cuanto a los dominios de nivel superior o TLD, suelen indicar un cierto tipo de dominio bastante general asociados a ellos. A continuación se especifican algunos ejemplos:

- **COM** – servicios web comerciales.
- **ORG** – servicios web para organizaciones sin ánimo de lucro.

- **GOV** – restringido a servicios web del gobierno, sólo para EE.UU.
- **ES** – servicios web de España.
- **AR, UK, FR** – servicios referentes a diferentes de los países correspondientes con sus siglas. En este caso Argentina, Reino Unido y Francia.

Teniendo en cuenta que cada nombre de dominio debe ser único, existen organizaciones nacionales e internacionales encargadas de gestionar y registrar dichos nombres de dominios, tales como la organización NIC (Network Information Center), gestionada por el Ministerio de Industria, Energía y Turismo del Gobierno de España (existe una por cada país del mundo) o la organización ICANN (Internet Corporation for Assigned Names and Numbers), que es la principal responsable a nivel internacional de asignar el espacio de direcciones numéricas del protocolo IP, de los identificadores de protocolos, de las funciones de gestión del sistema de nombres de dominio y de la administración del sistema de servidores DNS raíz.

3.4. Tipos de servidores DNS

En el protocolo DNS se pueden diferenciar tres tipos de servidores según su ámbito:

- **Principales** o maestros. Son los que se encargan de guardar los datos de un cierto espacio de nombres correspondiente a su zona. También son los encargados de mantener actualizada toda la información de los dominios.

- **Secundarios** o esclavos**.** Estos servidores obtienen la información mediante consultas al servidor principal asociado a su zona.

- **Caché** o locales. Suelen estar incluidos en el software. No tienen autoridad sobre ningún dominio. Su funcionalidad consiste en contactar con otros servidores para resolver las consultas que no tiene en su memoria caché. Dicha memoria contiene la información de las últimas consultas solicitadas, para poder mejorar la velocidad de resolución para el usuario.

3.5. Consultas DNS

El protocolo DNS utiliza dos tipos de consultas para responder a las peticiones de los clientes:

- **Recursiva.** El cliente realiza una petición al servidor caché. Éste busca la información en su memoria local, y si no la tiene acude al servidor con autoridad que la contiene. Este servidor con autoridad realizará la consulta en nombre del servidor caché, preguntando al siguiente servidor en el árbol de servidores, y así hasta llegar al servidor que contiene la información sobre la consulta realizada.

  En este tipo de consultas, el servidor caché es quien realiza él mismo las consultas a otros servidores en lugar del cliente.

- **Iterativa.** El servidor caché devuelve al cliente la mejor respuesta posible en función de su información en memoria. Si no dispone de dicha información indica la dirección IP del siguiente servidor autorizado al que preguntar, empezando en todos los casos por un servidor maestro.

  Dicho servidor le indica el servidor de siguiente nivel que contiene la información. El servidor caché vuelve a enviar la petición a este último servidor el cual, si no dispone de la información requerida realiza la misma acción que el servidor superior, es decir, envía al servidor caché la dirección IP de un servidor que podría resolver la consulta, y así sucesivamente hasta llegar al servidor que contiene la información sobre el dominio solicitado.

3.6. Principales tipos de registro

El protocolo DNS consta de varios tipos de registros según el tipo de información al que se refieren. Los más utilizados son:

- **A** – Address (Dirección) registro utilizado para traducir nombres de dominio a direcciones IPv4.
- **AAAA** – Address (Dirección) registro utilizado para traducir nombres de dominio a direcciones IPv6.
- **CNAME** – Canonical Name (Nombre Canónico) registro utilizado para referirse a alias de los servidores de hosts de un dominio. Se utiliza cuando se ejecutan múltiples servicios en un mismo host con una sola dirección IP, o cuando se utiliza un alias más sencillo de recordar que el nombre canónico original.
- **NS** – Name Server (Servidor de Nombres) registro que nos indica una lista de servidores de nombres con autoridad para ese dominio.
- **MX** – Mail Exchanger (Intercambio de Correo) registro que informa de una lista de servidores de correos para ese dominio.
- **PTR** – Pointer (Puntero) registro utilizado para la resolución inversa. Funciona al contrario que el registro A, traduciendo direcciones IP en nombres de dominio.
- **SOA** – Start of Authority (Inicio de Autoridad) registro que indica el servidor maestro de la zona referida en la consulta.
- **SRV** – Service (Servicio) registro de servicios que informa de todos los servicios disponibles para un dominio. Protocolos como SIP (Session Initiation Protocol) o XMPP (Extensible Messaging and Presence Protocol) suelen requerir registros de este tipo para proporcionar información a los clientes.

3.7. Funcionamiento

El protocolo DNS tiene un funcionamiento en segundo plano de cara al usuario, puesto que los que se encargan de realizar las consultas necesarias para su funcionamiento son los programas utilizados para acceder a la red, tales como navegadores, o gestores de correo.

Pongamos por ejemplo, para explicar dicho funcionamiento, que el usuario quiere acceder a un sitio web desde su navegador. En este caso el funcionamiento dependería de si la consulta es de tipo recursiva o iterativa. A continuación se explicarán los pasos que se siguen en ambos casos para resolver la dirección IP del dominio solicitado.

Consulta recursiva.

- El navegador realiza la consulta a su servidor DNS local o de caché.
- El servidor devuelve la dirección IP en caso de que se encuentre en su base de datos. Si no es así, el servidor local realizará la consulta en nombre del cliente, preguntando a un servidor secundario o al servidor principal. En cualquier caso el cliente recibirá la dirección IP sin preguntar a otro servidor.

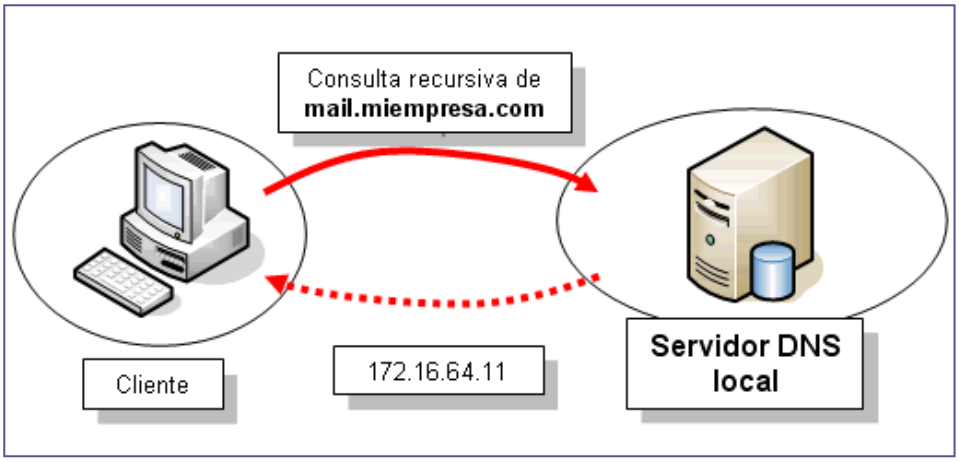


Figura 5. Esquema ejemplo de una consulta recursiva.
Fuente: www.adrformacion.com

Consulta iterativa.

- En primer lugar el navegador consulta su memoria DNS caché incluida dentro de la aplicación. Si no contiene la información solicitada realiza una consulta al servidor de nivel superior. Este paso se corresponde con una consulta recursiva, sin embargo al remitirla a otro servidor superior la convierte en iterativa.
- El servidor caché realiza la petición al servidor secundario correspondiente, el cual consulta en su memoria caché, si no contiene la información requerida envía al servidor local la dirección IP de un servidor que podría resolver la consulta.
- Cuando la consulta llega al servidor maestro, éste estudia la petición de información y envía al servidor local la dirección IP del servidor TLD correspondiente a ese dominio (p.ej. dominio ".com").

- El servidor local pregunta al servidor TLD por la dirección IP y si no tiene la respuesta, éste último envía al servidor local la dirección IP del servidor al que pertenece ese dominio.

- Una vez en el servidor autorizado para ese dominio, el servidor local le solicita la información correspondiente a la dirección IP, puesto que los servidores DNS almacenan para cada dominio varios registros con distintos tipos de información. Esta dirección IP se remite al servidor DNS local o de caché.

- El servidor caché del equipo almacena la información obtenida en su memoria DNS, para que en caso de volver a necesitarla no tener que volver a realizar la consulta completa.

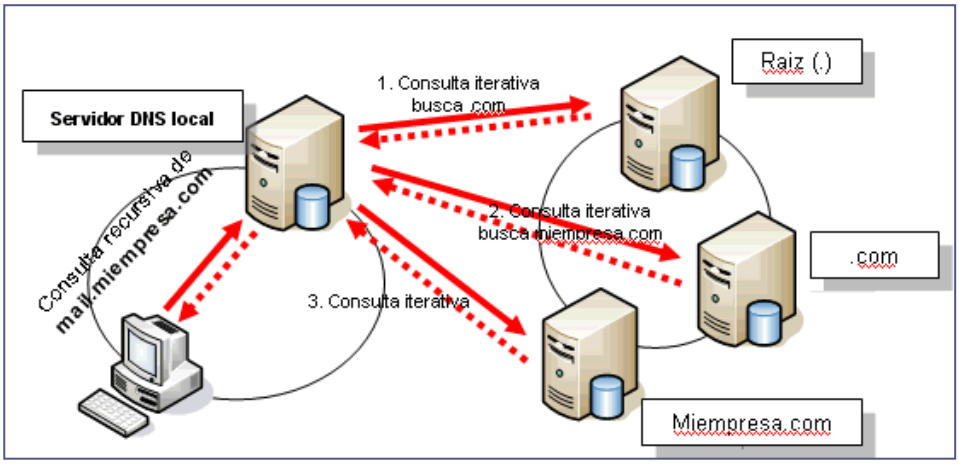


Figura 6. Esquema ejemplo de una consulta recursiva.
Fuente: www.adrformacion.com

4. Aplicación desarrollada

A continuación se va a proceder a describir los pasos que se han seguido para la implementación de la aplicación para este proyecto.

Dicha aplicación está desarrollada bajo el lenguaje de programación C++ y funciona de manera independiente a cualquier aplicación web. Para ello realiza una ejecución en segundo plano de la herramienta **nslookup**, que se encuentra en los sistemas operativos *Windows* de *Microsoft*, y analiza posteriormente los resultados devueltos por ésta.

Como resultado final, y para un fácil entendimiento por parte del usuario, se muestra la información obtenida después de realizar la consulta de una manera visual y comprensible para el usuario.

4.1. Entorno de desarrollo

Para el desarrollo de este proyecto se eligió el sistema operativo *Windows XP*, puesto que ofrece una estabilidad superior del entorno *Microsoft Visual Studio*, que otros sistemas como por ejemplo *Windows Vista*.

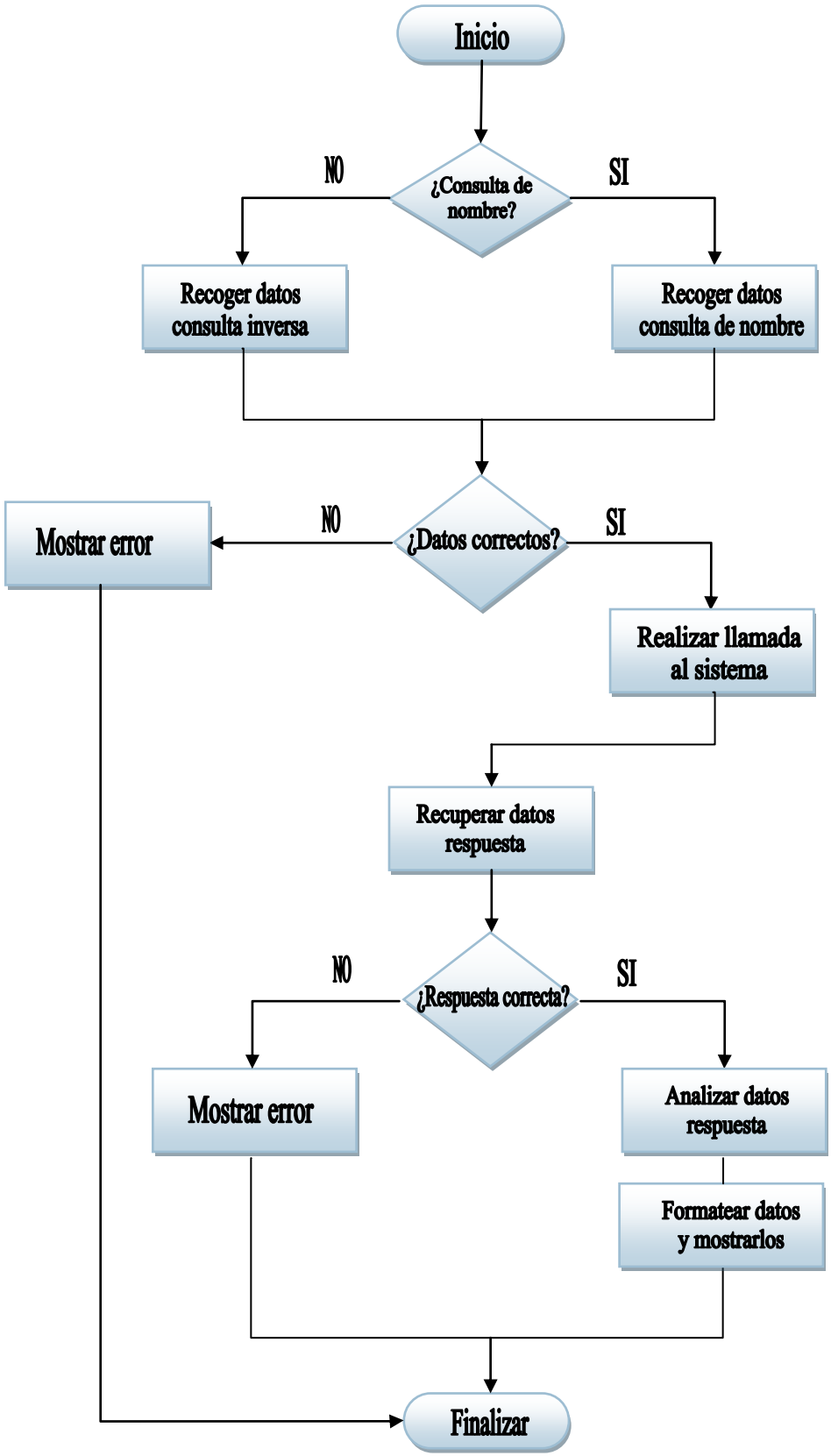
La elección de *Microsoft Visual Studio* responde a varios factores. El primero, como se ha indicado anteriormente, es la estabilidad bajo la que funciona en *Windows XP*.

El segundo factor corresponde a que la licencia de dicho programa, que gracias al acuerdo existente entre *Microsoft* y la Universidad Politécnica de Valencia mediante la plataforma *Polidotnet*, permite descargarlo y utilizarlo bajo licencia de uso universitario. Este factor propinó la elección frente a otros entornos de desarrollo de licencia gratuita, tales como *Dev C++* o *Netbeans*, herramientas que están bastante extendidas entre los usuarios dedicados al desarrollo de aplicaciones, sobre todo para su uso personal más que profesional.

El tercer y último factor para la elección se refiere a la familiarización que tengo con este entorno de programación, puesto que durante los años de estudio de la carrera he utilizado dicha herramienta para el desarrollo de aplicaciones para las prácticas de varias asignaturas, por lo que me era más sencillo configurar este entorno, así como poder consultar la información dentro del proyecto.

4.2. Diagrama de ejecución

El diagrama que se expone a continuación se corresponde al funcionamiento de la aplicación.



Según se puede ver en el diagrama, el funcionamiento de la aplicación es el siguiente. Una vez el usuario rellena los datos correspondientes para la consulta, pulsa el botón que inicia la misma. En ese momento el programa comprueba qué tipo de consulta ha sido seleccionada por el usuario, y recoge los datos correspondientes. Si los datos tienen el formato correcto se prosigue con la consulta, si no es así se muestra un mensaje de error para informar al usuario.

Para poder realizar la consulta, en primer lugar el programa tiene que componer el comando de ejecución. Para ello añade el texto de la consulta introducido por el usuario al final de la sentencia de comandos. Después de realizar la composición del comando, ejecuta la llamada del sistema para ejecutar la herramienta *nslookup* en segundo plano.

Cuando se ejecuta la llamada al sistema los valores de respuesta son almacenados temporalmente para su análisis posterior. Si los datos no son correctos, es decir, si la consulta no ha sido exitosa, se muestra mediante un error al usuario.

Si la consulta ha sido correcta se procede al análisis de los resultados. Cabe destacar que la herramienta *nslookup* es bastante confusa en cuanto al formato de los resultados finales se refiere, sobre todo para usuarios que no sean expertos en su manejo. Para dicho análisis se buscan solamente los datos necesarios, realizando con ello un filtrado de información no necesaria para nuestros objetivos. En este sentido cabe destacar que se utilizarán los registros de tipo Address, CName y Pointer, puesto que esto nos permite mostrar mediante los resultados, las subconsultas que se realizan durante la ejecución de una consulta principal.

Después del análisis se procede a formatear los datos de la consulta, para mostrar la información mucho más específica y de forma entendible para el usuario. Se agrupan los datos correspondientes a los totales y se muestran finalmente las subconsultas realizadas.

4.3. Interfaz de la aplicación

En este apartado se va a mostrar y explicar los campos que contiene la interfaz gráfica visible para el usuario.

Figura 7. Campos de consulta inversa.

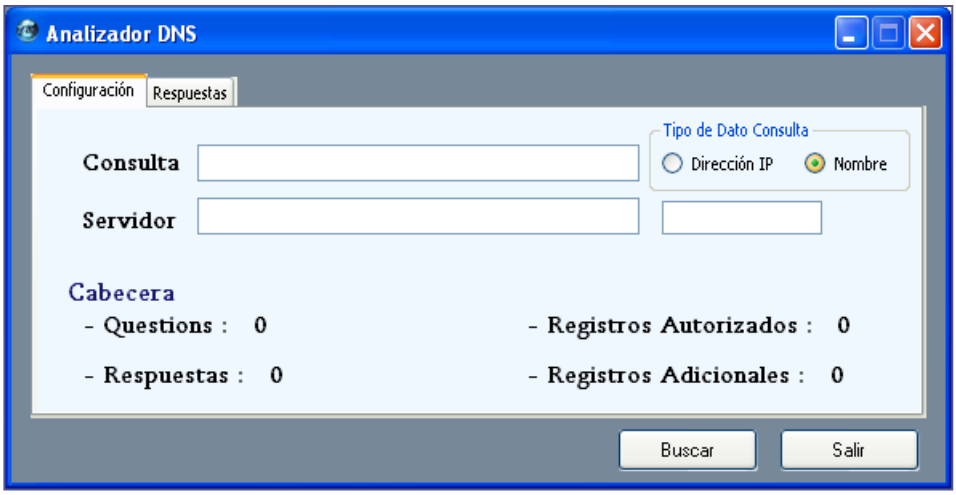


Figura 8. Interfaz de la aplicación, pestaña de "Configuración" de consulta.

En las figuras 7 y 8 se muestran las dos opciones que posee la interfaz de la aplicación en la pestaña de "Configuración". Mediante el dato "Tipo de Dato Consulta", el usuario elige la consulta que quiere realizar, mostrando u ocultando para ello el campo correspondiente.

Los dos únicos campos que el usuario puede modificar son, el ya anteriormente mencionado "Tipo de Dato Consulta" y el campo de "Consulta".

Campos de la interfaz.

Pestaña "Configuración"

- **Tipo de Dato Consulta:** Como ya se ha indicado anteriormente, el usuario puede seleccionar el tipo de consulta a realizar. Por defecto aparece marcado el tipo "Nombre", que muestra en el campo "Consulta" un cuadro de texto para el nombre de dominio. Si el usuario marca el tipo "Dirección IP" se muestra un campo formateado para introducir la dirección IP de la consulta, como puede verse en la figura 7.

- **Consulta:** en este campo el usuario introducirá el nombre de la consulta a realizar en un cuadro de texto, o por el contrario, si la consulta es de tipo inversa, introducirá una dirección IP en el cuadro formateado para ello.

- **Servidor:** campo que muestra la información referente al servidor que, finalmente resuelve la consulta realizada. Consta de dos cuadros de texto, uno para indicar el nombre del servidor y otro que nos muestra la dirección IP del mismo.

- **Cabecera:** esta sección del interfaz muestra los datos resultantes en la cabecera de las respuestas de la consulta. Se muestra en cuatro campos:
    - **Questions:** indica el número total de subconsultas realizadas para resolver la consulta principal.
    - **Respuestas:** indica el número total de respuestas encontradas a las subconsultas realizadas mediante la resolución.
    - **Registros Autorizados:** o autoritativos, indica el número de los registros recibidos por parte de servidores autorizados durante la consulta.
    - **Registros Adicionales:** indica el número de registros adicionales recibidos en las respuestas a la consulta.

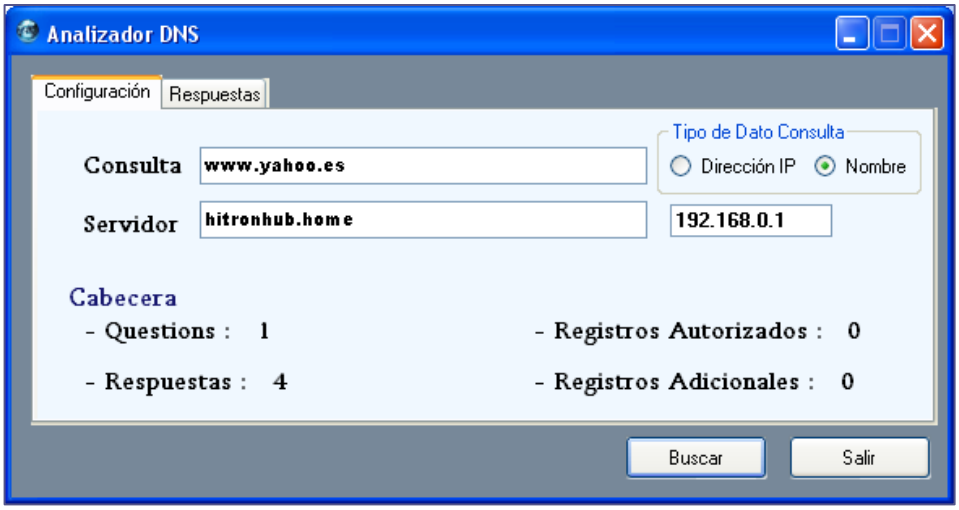


Figura 9. Interfaz resultante después de realizar una consulta.

Pestaña "Respuestas"

En esta pestaña se muestran las diferentes consultas y respuestas que se han ido realizando para resolver la consulta principal. De esta manera el usuario puede ver de una manera sencilla cual ha sido el desarrollo seguido para obtener el resultado correcto.

En este caso se muestran las respuestas a la consulta del nombre de dominio "www.yahoo.es". Cada respuesta consta de dos líneas. La primera indica la consulta realizada, y la segunda nos muestra el tipo de registro que se ha obtenido y su valor. Cada respuesta se separa con una línea en blanco. El último registro mostrado, el cual se separa mediante dos líneas en blanco, se refiere al resultado final de la consulta. En este caso nos indica el nombre, la dirección y el alias correspondientes al dominio "www.yahoo.es".

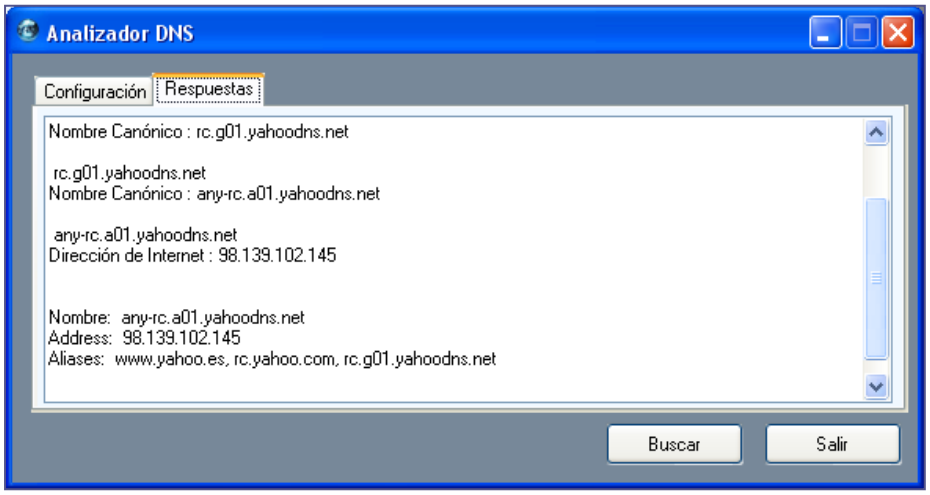


Figura 10. Interfaz de la aplicación, pestaña de "Respuestas" de la consulta.

Botón "Buscar": Cuando el usuario pulsa este botón se procede a la ejecución del proceso de consulta, sobrescribiendo los datos de una consulta anterior si los hubiere.

Botón "Salir": Cierra la aplicación.

4.4. Ejecución de la aplicación

Para ejecutar correctamente esta aplicación, se ha creado un instalador de la misma, el cual comprueba los paquetes instalados en el terminal en el cual se va a instalar, y si fuese necesario los instala. Para la instalación de la aplicación se elige la ruta C:/Archivos de programas/UPV/AnalizadorDNS, en donde crea las carpetas necesarias para su ejecución. A su vez crea un acceso directo en el escritorio para su fácil localización.

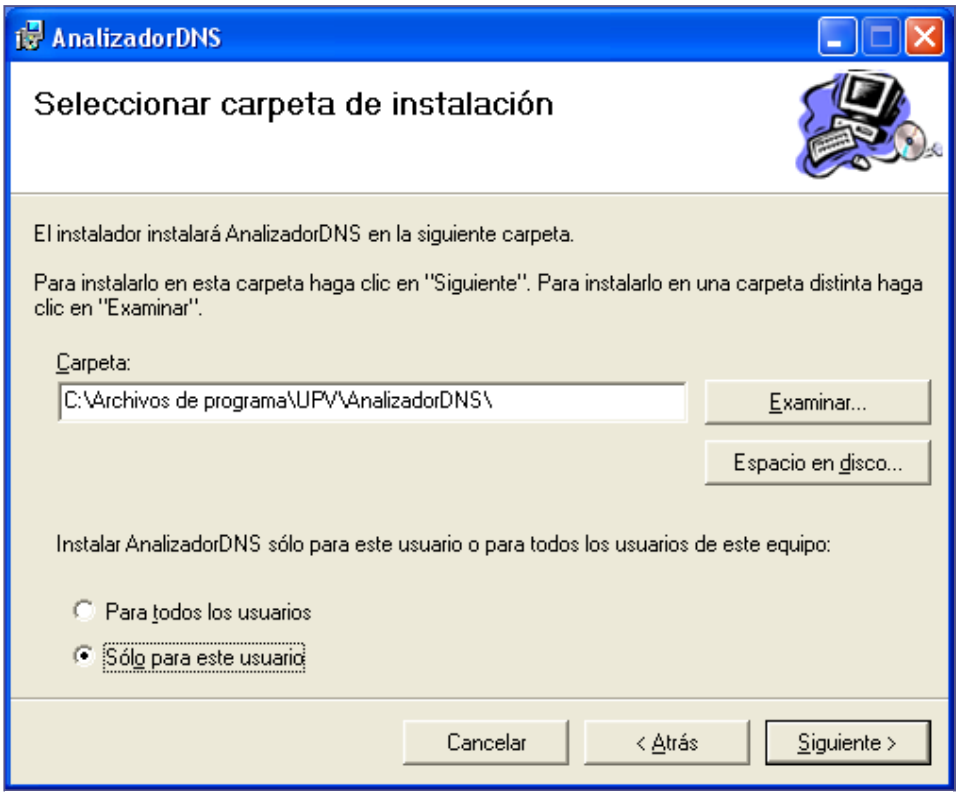


Figura 11. Instalador de la aplicación

Figura 12. Acceso directo de la aplicación en el escritorio.

4.5. Posibles ampliaciones

Una vez finalizada la aplicación, han sido analizadas algunas posibles ampliaciones que servirían para hacer un estudio de mayor amplitud sobre el protocolo DNS.

Una primera ampliación posible tiene como objetivo la ampliación de los registros analizados. En esta primera versión de la aplicación se analizan los registros de tipo Address, CName y Pointer, tipos de registros que nos permiten realizar el análisis deseado. Sin embargo, en un futuro, para otras funcionalidades se podría ampliar el número y tipo de registros analizados, para obtener información más concreta y ampliada.

Esta primera versión de la aplicación analiza perfectamente el tráfico de IPv4, por lo que una ampliación para tráfico de IPv6 sería una posibilidad susceptible de ser estudiada y realizada.

5. Bibliografía

DNS

- RFC 2181 - Clarifications to the DNS Specification.
- RFC 1123 - Requirements for Internet Hosts - Application and Support.
- RFC 1034 - Domain names - concepts and facilities.
- RFC 1035 - Domain names - implementation and specification.
- http://es.wikipedia.org/wiki/Domain_Name_System
- http://www.redesyseguridad.es/el-protocolo-dns/
- http://www.alcancelibre.org/staticpages/index.php/introduccion-protocolo-dns
- http://www.24x7linux.com/documentation/internet/how_dns_works.shtml
- http://www.adrformacion.com/cursos/exchange10/leccion2/tutorial6.html

Nslookup

- http://support.microsoft.com/kb/200525/es
- http://www.openredes.com/2011/04/28/nslookup-obtener-ip-de-un-dominio-servidores-de-nombres-dns-dns-inverso-y-servidores-de-mail-de-dominios/

C++

- http://c.conclase.net
- http://www.cplusplus.com
- http://msdn.microsoft.com/es-es/library/ms123401

Tratamiento Ficheros

- http://www.aprendeaprogramar.com/mod/resource/view.php?id=171
- http://www.latindevelopers.com/forum/como-eliminar-archivos-de-texto-t556.html

- http://c.conclase.net/ficheros/index.php?cap=002

Anexo I – Código Fuente

A continuación se presenta el código fuente de la aplicación, indicando a que archivo pertenece cada uno.

Fichero “AnalizadorDNS.cpp”.

```
// AnalizadorDNS.cpp: archivo de proyecto principal.

#include <stdafx.h>
#include "Form1.h"

using namespace AnalizadorDNS;

[STAThreadAttribute]


int main(array<System::String ^> ^args)
{
	// Habilitar los efectos visuales de Windows XP antes de crear ningún control
	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);

	// Crear la ventana principal y ejecutarla
	Application::Run(gcnew Form1());
	return 0;
}

System::String^ EliminarBlancos(System::String^ cadena2)/*Función para eliminar los blancos de las cadenas de texto recogidas como datos de búsqueda*/
{

	System::String^ aux;
	char caracter;
	int i,j=0;
	for (i=0;i< cadena2->Length; i++)
	{
		caracter = cadena2[i];

		if (caracter != ' '){

			aux=aux+cadena2[i];
			j++;
		}
	}

	return aux;
}

void ConvertirACharArray(System::String^ cadena1, char *cad1)
{
	int i=0;
	for (i=0; i< cadena1->Length; i++)
	{
		cad1[i]=cadena1[i];
	}
```

```
	cad1[i++]='\n';
}

System::String^ ObtenerFicheroQuestions()//Obtiene los datos
correspondientes a Questions del Fichero
{
	System::String^ resultadoFinal;
	System::String^ cadenaString;
	FILE *f;
	char cadena[100];
	char *dato;
	char aux[30];

	f = fopen("fichero.txt", "r");
	//Comprobamos que se abre bien el fichero
	if(!f)
	{
		resultadoFinal = "Fallo";
		return resultadoFinal;
	}
	//Recorremos el fichero linea por linea
	while (!feof(f)&&(fgets(cadena, 100, f)!=NULL))
	{
		dato = strstr(cadena, "questions =");
		if(dato != NULL)
		{
			cadenaString = gcnew System::String(cadena);
			cadenaString = cadenaString->Trim();
			cadenaString = EliminarBlancos(cadenaString);
			int indinicio, indcoma;
			indinicio = cadenaString->IndexOf("questions=");
			indinicio+=10;//indinicio muestra el indice de
inicio, le sumamos la longitud (10)para poder coger el valor del dato
			indcoma = cadenaString->IndexOf(',',indinicio);
			resultadoFinal = cadenaString->Substring
(indinicio,indcoma-indinicio);
		}
	}
	int cerrar = fclose(f);

	return resultadoFinal;
}

System::String^ ObtenerFicheroNumRespuestas()//Obtiene los datos de
respuestas del fichero
{
	System::String^ resultadoFinal;
	System::String^ cadenaString;
	FILE *f;
	char cadena[100];
	char *dato;
	char aux[30];

	f = fopen("fichero.txt", "r");
	//Comprobamos que se abre bien el fichero
	if(!f)
	{
		resultadoFinal = "Fallo";
		return resultadoFinal;
	}
	//Recorremos el fichero linea por linea
```

```
	while (!feof(f)&&(fgets(cadena, 100, f)!=NULL))
	{
		dato = strstr(cadena, "answers =");
		if(dato != NULL)
		{
			cadenaString = gcnew System::String(cadena);
			cadenaString = cadenaString->Trim();
			cadenaString = EliminarBlancos(cadenaString);
			int indinicio, indcoma;
			indinicio = cadenaString->IndexOf("answers=");
			indinicio+=8;//indinicio muestra el indice de inicio,
le sumamos la longitud (10) para poder coger el valor del dato
			indcoma = cadenaString->IndexOf(',',indinicio);
			resultadoFinal = cadenaString->Substring
(indinicio,indcoma-indinicio);
		}
	}
	int cerrar = fclose(f);

	return resultadoFinal;
}

System::String^ ObtenerFicheroRegistrosAutorizados()//Obtiene los
datos de registros autorizados del fichero
{
	System::String^ resultadoFinal;
	System::String^ cadenaString;
	FILE *f;
	char cadena[100];
	char *dato;
	char aux[30];

	f = fopen("fichero.txt", "r");
	//Comprobamos que se abre bien el fichero
	if(!f)
	{
		resultadoFinal = "Fallo";
		return resultadoFinal;
	}
	//Recorremos el fichero linea por linea
	while (!feof(f)&&(fgets(cadena, 100, f)!=NULL))
	{
		dato = strstr(cadena, "authority records =");
		if(dato != NULL)
		{
			cadenaString = gcnew System::String(cadena);
			cadenaString = cadenaString->Trim();
			cadenaString = EliminarBlancos(cadenaString);
			int indinicio, indcoma;
			indinicio = cadenaString->IndexOf
("authorityrecords=");
			indinicio+=17;//indinicio muestra el indice de
inicio, le sumamos la longitud (10) para poder coger el valor del dato
			indcoma = cadenaString->IndexOf(',',indinicio);
			resultadoFinal = cadenaString->Substring
(indinicio,indcoma-indinicio);
		}
	}
	int cerrar = fclose(f);

	return resultadoFinal;
```

```
}

System::String^ ObtenerFicheroRegistrosAdicionales()
{
	System::String^ resultadoFinal;
	System::String^ cadenaString;
	FILE *f;
	char cadena[100];
	char *dato;
	char aux[30];

	f = fopen("fichero.txt", "r");
	//Comprobamos que se abre bien el fichero
	if(!f)
	{
		resultadoFinal = "Fallo";
		return resultadoFinal;
	}
	//Recorremos el fichero linea por linea
	while (!feof(f)&&(fgets(cadena, 100, f)!=NULL))
	{
		dato = strstr(cadena, "additional =");
		if(dato != NULL)
		{
			cadenaString = gcnew System::String(cadena);
			cadenaString = cadenaString->Trim();
			cadenaString = EliminarBlancos(cadenaString);
			int indinicio;
			indinicio = cadenaString->IndexOf("additional=");
			indinicio+=11;//indinicio muestra el indice de
inicio, le sumamos la longitud (10) para poder coger el valor del dato
			resultadoFinal = cadenaString->Substring
(indinicio,cadenaString->Length-indinicio);
		}
	}
	int cerrar = fclose(f);

	return resultadoFinal;
}

System::String^ ObtenerFicheroDatosServidorNombre()
{
	System::String^ resultadoFinal;
	System::String^ cadenaString;
	FILE *f;
	char cadena[100];
	char *dato;
	char aux[30];

	f = fopen("fichero.txt", "r");
	//Comprobamos que se abre bien el fichero
	if(!f)
	{
		resultadoFinal = "Fallo";
		return resultadoFinal;
	}
	//Recorremos el fichero linea por linea
	while (!feof(f)&&(fgets(cadena, 100, f)!=NULL))
	{
		dato = strstr(cadena, "Servidor:");
		if(dato != NULL)
```

```
			{
				cadenaString = gcnew System::String(cadena);
				cadenaString = cadenaString->Trim();
				cadenaString = EliminarBlancos(cadenaString);
				int indinicio;
				indinicio = cadenaString->IndexOf(":")+1;
				resultadoFinal = cadenaString->Substring
(indinicio,cadenaString->Length-indinicio);
			}
		}
		int cerrar = fclose(f);
		return resultadoFinal;
}

System::String^ ObtenerFicheroDatosServidorIP()
{
		System::String^ resultadoFinal;
		System::String^ cadenaString;
		FILE *f;
		char cadena[100];
		char *dato; //Se utiliza para saber si encuentra la cadena
especificada
		char aux[30];

		f = fopen("fichero.txt", "r");
		//Comprobamos que se abre bien el fichero
		if(!f)
		{
			resultadoFinal = "Fallo";
			return resultadoFinal;
		}
		//Recorremos el fichero linea por linea
		while (!feof(f)&&(fgets(cadena, 100, f)!=NULL))
		{
			dato = strstr(cadena, "Servidor:");
			if(dato != NULL)
			{
				fgets(cadena, 100, f);
				cadenaString = gcnew System::String(cadena);
				cadenaString = cadenaString->Trim();
				cadenaString = EliminarBlancos(cadenaString);
				int indinicio;
				indinicio = cadenaString->IndexOf(":")+1;
				resultadoFinal = cadenaString->Substring
(indinicio,cadenaString->Length-indinicio);
				break;
			}
		}
		int cerrar = fclose(f);
		return resultadoFinal;
}

System::String^ ObtenerFicheroDatosRespuestasFinales()
{
		System::String^ resultadoFinal;
		System::String^ cadenaString;
		FILE *f;
		char cadena[100];
		char *dato;
		fpos_t posicion;
```

```
	f = fopen("fichero.txt", "r");
	//Comprobamos que se abre bien el fichero
	if(!f)
	{
		resultadoFinal = "Fallo";
		return resultadoFinal;
	}


	//Recorremos el fichero linea por linea
	while (!feof(f)&&(fgets(cadena, 100, f)!=NULL))
	{
		dato = strstr(cadena, "------------");/*de esta forma solo
guardaremos la última aparición de "-------" para coger los datos
finales*/
		if(dato != NULL)
		{
			fgetpos(f, &posicion);
		}
	}
	fsetpos(f, &posicion);

	while (!feof(f) && (fgets(cadena, 100, f)!=NULL))
	{
		cadenaString = gcnew System::String(cadena);
		cadenaString = cadenaString->TrimEnd();
		resultadoFinal = resultadoFinal + cadenaString +
Environment::NewLine;
	}
	int cerrar = fclose(f);
	return resultadoFinal;
}

System::String^ ObtenerFicheroRespuestas()
{
	System::String^ resultadoFinal;
	System::String^ cadenaString;
	System::String^ cadenaString2;
	System::String^ cadenaString3;
	System::String^ tipoReg;
	FILE *f;
	char cadena[100],cadena1[100],cadena2[100],cadena3[100];
	char *dato;
	fpos_t posicion;

	f = fopen("fichero.txt", "r");
	//Comprobamos que se abre bien el fichero
	if(!f)
	{
		resultadoFinal = "Fallo";
		return resultadoFinal;
	}


	//Recorremos el fichero linea por linea
	while (!feof(f)&&(fgets(cadena, 100, f)!=NULL))
	{
		dato = strstr(cadena, "ANSWERS:");//buscamos la palabra
"ANSWERS:"
		if(dato != NULL)//Si está guardamos la posición
		{
```

```
                  fgetpos(f, &posicion);
            }
      }
      fsetpos(f, &posicion);//ponemos el fichero donde hemos guardado
antes

      while (!feof(f) && (fgets(cadena, 100, f)!=NULL))
      {
            strcpy(cadena1,cadena);

            fgets(cadena2, 100, f);
            // con esta cadena no haremos nada solamente almacena la
info de las ttls
            fgets(cadena3, 100, f);


            //Para la cadena1
            cadenaString = gcnew System::String(cadena1);
            cadenaString = cadenaString->Trim();
            //hay que tener en cuenta que esta cadena es la primera que
encuentra una fila en blanco
            if(cadenaString->Length == 0)
            {
                  break;
            }
            //con esta linea evitamos el simbolo "-> " que hay antes de
la información
            cadenaString = cadenaString->Substring(3);

            resultadoFinal = resultadoFinal + cadenaString +
Environment::NewLine;

            //Para la cadena2
            cadenaString2 = gcnew System::String(cadena2);
            cadenaString2 = cadenaString2->Trim();
            tipoReg = cadenaString2->Substring(0,cadenaString2->
IndexOf("="));
            tipoReg = tipoReg->Trim();

            //en el caso de registro del tipo nombre canónico
            if(tipoReg->Equals(tipoReg, "canonical name"))
            {
                  cadenaString2 ="Nombre Canónico : " + cadenaString2->
Substring(cadenaString2->IndexOf("=")+2);
            }
            //en el caso de registro del tipo dirección de internet
            if(tipoReg->Equals(tipoReg, "internet address"))
            {
                  cadenaString2 = "Dirección de Internet : " +
cadenaString2->Substring(cadenaString2->IndexOf("=")+2);
            }
            //en el caso de registro del tipo puntero
            if(tipoReg->Equals(tipoReg, "pointer"))
            {
                  cadenaString2 = "Puntero : " + cadenaString2->
Substring(cadenaString2->IndexOf("=")+2);
            }
            //en el caso de registro del tipo nombre canónico para las
consultas inversas
            if(tipoReg->Equals(tipoReg, "name"))
            {
```

```
                  cadenaString2 = "Nombre Canónico: " + cadenaString2->
Substring(cadenaString2->IndexOf("=")+2);
            }

            if(cadenaString2->Length == 0)
            {
                  break;
            }

            resultadoFinal = resultadoFinal + cadenaString2 +
Environment::NewLine + Environment::NewLine;

      }
      int cerrar = fclose(f);
      return resultadoFinal;
}
```

Fichero “Form.h”.

```
#pragma once
#include <string.h>
#include <windows.h>
#include <stdlib.h> // contiene la función system, posible solucion
para problemas en xp
#include <stdio.h>

namespace AnalizadorDNS {

      using namespace System;
      using namespace System::ComponentModel;
      using namespace System::Collections;
      using namespace System::Windows::Forms;
      using namespace System::Data;
      using namespace System::Drawing;

      /// <summary>
      /// Resumen de Form1
      ///
      /// ADVERTENCIA: si cambia el nombre de esta clase, deberá
cambiar la
      ///          propiedad 'Nombre de archivos de recursos' de la
herramienta de compilación de recursos administrados
      ///          asociada con todos los archivos .resx de los que
depende esta clase. De lo contrario,
      ///          los diseñadores no podrán interactuar correctamente
con los
      ///          recursos adaptados asociados con este formulario.
      /// </summary>
      public ref class Form1 : public System::Windows::Forms::Form
      {
      public:
            Form1(void)
            {
                  InitializeComponent();
                  //
                  //TODO: agregar código de constructor aquí
                  //
            }
```

```
        protected:
              /// <summary>
              /// Limpiar los recursos que se estén utilizando.
              /// </summary>
              ~Form1()
              {
                    if (components)
                    {
                          delete components;
                    }
              }
        private: System::Windows::Forms::Button^  BtnBuscar;
        private: System::Windows::Forms::TabControl^  TabCntrPanel;
        private: System::Windows::Forms::TabPage^  TabPagPestaña1;
        private: System::Windows::Forms::TabPage^  TabPagPestaña2;
        protected:

        protected:




        private: System::Windows::Forms::Button^  BtnSalir;
        private: System::Windows::Forms::Label^  LblServidor;
        private: System::Windows::Forms::Label^  LblConsulta;
        private: System::Windows::Forms::MaskedTextBox^  MTbxIPConsulta;









        private: System::Windows::Forms::MaskedTextBox^  MTbxIPServidor;









        private: System::Windows::Forms::GroupBox^  GrbxConsulta;
        private: System::Windows::Forms::RadioButton^
RdbNombreConsulta;



        private: System::Windows::Forms::RadioButton^  RdbDirIPConsulta;

        private: System::Windows::Forms::TextBox^  TxtBxNombreConsulta;
        private: System::Windows::Forms::TextBox^  TxtBxNombreServidor;
        private: System::Windows::Forms::Label^  LblCabecera;
        private: System::Windows::Forms::Label^  LblQuestions;
        private: System::Windows::Forms::Label^  LblRespuestas;
        private: System::Windows::Forms::Label^
LblRegistrosAdicionales;
```

```
      private: System::Windows::Forms::Label^
LblResgitrosAutorizados;
      private: System::Windows::Forms::TextBox^  TxtBxRespuestas;

      private: System::Windows::Forms::Label^  LblNumQuestions;
      private: System::Windows::Forms::Label^  LblNumRegAdicionales;
      private: System::Windows::Forms::Label^  LblNumRegAutorizados;
      private: System::Windows::Forms::Label^  LblNumRespuestas;









      protected:

      private:
            /// <summary>
            /// Variable del diseñador requerida.
            /// </summary>
            System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
            /// <summary>
            /// Método necesario para admitir el Diseñador. No se puede
modificar
            /// el contenido del método con el editor de código.
            /// </summary>
            void InitializeComponent(void)
            {
                  System::ComponentModel::ComponentResourceManager^
resources = (gcnew System::ComponentModel::ComponentResourceManager
(Form1::typeid));
                  this->BtnBuscar = (gcnew System::Windows::Forms::
Button());
                  this->TabCntrPanel = (gcnew System::Windows::
Forms::TabControl());
                  this->TabPagPestaña1 = (gcnew System::Windows::
Forms::TabPage());
                  this->LblNumRegAdicionales = (gcnew System::Windows::
Forms::Label());
                  this->LblNumRegAutorizados = (gcnew System::Windows::
Forms::Label());
                  this->LblNumRespuestas = (gcnew System::Windows::
Forms::Label());
                  this->LblNumQuestions = (gcnew System::Windows::
Forms::Label());
                  this->LblRegistrosAdicionales = (gcnew System::
Windows::Forms::Label());
                  this->LblResgitrosAutorizados = (gcnew System::
Windows::Forms::Label());
                  this->LblRespuestas = (gcnew System::Windows::
Forms::Label());
```

```
                    this->LblQuestions = (gcnew System::Windows::
Forms::Label());
                    this->LblCabecera = (gcnew System::Windows::
Forms::Label());
                    this->GrbxConsulta = (gcnew System::Windows::
Forms::GroupBox());
                    this->RdbNombreConsulta = (gcnew System::Windows::
Forms::RadioButton());
                    this->RdbDirIPConsulta = (gcnew System::Windows::
Forms::RadioButton());
                    this->MTbxIPServidor = (gcnew System::Windows::
Forms::MaskedTextBox());
                    this->MTbxIPConsulta = (gcnew System::Windows::
Forms::MaskedTextBox());
                    this->LblConsulta = (gcnew System::Windows::
Forms::Label());
                    this->LblServidor = (gcnew System::Windows::
Forms::Label());
                    this->TxtBxNombreConsulta = (gcnew System::Windows::
Forms::TextBox());
                    this->TxtBxNombreServidor = (gcnew System::Windows::
Forms::TextBox());
                    this->TabPagPestaña2 = (gcnew System::Windows::
Forms::TabPage());
                    this->TxtBxRespuestas = (gcnew System::Windows::
Forms::TextBox());
                    this->BtnSalir = (gcnew System::Windows::
Forms::Button());
                    this->TabCntrPanel->SuspendLayout();
                    this->TabPagPestaña1->SuspendLayout();
                    this->GrbxConsulta->SuspendLayout();
                    this->TabPagPestaña2->SuspendLayout();
                    this->SuspendLayout();
                    //
                    // BtnBuscar
                    //
                    this->BtnBuscar->AutoSize = true;
                    this->BtnBuscar->Location = System::Drawing::Point
(378, 232);
                    this->BtnBuscar->Name = L"BtnBuscar";
                    this->BtnBuscar->Size = System::Drawing::Size(88,
27);
                    this->BtnBuscar->TabIndex = 3;
                    this->BtnBuscar->Text = L"Buscar";
                    this->BtnBuscar->UseVisualStyleBackColor = true;
                    this->BtnBuscar->Click += gcnew System::EventHandler
(this, &Form1::button1_Click);
                    //
                    // TabCntrPanel
                    //
                    this->TabCntrPanel->Controls->Add(this->
TabPagPestaña1);
                    this->TabCntrPanel->Controls->Add(this->
TabPagPestaña2);
                    this->TabCntrPanel->Font = (gcnew System::Drawing::
Font(L"Nina", 8.25F, System::Drawing::FontStyle::Regular, System::
Drawing::GraphicsUnit::Point,
                         static_cast<System::Byte>(0)));
                    this->TabCntrPanel->Location = System::Drawing::
Point(12, 12);
                    this->TabCntrPanel->Name = L"TabCntrPanel";
```

```
                    this->TabCntrPanel->SelectedIndex = 0;
                    this->TabCntrPanel->Size = System::Drawing::Size(559,
214);
                    this->TabCntrPanel->SizeMode = System::Windows::
Forms::TabSizeMode::FillToRight;
                    this->TabCntrPanel->TabIndex = 1;
                    //
                    // TabPagPestaña1
                    //
                    this->TabPagPestaña1->BackColor = System::Drawing::
Color::AliceBlue;
                    this->TabPagPestaña1->Controls->Add(this->
LblNumRegAdicionales);
                    this->TabPagPestaña1->Controls->Add(this->
LblNumRegAutorizados);
                    this->TabPagPestaña1->Controls->Add(this->
LblNumRespuestas);
                    this->TabPagPestaña1->Controls->Add(this->
LblNumQuestions);
                    this->TabPagPestaña1->Controls->Add(this->
LblRegistrosAdicionales);
                    this->TabPagPestaña1->Controls->Add(this->
LblResgitrosAutorizados);
                    this->TabPagPestaña1->Controls->Add(this->
LblRespuestas);
                    this->TabPagPestaña1->Controls->Add(this->
LblQuestions);
                    this->TabPagPestaña1->Controls->Add(this->
LblCabecera);
                    this->TabPagPestaña1->Controls->Add(this->
GrbxConsulta);
                    this->TabPagPestaña1->Controls->Add(this->
MTbxIPServidor);
                    this->TabPagPestaña1->Controls->Add(this->
MTbxIPConsulta);
                    this->TabPagPestaña1->Controls->Add(this->
LblConsulta);
                    this->TabPagPestaña1->Controls->Add(this->
LblServidor);
                    this->TabPagPestaña1->Controls->Add(this->
TxtBxNombreConsulta);
                    this->TabPagPestaña1->Controls->Add(this->
TxtBxNombreServidor);
                    this->TabPagPestaña1->Location =
System::Drawing::Point(4, 22);
                    this->TabPagPestaña1->Name = L"TabPagPestaña1";
                    this->TabPagPestaña1->Padding = System::Windows::
Forms::Padding(3);
                    this->TabPagPestaña1->Size = System::Drawing::
Size(551, 188);
                    this->TabPagPestaña1->TabIndex = 0;
                    this->TabPagPestaña1->Text = L"Configuración";
                    //
                    // LblNumRegAdicionales
                    //
                    this->LblNumRegAdicionales->AutoSize = true;
                    this->LblNumRegAdicionales->Font = (gcnew System::
Drawing::Font(L"Sylfaen", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(0)));
```

```
                    this->LblNumRegAdicionales->Location = System::
Drawing::Point(491, 155);
                    this->LblNumRegAdicionales->Name =
L"LblNumRegAdicionales";
                    this->LblNumRegAdicionales->Size = System::
Drawing::Size(18, 19);
                    this->LblNumRegAdicionales->TabIndex = 23;
                    this->LblNumRegAdicionales->Text = L"0";
                    this->LblNumRegAdicionales->Click += gcnew System::
EventHandler(this, &Form1::LblNumRegAdicionales_Click);
                    //
                    // LblNumRegAutorizados
                    //
                    this->LblNumRegAutorizados->AutoSize = true;
                    this->LblNumRegAutorizados->Font = (gcnew System::
Drawing::Font(L"Sylfaen", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->LblNumRegAutorizados->Location = System::
Drawing::Point(495, 124);
                    this->LblNumRegAutorizados->Name =
L"LblNumRegAutorizados";
                    this->LblNumRegAutorizados->Size = System::Drawing
::Size(18, 19);
                    this->LblNumRegAutorizados->TabIndex = 22;
                    this->LblNumRegAutorizados->Text = L"0";
                    //
                    // LblNumRespuestas
                    //
                    this->LblNumRespuestas->AutoSize = true;
                    this->LblNumRespuestas->Font = (gcnew System::
Drawing::Font(L"Sylfaen", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->LblNumRespuestas->Location = System::Drawing
::Point(141, 155);
                    this->LblNumRespuestas->Name = L"LblNumRespuestas";
                    this->LblNumRespuestas->Size =
System::Drawing::Size(18, 19);
                    this->LblNumRespuestas->TabIndex = 21;
                    this->LblNumRespuestas->Text = L"0";
                    //
                    // LblNumQuestions
                    //
                    this->LblNumQuestions->AutoSize = true;
                    this->LblNumQuestions->Font = (gcnew System::
Drawing::Font(L"Sylfaen", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->LblNumQuestions->Location = System::Drawing::
Point(131, 124);
                    this->LblNumQuestions->Name = L"LblNumQuestions";
                    this->LblNumQuestions->Size = System::Drawing::
Size(18, 19);
                    this->LblNumQuestions->TabIndex = 20;
                    this->LblNumQuestions->Text = L"0";
                    //
                    // LblRegistrosAdicionales
                    //
                    this->LblRegistrosAdicionales->AutoSize = true;
```

```
                    this->LblRegistrosAdicionales->Font = (gcnew
System::Drawing::Font(L"Sylfaen", 11.25F, System::Drawing::
FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->LblRegistrosAdicionales->Location = System::
Drawing::Point(302, 155);
                    this->LblRegistrosAdicionales->Name =
L"LblRegistrosAdicionales";
                    this->LblRegistrosAdicionales->Size =
System::Drawing::Size(183, 19);
                    this->LblRegistrosAdicionales->TabIndex = 19;
                    this->LblRegistrosAdicionales->Text = L"- Registros
Adicionales :";
                    //
                    // LblResgitrosAutorizados
                    //
                    this->LblResgitrosAutorizados->AutoSize = true;
                    this->LblResgitrosAutorizados->Font = (gcnew
System::Drawing::Font(L"Sylfaen", 11.25F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->LblResgitrosAutorizados->Location =
System::Drawing::Point(302, 124);
                    this->LblResgitrosAutorizados->Name =
L"LblResgitrosAutorizados";
                    this->LblResgitrosAutorizados->Size =
System::Drawing::Size(188, 19);
                    this->LblResgitrosAutorizados->TabIndex = 18;
                    this->LblResgitrosAutorizados->Text = L"- Registros
Autorizados :";
                    //
                    // LblRespuestas
                    //
                    this->LblRespuestas->AutoSize = true;
                    this->LblRespuestas->Font = (gcnew System::Drawing::
Font(L"Sylfaen", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->LblRespuestas->Location = System::Drawing::
Point(25, 155);
                    this->LblRespuestas->Name = L"LblRespuestas";
                    this->LblRespuestas->Size = System::Drawing::
Size(110, 19);
                    this->LblRespuestas->TabIndex = 17;
                    this->LblRespuestas->Text = L"- Respuestas :";
                    //
                    // LblQuestions
                    //
                    this->LblQuestions->AutoSize = true;
                    this->LblQuestions->Font = (gcnew System::Drawing::
Font(L"Sylfaen", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->LblQuestions->Location = System::Drawing::
Point(25, 124);
                    this->LblQuestions->Name = L"LblQuestions";
                    this->LblQuestions->Size = System::Drawing::Size(100,
19);
                    this->LblQuestions->TabIndex = 16;
                    this->LblQuestions->Text = L"- Questions :\r\n";
```

```
                    //
                    // LblCabecera
                    //
                    this->LblCabecera->AutoSize = true;
                    this->LblCabecera->Font = (gcnew System::Drawing::
Font(L"Sylfaen", 12, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                           static_cast<System::Byte>(0)));
                    this->LblCabecera->ForeColor = System::Drawing::
Color::MidnightBlue;
                    this->LblCabecera->Location = System::Drawing::
Point(15, 102);
                    this->LblCabecera->Name = L"LblCabecera";
                    this->LblCabecera->Size = System::Drawing::Size(77,
22);
                    this->LblCabecera->TabIndex = 15;
                    this->LblCabecera->Text = L"Cabecera";
                    //
                    // GrbxConsulta
                    //
                    this->GrbxConsulta->Controls->Add(this->
RdbNombreConsulta);
                    this->GrbxConsulta->Controls->Add(this->
RdbDirIPConsulta);
                    this->GrbxConsulta->Location = System::Drawing::
Point(382, 6);
                    this->GrbxConsulta->Name = L"GrbxConsulta";
                    this->GrbxConsulta->Size = System::Drawing::Size(163,
45);
                    this->GrbxConsulta->TabIndex = 3;
                    this->GrbxConsulta->TabStop = false;
                    this->GrbxConsulta->Text = L"Tipo de Dato Consulta";
                    //
                    // RdbNombreConsulta
                    //
                    this->RdbNombreConsulta->AutoSize = true;
                    this->RdbNombreConsulta->Checked = true;
                    this->RdbNombreConsulta->Location = System::Drawing::
Point(97, 19);
                    this->RdbNombreConsulta->Name = L"RdbNombreConsulta";
                    this->RdbNombreConsulta->Size = System::Drawing::
Size(58, 17);
                    this->RdbNombreConsulta->TabIndex = 3;
                    this->RdbNombreConsulta->TabStop = true;
                    this->RdbNombreConsulta->Text = L"Nombre";
                    this->RdbNombreConsulta->UseVisualStyleBackColor =
true;
                    //
                    // RdbDirIPConsulta
                    //
                    this->RdbDirIPConsulta->AutoSize = true;
                    this->RdbDirIPConsulta->Location = System::Drawing::
Point(8, 19);
                    this->RdbDirIPConsulta->Name = L"RdbDirIPConsulta";
                    this->RdbDirIPConsulta->Size = System::Drawing::
Size(76, 17);
                    this->RdbDirIPConsulta->TabIndex = 4;
                    this->RdbDirIPConsulta->Text = L"Dirección IP";
                    this->RdbDirIPConsulta->UseVisualStyleBackColor =
true;
```

```
                    this->RdbDirIPConsulta->CheckedChanged += gcnew
System::EventHandler(this, &Form1::RdbDirIPConsulta_CheckedChanged);
                    //
                    // MTbxIPServidor
                    //
                    this->MTbxIPServidor->BackColor = System::Drawing::
SystemColors::Window;
                    this->MTbxIPServidor->Font = (gcnew System::Drawing::
Font(L"Nina", 8.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->MTbxIPServidor->Location = System::Drawing::
Point(390, 57);
                    this->MTbxIPServidor->Name = L"MTbxIPServidor";
                    this->MTbxIPServidor->ReadOnly = true;
                    this->MTbxIPServidor->Size = System::Drawing::
Size(100, 21);
                    this->MTbxIPServidor->TabIndex = 8;
                    this->MTbxIPServidor->KeyPress += gcnew System::
Windows::Forms::KeyPressEventHandler(this, &Form1::
MTbxIPServidor1_KeyPress);
                    //
                    // MTbxIPConsulta
                    //
                    this->MTbxIPConsulta->Font = (gcnew System::Drawing::
Font(L"Nina", 8.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->MTbxIPConsulta->Location = System::Drawing::
Point(100, 22);
                    this->MTbxIPConsulta->Mask = L"009.009.009.009";
                    this->MTbxIPConsulta->Name = L"MTbxIPConsulta";
                    this->MTbxIPConsulta->Size = System::Drawing::
Size(100, 21);
                    this->MTbxIPConsulta->TabIndex = 2;
                    this->MTbxIPConsulta->Visible = false;
                    //
                    // LblConsulta
                    //
                    this->LblConsulta->AutoSize = true;
                    this->LblConsulta->Font = (gcnew System::Drawing::
Font(L"Sylfaen", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->LblConsulta->Location = System::Drawing::
Point(24, 23);
                    this->LblConsulta->Name = L"LblConsulta";
                    this->LblConsulta->Size = System::Drawing::Size(70,
19);
                    this->LblConsulta->TabIndex = 2;
                    this->LblConsulta->Text = L"Consulta";
                    //
                    // LblServidor
                    //
                    this->LblServidor->AutoSize = true;
                    this->LblServidor->Font = (gcnew System::Drawing::
Font(L"Sylfaen", 11.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                          static_cast<System::Byte>(0)));
                    this->LblServidor->Location = System::Drawing::
Point(24, 59);
```

```
                    this->LblServidor->Name = L"LblServidor";
                    this->LblServidor->Size = System::Drawing::Size(68,
19);
                    this->LblServidor->TabIndex = 0;
                    this->LblServidor->Text = L"Servidor";
                    //
                    // TxtBxNombreConsulta
                    //
                    this->TxtBxNombreConsulta->Font = (gcnew System::
Drawing::Font(L"Arial Black", 8.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(0)));
                    this->TxtBxNombreConsulta->Location = System::
Drawing::Point(100, 22);
                    this->TxtBxNombreConsulta->Name =
L"TxtBxNombreConsulta";
                    this->TxtBxNombreConsulta->Size = System::Drawing::
Size(276, 23);
                    this->TxtBxNombreConsulta->TabIndex = 1;
                    //
                    // TxtBxNombreServidor
                    //
                    this->TxtBxNombreServidor->BackColor = System::
Drawing::SystemColors::Window;
                    this->TxtBxNombreServidor->Font = (gcnew System::
Drawing::Font(L"Arial Black", 8.25F, System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(0)));
                    this->TxtBxNombreServidor->Location = System::
Drawing::Point(100, 55);
                    this->TxtBxNombreServidor->Name =
L"TxtBxNombreServidor";
                    this->TxtBxNombreServidor->ReadOnly = true;
                    this->TxtBxNombreServidor->Size = System::Drawing::
Size(276, 23);
                    this->TxtBxNombreServidor->TabIndex = 7;
                    //
                    // TabPagPestaña2
                    //
                    this->TabPagPestaña2->BackColor = System::Drawing::
Color::AliceBlue;
                    this->TabPagPestaña2->Controls->Add(this->
TxtBxRespuestas);
                    this->TabPagPestaña2->Location = System::Drawing::
Point(4, 22);
                    this->TabPagPestaña2->Name = L"TabPagPestaña2";
                    this->TabPagPestaña2->Padding = System::Windows::
Forms::Padding(3);
                    this->TabPagPestaña2->Size = System::Drawing::
Size(551, 188);
                    this->TabPagPestaña2->TabIndex = 1;
                    this->TabPagPestaña2->Text = L"Respuestas";
                    //
                    // TxtBxRespuestas
                    //
                    this->TxtBxRespuestas->AcceptsReturn = true;
                    this->TxtBxRespuestas->BackColor = System::Drawing::
SystemColors::ActiveCaptionText;
                    this->TxtBxRespuestas->Location = System::Drawing::
Point(4, 5);
                    this->TxtBxRespuestas->Multiline = true;
```

```
                this->TxtBxRespuestas->Name = L"TxtBxRespuestas";
                this->TxtBxRespuestas->ReadOnly = true;
                this->TxtBxRespuestas->ScrollBars = System::Windows::
Forms::ScrollBars::Both;
                this->TxtBxRespuestas->Size = System::Drawing::
Size(541, 181);
                this->TxtBxRespuestas->TabIndex = 0;
                //
                // BtnSalir
                //
                this->BtnSalir->AutoSize = true;
                this->BtnSalir->Location = System::Drawing::
Point(479, 232);
                this->BtnSalir->Name = L"BtnSalir";
                this->BtnSalir->Size = System::Drawing::Size(88, 27);
                this->BtnSalir->TabIndex = 4;
                this->BtnSalir->Text = L"Salir";
                this->BtnSalir->UseVisualStyleBackColor = true;
                this->BtnSalir->Click += gcnew System::
EventHandler(this, &Form1::BtnSalir_Click);
                //
                // Form1
                //
                this->AllowDrop = true;
                this->AutoScaleDimensions = System::Drawing::SizeF(6,
13);
                this->AutoScaleMode = System::Windows::
Forms::AutoScaleMode::Font;
                this->AutoSize = true;
                this->BackColor = System::Drawing::
Color::LightSlateGray;
                this->ClientSize = System::Drawing::Size(583, 266);
                this->Controls->Add(this->BtnSalir);
                this->Controls->Add(this->TabCntrPanel);
                this->Controls->Add(this->BtnBuscar);
                this->Icon = (cli::safe_cast<System::Drawing::Icon^
>(resources->GetObject(L"$this.Icon")));
                this->MaximizeBox = false;
                this->Name = L"Form1";
                this->StartPosition = System::Windows::
Forms::FormStartPosition::CenterScreen;
                this->Text = L"Analizador DNS";
                this->TabCntrPanel->ResumeLayout(false);
                this->TabPagPestaña1->ResumeLayout(false);
                this->TabPagPestaña1->PerformLayout();
                this->GrbxConsulta->ResumeLayout(false);
                this->GrbxConsulta->PerformLayout();
                this->TabPagPestaña2->ResumeLayout(false);
                this->TabPagPestaña2->PerformLayout();
                this->ResumeLayout(false);
                this->PerformLayout();

            }
#pragma endregion
      private: System::Void button1_Click(System::Object^  sender,
System::EventArgs^  e) {
                         System::String ^ consulta;

                         char sentenciaArray[150];
                         System::String^ sentenciaString;
                         System::String^ resultado;
```

```
                        if( RdbNombreConsulta->Checked)
                        {
                             consulta = TxtBxNombreConsulta->Text;
                        }else
                        {
                             consulta = MTbxIPConsulta->Text;
                        }

                        consulta = EliminarBlancos(consulta);

                        //Reseteamos los datos de la interfaz
                        ResetearInterfaz();


                        sentenciaString += "nslookup -debug ";
                        sentenciaString += consulta;
                        sentenciaString += " > fichero.txt";

                        if(ComprobarFormato(consulta) == true)
                        {

                              ConvertirACharArray(sentenciaString,
sentenciaArray);

                              system(sentenciaArray);

                              if(ComprobarResultados() == true)
                              {
                                    //Obtenemos los datos de las
Questions
                                    resultado =
ObtenerFicheroQuestions();

                                    LblNumQuestions->Text = resultado;

                                    //Obtenemos los datos del num de
Respuestas
                                    resultado =
ObtenerFicheroNumRespuestas();

                                    LblNumRespuestas->Text =
resultado;

                                    //Obtenemos los datos de los
Registros Autorizados
                                    resultado =
ObtenerFicheroRegistrosAutorizados();

                                    LblNumRegAutorizados->Text =
resultado;

                                    //Obtenemos los datos de los
Registros Adicionales
                                    resultado =
ObtenerFicheroRegistrosAdicionales();

                                    LblNumRegAdicionales->Text =
resultado;
```

```
                                    //Obtenemos el nombre del Servidor
                                    resultado =
ObtenerFicheroDatosServidorNombre();

                                    TxtBxNombreServidor->Text =
resultado;

                                    //Obtenemos la direccion IP del
Servidor
                                    resultado =
ObtenerFicheroDatosServidorIP();

                                    MTbxIPServidor->Text = resultado;

                                    //Obtenemos las respuestas a la
consulta
                                    resultado =
ObtenerFicheroRespuestas();

                                    resultado = resultado +
Environment::NewLine + ObtenerFicheroDatosRespuestasFinales();

                                    TxtBxRespuestas->Text = resultado;
                              }
                        }
                  }
      private: System::Void BtnSalir_Click(System::Object^  sender,
System::EventArgs^  e)
                  {
                        Close();
                  }
      private: System::Void MTbxIPServidor1_KeyPress(System::Object^
sender, System::Windows::Forms::KeyPressEventArgs^  e) {

            }


private: System::Void RdbDirIPConsulta_CheckedChanged(System::Object^
sender, System::EventArgs^  e) {
                  if (RdbDirIPConsulta->Checked==true)
                  {
                        TxtBxNombreConsulta->Visible=false;
                        MTbxIPConsulta->Text="";
                        MTbxIPConsulta->Visible=true;
                  }
                  else {
                        TxtBxNombreConsulta->Text="";
                        TxtBxNombreConsulta->Visible=true;
                        MTbxIPConsulta->Visible=false;
                  }
            }
private: System::Void LblNumRegAdicionales_Click(System::Object^
sender, System::EventArgs^  e) {
            }
//Función para comprobar el formato de las consultas
private: bool ComprobarFormato(System::String^ consulta)
            {
                  System::String^ respuesta;

                  //cuando la consulta está vacia
                  if ((consulta->IsNullOrEmpty(consulta)))
```

```
                    {
                        MessageBox::Show("Por favor introduzca el texto
para la
consulta","Error",MessageBoxButtons::OK,MessageBoxIcon::Error);
                        return false;
                    }

                    //cuando no hay ninguna IP para la consulta
                    if (consulta->Equals("..."))
                    {
                        MessageBox::Show("Por favor introduzca la
dirección IP para la
consulta","Error",MessageBoxButtons::OK,MessageBoxIcon::Error);
                        return false;
                    }
                    return true;
                }

private: bool ComprobarResultados()
                {
                    System::String^ respuesta =
ObtenerFicheroDatosRespuestasFinales();
                    System::String^ numrespuestas =
ObtenerFicheroNumRespuestas();
                   //Comprobamos si hay respuesta final
                    if ((respuesta->IsNullOrEmpty(respuesta)) ||
(numrespuestas->Equals("0")))
                    {
                        MessageBox::Show("La consulta no ha devuelto
ningún
resultado","Error",MessageBoxButtons::OK,MessageBoxIcon::Error);
                        return false;
                    }

                    return true;
                }

 private: void ResetearInterfaz()
                {
                    //Reseteamos los datos de la interfaz
                    LblNumQuestions->Text = "0";
                    LblNumRegAdicionales->Text = "0";
                    LblNumRegAutorizados->Text = "0";
                    LblNumRespuestas->Text = "0";

                    TxtBxNombreServidor->Text = "";
                    MTbxIPServidor->Text = "";

                    TxtBxRespuestas->Text = "";
                }
};
}
```

Fichero “stdafx.h”

```
// stdafx.h: archivo de inclusión de los archivos de inclusión
estándar del sistema
// o archivos de inclusión específicos de un proyecto utilizados
frecuentemente,
// pero rara vez modificados
```

```
System::String^ EliminarBlancos(System::String^ cadena2);

void ConvertirACharArray(System::String^ cadena1, char *cad1);

System::String^ ObtenerFicheroQuestions();

System::String^ ObtenerFicheroNumRespuestas();

System::String^ ObtenerFicheroRegistrosAutorizados();

System::String^ ObtenerFicheroRegistrosAdicionales();

System::String^ ObtenerFicheroDatosServidorNombre();

System::String^ ObtenerFicheroDatosServidorIP();

System::String^ ObtenerFicheroDatosRespuestasFinales();

System::String^ ObtenerFicheroRespuestas();

bool ComprobarFormato(System::String^ consulta);

bool ComprobarResultados();

void ResetearInterfaz();
// TODO: mencionar aquí los encabezados adicionales que el programa
necesita
```

Fichero “stdafx.cpp”

```
// stdafx.cpp: archivo de código fuente que contiene sólo las
inclusiones estándar
// AnalizadorDNS.pch será el encabezado precompilado
// stdafx.obj contiene la información de tipos precompilada

#include "stdafx.h"
```

Fichero “AssemblyInfo.cpp”

```
#include "stdafx.h"

using namespace System;
using namespace System::Reflection;
using namespace System::Runtime::CompilerServices;
using namespace System::Runtime::InteropServices;
using namespace System::Security::Permissions;

//
// La información general sobre un ensamblado se controla mediante el
siguiente
// conjunto de atributos. Cambie estos atributos para modificar la
información
// asociada con un ensamblado.
```

```
//
[assembly:AssemblyTitleAttribute("AnalizadorDNS")];
[assembly:AssemblyDescriptionAttribute("")];
[assembly:AssemblyConfigurationAttribute("")];
[assembly:AssemblyCompanyAttribute("")];
[assembly:AssemblyProductAttribute("AnalizadorDNS")];
[assembly:AssemblyCopyrightAttribute("Copyright (c)  2011")];
[assembly:AssemblyTrademarkAttribute("")];
[assembly:AssemblyCultureAttribute("")];

//
// La información de versión de un ensamblado consta de los cuatro
valores siguientes:
//
//      Versión principal
//      Versión secundaria
//      Número de versión de compilación
//      Revisión
//
// Puede especificar todos los valores o usar los valores
predeterminados de número de versión de compilación y de revisión
// mediante el asterisco ('*'), como se muestra a continuación:

[assembly:AssemblyVersionAttribute("1.0.*")];

[assembly:ComVisible(false)];

[assembly:CLSCompliantAttribute(true)];

[assembly:SecurityPermission(SecurityAction::RequestMinimum,
UnmanagedCode = true)];
```