UNIVERSIDAD POLITECNICA DE VALENCIA

etsinf

Escola Tècnica
Superior d'Enginyeria
Informàtica

## Escola Tècnica Superior d'Enginyeria Informàtica

PROYECTO FIN DE CARRERA / PROJECTE FI DE CARRERA

**Análisis de tráfico DNS utilizando la Web**

Para optar a la titulación de / per a optar a la titulació de

**Ingeniería Técnica en Informática de Sistemas (ITIS)**

presentado por / presentat per

**José Ignacio Rodríguez de la Torre**

Dirigido / tutorizado por

dirigit / tutoritzat per

**Joaquín Gracia Morán**

Valencia a 28 de julio de 2011

## ÍNDICE

# 1 INTRODUCCIÓN

El estudio del funcionamiento de diferentes protocolos de comunicaciones suele ser una tarea habitual entre Ingenieros Informáticos. Este estudio, que se realiza principalmente para resolver problemas, puede servir también para obtener un conocimiento más profundo tanto del comportamiento de los diferentes protocolos de comunicaciones como de la red a la que se está conectado.

Actualmente, **Internet** es la red de comunicaciones preponderante a nivel mundial. Internet es una red de área amplia de computadores que interconecta millones de dispositivos en todo el mundo. Estos dispositivos es lo que conocemos como ***Hosts*** o ***Sistemas Terminales***, y pueden ser PCs, estaciones de trabajo, PDAs, portátiles, servidores, etc. Es decir, Internet está formada por muchas redes de datos interconectadas. Para poder identificar a los diferentes equipos conectados a Internet, se utiliza el protocolo de comunicaciones denominado **Sistema de Nombres de Dominio** o **DNS**.

Aunque un host queda perfectamente definido con su dirección IP, los usuarios prefieren asignar a los equipos nombres fáciles de pronunciar y recordar. El protocolo **DNS** proporciona un mecanismo para traducir los nombres de dominio con los que se nombran los recursos disponibles en Internet en direcciones IP utilizables por los protocolos de comunicación y las aplicaciones.

El servicio DNS se ha convertido en un servicio crítico, ya que si no funciona o no está bien configurado, no es posible conocer las direcciones IP que hay tras los nombres de los servidores y éstos no son, por lo tanto, accesibles.

La comprobación del correcto funcionamiento de las consultas DNS es uno de los primeros puntos de comprobación ante cualquier incidencia que afecte a la conectividad de un sistema de información.

Las herramientas disponibles para hacer esta comprobación son, tradicionalmente, utilidades de línea de comandos del sistema operativo. Esto, combinado con la variedad de parámetros que se pueden utilizar, provoca que para un usuario no acostumbrado sea dificultoso realizar estas comprobaciones.

El presente proyecto propone el desarrollo y la implementación de una herramienta que permita realizar de forma gráfica consultas DNS, de tal manera que ésta herramienta pueda ser utilizada por usuarios que no tengan un conocimiento específico del protocolo DNS o las herramientas asociadas con este protocolo.

# 2 JUSTIFICACIÓN DEL PROBLEMA

En este apartado se va a presentar la problemática existente para el estudio del protocolo DNS, la cual justifica el desarrollo e implementación de la herramienta presentada en este proyecto fin de carrera.

## 2.1 Introducción

El Sistema de Nombres de Dominio o DNS (del inglés *Domain Name System*) es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada. Este sistema asocia información variada con nombres de dominios asignados a cada uno de los participantes. Su función más importante es traducir (o resolver) nombres inteligibles para los humanos en identificadores binarios asociados con los equipos conectados a la red, con el fin de de poder localizar y direccionar estos equipos mundialmente.

El sistema DNS utiliza una base de datos distribuida y jerárquica que almacena información asociada a nombres de dominio en redes como Internet. Aunque como base de datos el DNS es capaz de asociar diferentes tipos de información a cada nombre, los usos más comunes son la asignación de nombres de dominio a direcciones IP y la localización de los servidores de correo electrónico de cada dominio.

El DNS facilita el acceso a los ordenadores conectados a Internet al permitir utilizar una cadena familiar de letras (el "nombre de dominio") en lugar de un identificador binario (la dirección IP). Este sistema permite identificar de forma fácil a los distintos equipos conectados a una red. Por ejemplo, si introducimos en un navegador web la dirección www.upv.es, el ordenador local se conectará al servidor DNS y averiguará que el servidor del sitio web de la UPV tiene la dirección IP 158.42.4.23. El ordenador local puede ahora conectarse con ese servidor para poder navegar a través del sitio web.

Para saber si el DNS está realizando correctamente la traducción entre nombre de *host* y dirección IP, normalmente se utiliza la herramienta nslookup, que funciona tanto en Windows como en UNIX. El nombre *nslookup* significa *name server lookup*.

*Nslookup* utiliza por defecto el DNS local del sistema operativo para realizar sus consultas. De este modo, esta herramienta se configura automáticamente por el contexto del propio sistema operativo. *nslookup* es una herramienta administrativa de línea de comandos que se utiliza para probar y solucionar problemas de los servidores DNS.

*nslookup* envía consultas a los servidores DNS. Dispone de dos modos: interactivo y no interactivo. El modo interactivo permite al usuario ponerse en contacto con los servidores para obtener información sobre varios hosts y dominios o para mostrar una lista de hosts en un dominio. El modo no interactivo se utiliza para mostrar sólo el nombre y la información solicitada para un servidor o dominio.

En resumen, *nslookup* permite obtener la dirección IP conociendo el nombre del host, y viceversa, así como cualquier otro registro especifico DNS.

## 2.2 Herramientas disponibles

En internet existen algunas webs que proporcionan herramientas visuales para la resolución de DNS. Un análisis de las mismas ha reflejado las siguientes carencias:

- Son pocas y poco accesibles.
- No todas permiten seleccionar el tipo de registro que queremos consultar.
- No todas permiten seleccionar el servidor DNS al que se quiere dirigir la consulta.
- Algunas están saturadas de publicidad en la página web.
- Las más completas son de pago.
- En algunos casos en realidad, realizan *traceroutes*[1], por lo que no nos dan la información que ofrece la herramienta *nslookup*.
- Están basadas en WEB, y las direcciones que se muestran parten del servidor WEB de la compañía y no desde el equipo desde el que nos interesa hacer la comprobación.

Sin duda, el último punto es el más significativo de todos, ya que al lanzar la consulta desde el servidor web que aloja la herramienta si bien es posible obtener información sobre el dominio que interese, no es posible verificar si el servidor DNS local del usuario tiene realmente acceso a esa información.

Algunos ejemplos de las herramientas que han sido analizadas son:

[1] El comando *traceroute* permite obtener la ruta entre el *host* de origen (normalmente desde el que se hace la consulta) y un *host* de destino.

**DNSQUERIES**

http://www.dnsqueries.com/en/dns_lookup.php

Contiene publicidad y no permite seleccionar el servidor DNS al que se quiere consultar. La Figura 1 muestra la ventana que se obtiene al conectarse, y en la que se ha eliminado gran parte de la publicidad existente en esta herramienta.

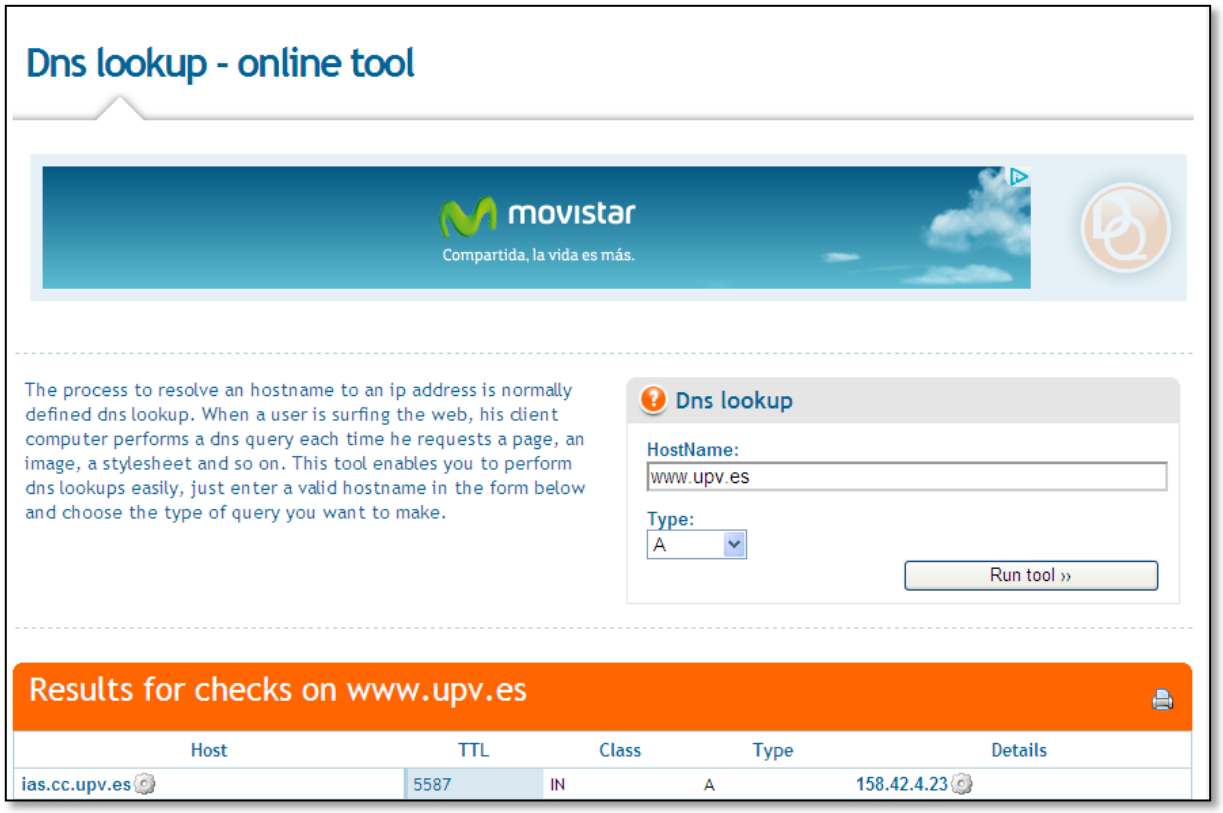



**Figura 1. DNSQUERIES.**

**DNSstuff**

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

Sin publicidad y pudiendo seleccionar el tipo de registro a consultar, pero de pago. El interfaz se puede ver en la Figura 2.

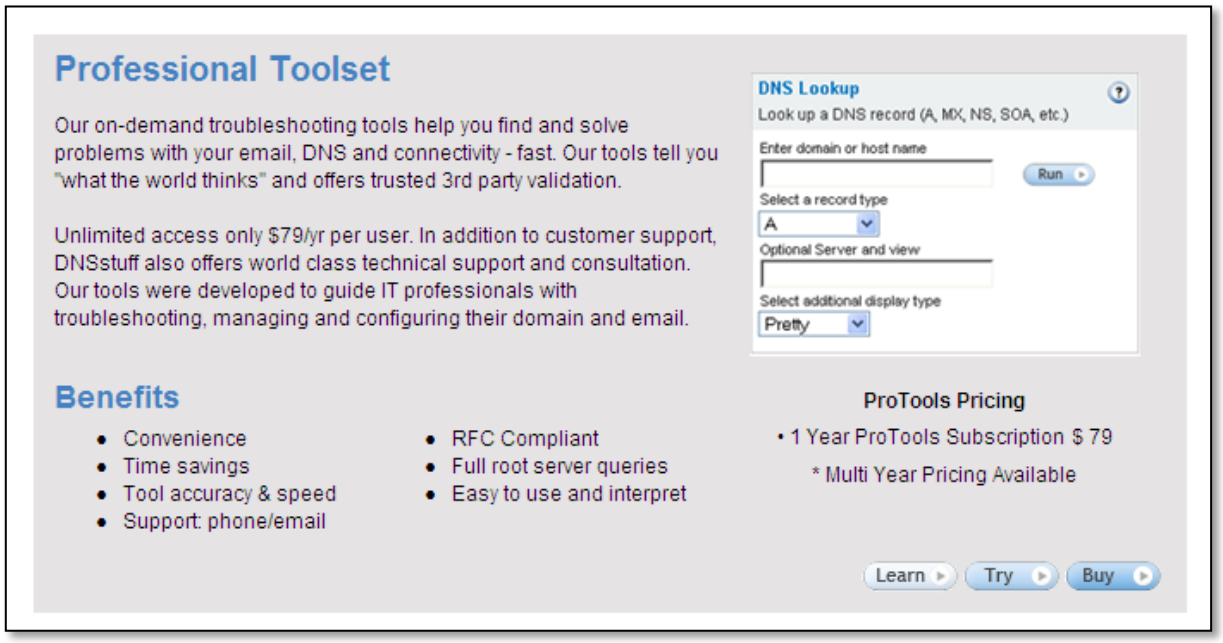



**Figura 2. DNSstuff.**

**DNSBajaj**

http://www.zonecut.net/dns/

Sin publicidad y con representación gráfica de la jerarquía de servidores consultados, pero únicamente devuelve registros de tipo NS y no permite elegir el servidor para hacer la consulta. Podemos ver el aspecto de esta web en la Figura 3.

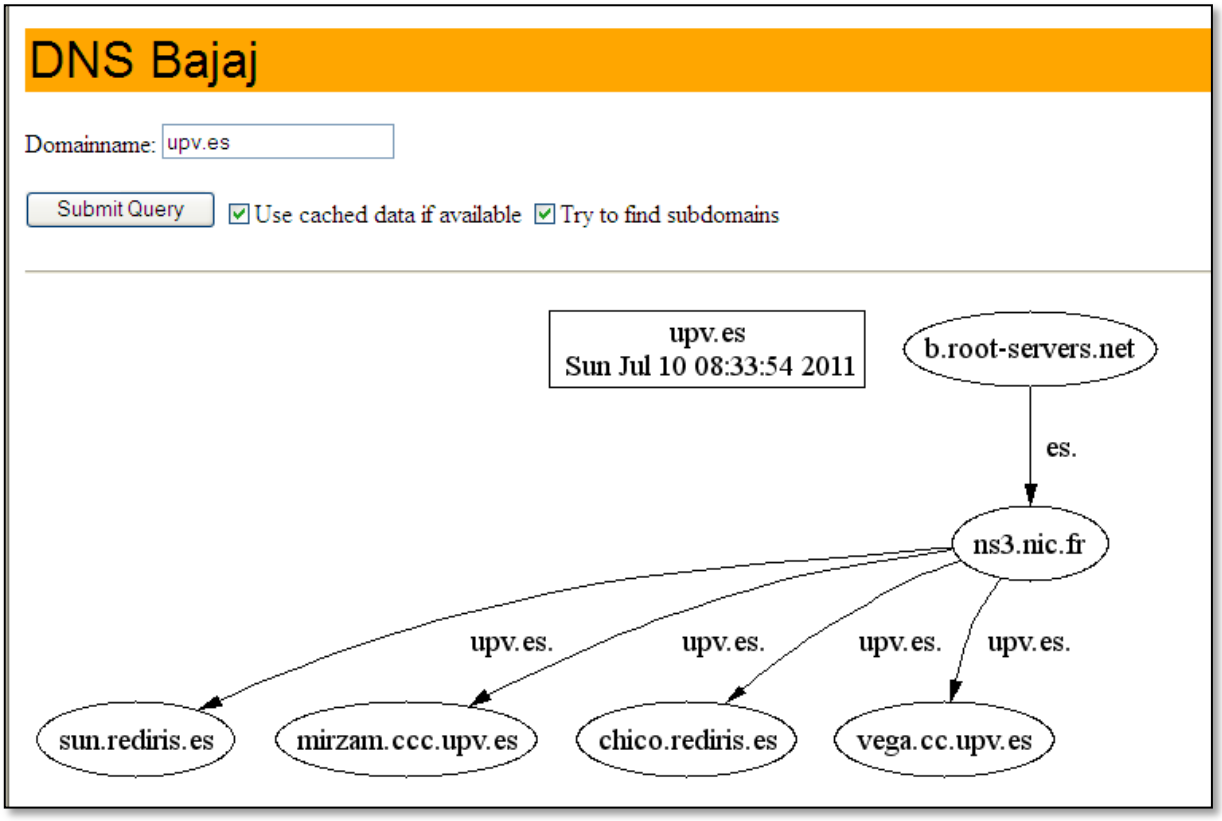



**Figura 3. DNSBajaj.**

## 2.3 Alternativas a *nslookup*

Para poder utilizar una herramienta que se adapte a nuestras necesidades, es posible realizar programas desarrollados en lenguajes de alto nivel que permiten realizar consultas DNS. Para ello, simplemente hay que utilizar las librerías adecuadas de lenguajes de programación. Es decir, existen librerías que permiten hacer consultas DNS desde lenguajes de programación de alto nivel. Un ejemplo serían **DNSJAVA** o **JNDI**, librerías utilizadas en el lenguaje Java.

**DNSJAVA** es una implementación de DNS en Java. Puede ser utilizado para las consultas, las transferencias de zona, y las actualizaciones dinámicas. Incluye una memoria caché que puede ser utilizada por los clientes, así como una implementación básica de un servidor DNS.

DNSJAVA es compatible con todos los tipos de registros comunes y proporciona además acceso de alto y bajo nivel a los servidores DNS. Las funciones de alto nivel realizan consultas de registros de un determinado nombre, tipo y clase, y devuelve un vector de registros. DNSJAVA permite utilizar memoria caché para reducir el número de consultas enviadas. Las funciones de bajo nivel permiten la manipulación directa de los mensajes y registros DNS, así como ajustar propiedades adicionales de la resolución. También incluye un clon de "*dig*"[2], un programa de actualización dinámica y un servidor DNS básico.

Por otro lado, **JNDI** (*Java Naming and Directory Interface*) es una Interfaz de Programación de Aplicaciones (*API)* que proporciona la funcionalidad de asignación de nombres y de directorio para aplicaciones escritas con el lenguaje de programación Java. Esto permite a los clientes descubrir y acceder a objetos y datos a través de su nombre y como el API está implementado en Java, es independiente de la implementación subyacente. Adicionalmente, especifica una interfaz de proveedor de servicio (SPI) que permite integrar implementaciones del servicio de directorio en el *framework*[3]**.** Las implementaciones pueden hacer uso de un servidor, un fichero, o una base de datos. Además, JNDI es independiente de cualquier implementación de servicios de directorio específicos, y por lo tanto, puede acceder a una gran variedad de directorios como LDAP, NDS, DNS y NIS.

A la hora de realizar aplicaciones web, se suelen utilizar *applets*. Un *applet* es un componente de una aplicación que se ejecuta en el contexto de otro programa, como por ejemplo un navegador web. Un *applet* puede ofrecer información gráfica, así como interactuar con el usuario. Sin embargo, y a diferencia de un programa, un *applet* no puede ejecutarse de manera independiente.

En aplicaciones web, un *applet* se incrusta en un documento HTML, es decir, en una

[2] Dig (*Domain Information Groper*) es una herramienta de línea de comandos disponible en prácticamente cualquier distribución linux y que permite hacer consultas a un servidor DNS. Dig forma parte de la suite de software del servidor de nombres de dominio BIND.

[3] Un framework hace referencia a una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de aplicaciones. En la práctica está compuesto por conjuntos de librerías para desarrollar aplicaciones (API's), además de librerías para su ejecución y herramientas para facilitar esta tarea (debuggers, GUI de desarrollo, etc).

página web. Cuando un navegador carga la página web que contiene el *applet*, éste se descarga en el navegador web y comienza a ejecutarse dentro del entorno del navegador. Este hecho permite crear programas que cualquier usuario puede ejecutar con tan solo descargar la página web en su navegador.

Entre las características de los *applets* de Java podemos mencionar que su esquema de seguridad no permite que tengan acceso a partes sensibles del sistema. La principal desventaja de este enfoque es que la entrega de permisos es engorrosa para el usuario común, lo cual juega en contra de uno de los objetivos de los Java *applets*: proporcionar una forma fácil de ejecutar aplicaciones desde el navegador web.

# 3 EL PROTOCOLO DNS

El sistema DNS utiliza una base de datos distribuida y jerárquica que almacena información asociada a nombres de dominio. Es compatible con el sistema de nombres de dominio como se describe en el RFC 1034 y RFC 1035, y actualizado y aclarado en el RFC 1123 y RFC 2181.

## 3.1 Componentes

El sistema DNS consta de tres componentes principales:

- **ZONAS DE AUTORIDAD**. Compuestas por el espacio de nombres de dominio y los registros de recursos para al menos un dominio. Conceptualmente, cada nodo y hoja del árbol del espacio de nombres de dominio hace referencia a un conjunto de información y permite, además, operaciones de consulta para extraer determinados tipos de información.
- **SERVIDORES DE NOMBRES** o servidores DNS. Programas que contestan a las consultas de los clientes. Mantienen información acerca de la estructura de uno o más arboles de dominio y el conjunto de información que los componen (sus zonas). Los servidores recursivos tienen la capacidad de reenviar la petición a otro servidor si no disponen de la dirección solicitada.
- ***RESOLVERS*** o Clientes DNS. Son programas que se ejecutan en el ordenador del usuario y que generan peticiones de resolución de nombres a los servidores DNS en respuesta a las consultas de las aplicaciones.

## 3.2 Nombres de dominio

Todo el espacio de Internet está organizado en dominios organizados jerárquicamente.

Un nombre de dominio está compuesto por dos o más cadenas de caracteres, llamadas etiquetas, separadas entre ellas por un punto (“.”).

La estructura de nombres de dominio se basa en una estructura jerárquica compuesta por un nodo raíz, representado por un punto (“.”), los dominios de nivel superior o TLD (del inglés *Top Level Domains*), los nombres de subdominio y, por último los nombres de host. Podemos ver un ejemplo de esta estructura en la Figura 4.

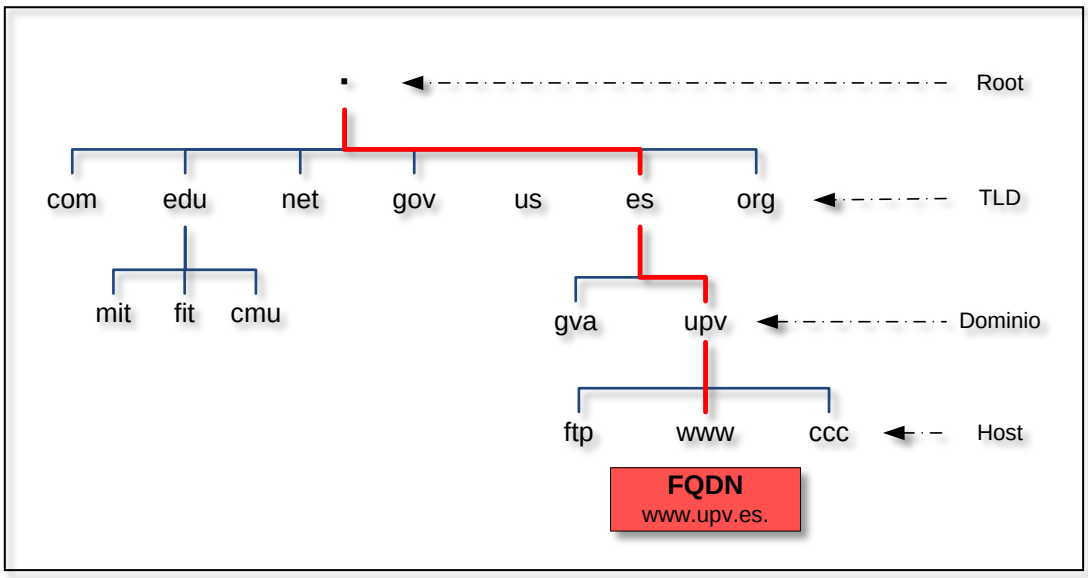



**Figura 4. Ejemplo de la estructura de los nombres de dominio del protocolo DNS.**

El nombre absoluto, que está compuesto por todas las etiquetas de nodo separadas por puntos e incluyendo el nodo raíz, se denomina la **dirección FQDN** (del inglés *Fully Qualified Domain Name*). La dirección FQDN permite identificar de manera unívoca un equipo en Internet.

La etiqueta del nodo raíz se suele omitir. Tras ella viene la etiqueta de un **dominio de nivel superior** o TLD (del inglés *top level domain*). Estos dominios de nivel superior son controlados por IANA en la base de datos de zona raíz (del inglés *Root Zone Database*). IANA (*Internet Assigned Numbers Authority*) es la autoridad responsable de la coordinación global de la zona raíz del DNS, las direcciones IP y otros recursos de protocolos de Internet.

Coloquialmente, un "**dominio**" normalmente hace referencia a la recopilación de las etiquetas de nodo con excepción de las de nodo raíz y de host.

Algunos dominios de nivel superior comunes son:

- COM - sitios Web comerciales, sin limitación.
- NET - sitios de la red Internet, sin limitación
- ORG - organizaciones sin fines de lucro, sin limitaciones
- EDU - limitado a las escuelas y organizaciones educativas
- MIL - restringido a los militares de EE.UU.
- GOV - restringido al gobierno de EE.UU.
- ES, UK, US y otros códigos de país de dos letras - cada uno se le asigna a la autoridad de nombres de dominio en el respectivo país.

Cada combinación de caracteres y punto a la izquierda del dominio de primer nivel indica un nivel en la estructura de dominio o **subdominio**. Cada nivel se refiere a un servidor o un grupo de servidores que manejan ese nivel de dominio.

La palabra más a la izquierda en el nombre de dominio, tal como *"www"* o *"mail"*, suele representar a la aplicación que se utiliza para acceder al servidor identificado con el nombre de dominio completo. En concreto, se especifica el nombre de una máquina en un dominio, por lo general dedicada a un propósito específico. Un dominio dado puede contener millones de nombres de host, siempre y cuando sean todos únicos en ese dominio.

Cada dominio o subdominio tiene una o más **zonas de autoridad**, las cuales publican la información acerca del dominio y los nombres de servicios de cualquier dominio incluido. La jerarquía de las zonas de autoridad coincide con la jerarquía de los dominios. Al inicio de esa jerarquía se encuentran los servidores raíz. Estos servidores son los que responden cuando se busca resolver un dominio de primer nivel.

Debido a que todos los nombres de un dominio dado deben ser únicos, tiene que haber alguna manera de controlar que no surgen duplicados. Ahí es donde aparece el papel de los registradores. Un registrador es una autoridad que puede asignar nombres de dominio directamente bajo uno o más dominios de nivel superior y registrarlos con InterNIC. InterNIC es un servicio de ICANN[4] que exige la unicidad de nombres de dominio a través de Internet. Cada registro de dominio se convierte en parte de una base de datos de registro de dominios central, conocido como la base de datos WHOIS.

[4] ICANN (*Internet Corporation for Assigned Names and Numbers*) es una organización que opera a nivel internacional y es la responsable 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.

### 3.3 Tipos de servidores DNS

**Primarios** (*Primary Name Servers*). Guardan los datos de un espacio de nombres en sus ficheros de zona. Son responsables de mantener la información actualizada y cualquier cambio debe ser notificado a este servidor.

**Secundarios** (*Secundary Name Servers*). Son aquellos que obtienen los datos de su zona desde otro servidor que tenga autoridad para esa zona. El proceso de copia de la información se denomina transferencia de zona.

**Locales** (*Caching-only servers*). No tienen autoridad sobre ningún dominio. Se limitan a contactar con otros servidores para resolver las peticiones de los clientes DNS. Estos servidores mantienen una memoria caché con las últimas preguntas contestadas. Cada vez que un cliente DNS le formula una pregunta, primero consulta en su memoria caché. Si encuentra la dirección IP solicitada, se la devuelve al cliente; si no, consulta a otros servidores, almacenando la respuesta en su memoria caché y comunicando la respuesta al cliente.

### 3.4 Tipos de consultas

Existen dos tipos de consultas que un cliente puede hacer a un servidor DNS:

- **Consulta recursiva.** Es aquella realizada a un servidor DNS, en la que el cliente DNS solicita al servidor DNS que proporcione una respuesta completa a la consulta. El servidor DNS comprueba la zona de búsqueda directa y la caché para encontrar una respuesta a la consulta.
- **Consulta iterativa.** Es aquélla efectuada a un servidor DNS en la que el cliente DNS solicita la mejor respuesta que el servidor DNS puede proporcionar, sin buscar ayuda adicional de otros servidores DNS. El resultado de una consulta iterativa suele ser una referencia a otro servidor DNS de nivel inferior en el árbol DNS.

El proceso de resolución normal de una consulta es el siguiente:

1. El servidor A recibe una consulta recursiva desde el cliente DNS.
2. El servidor A envía una consulta iterativa a B.
3. El servidor B refiere a A otro servidor de nombres, incluyendo a C.
4. El servidor A envía una consulta iterativa a C.
5. El servidor C refiere a A otro servidor de nombres, incluyendo a D.
6. El servidor A envía una consulta iterativa a D.
7. El servidor D responde.
8. El servidor A devuelve la respuesta al cliente DNS.
9. El cliente DNS entrega la resolución al programa que solicitó la información.

### 3.5 Tipos de registros DNS

En esta sección se van a describir los principales tipos de registros del protocolo DNS, que son:

- **A** = *Address* – (Dirección) Este registro se usa para traducir nombres de hosts a direcciones IPv4.
- **AAAA** = *Address* – (Dirección) Este registro se usa para traducir nombres de hosts a direcciones IPv6.
- **CNAME** = *Canonical Name* – (Nombre Canónico) Se usa para referirse a alias para los servidores de hosts de un dominio. Se utiliza cuando se están ejecutando 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. En cualquier caso, cada alias tiene su propia entrada DNS (por ejemplo, ftp.ejemplo.com y www.ejemplo.com).
- **NS** = *Name Server* – (Servidor de Nombres) Asocia un nombre de dominio a una lista de servidores de nombres para ese dominio.
- **MX** = *Mail Exchange* – (Registro de Intercambio de Correo) Asocia un nombre de dominio a una lista de servidores de correo para ese dominio.
- **PTR** = *Pointer* – (Indicador) También conocido como 'registro inverso'. Funciona a la inversa del registro A, traduciendo direcciones IP en nombres de dominio.
- **SOA** = *Start of authority* – (Autoridad de la zona) Asocia un nombre de dominio con el servidor DNS primario de su zona.
- **HINFO** = *Host INFOrmation* – (Información del sistema) Descripción del *host*. Permite que la gente conozca el tipo de máquina y sistema operativo al que corresponde un dominio.
- **TXT** = *TeXT* - (Información textual) Permite a los dominios identificarse de modos arbitrarios.
- **LOC** = LOCalización - Permite indicar las coordenadas del dominio.
- **SRV** = SeRVicios - Define la ubicación de los hosts para servicios específicos. Se define en el RFC 2782. Algunos protocolos como SIP o XMPP requieren un registro SRV asociado.
- **SPF** = *Sender Policy Framework* - Este registro especifica los *hosts* que están autorizados a enviar correo desde el dominio dado. Ayuda a combatir el *spam*.

### 3.6 Cómo funciona el protocolo DNS

Los usuarios no se comunican directamente con el servidor DNS. La resolución de nombres se realiza de forma transparente por las aplicaciones (por ejemplo, navegadores, clientes de correo y otras aplicaciones que usan Internet).

Cuando se visita un sitio web, como www.upv.es, el equipo sigue una serie de pasos para convertir el nombre del *host* en una dirección IP. Esto sucede cada vez que se utiliza un nombre de dominio, ya sea navegando por sitios web, enviando correo electrónico, o utilizando cualquier otro servicio por Internet.

Los pasos a seguir son los siguientes:

1.- Solicitud de información
El proceso comienza cuando el equipo local tiene que resolver un nombre de host. En primer lugar el *host* consulta su caché DNS local, que almacena la información que el equipo ha recibido recientemente. Si el equipo local no conoce la respuesta, necesitará realizar una consulta DNS para averiguarla.

2.- Consulta al servidor DNS local
Si la información no se encuentra en su cache DNS, el equipo local deberá consultar con el servidor DNS que tenga configurado localmente. Los servidores DNS tienen sus propias caches, por lo que el proceso por lo general termina aquí, y la información se devuelve al usuario.

3.- Consulta a los servidores de nombres raíz
Si el servidor local no tiene la respuesta, trasladan la consulta a uno de los servidores de nombres raíz. En internet, existen trece servidores de nombres raíz. El servidor de nombres raíz mirará la primera parte de la solicitud, leyendo de derecha a izquierda y dirigirá la consulta al servidor *Top-Level Domain* (TLD) para ese dominio. En el ejemplo que nos ocupa, para “www.upv.es” sería “.es”.

4.- Consulta a los servidores TLD
Cada TLD tiene su propio conjunto de servidores de nombres, que actúan como una recepcionista para cada TLD. El servidor de nombres de dominio de nivel superior revisa la siguiente parte de la petición y dirigirá la consulta a los servidores de nombres responsables de este dominio específico, en nuestro ejemplo “upv.es”.

5.- Consulta a los servidores DNS autorizados
Estos servidores de nombres autorizados son responsables de conocer toda la información sobre un dominio específico. Esta información se almacena en los registros de DNS. Hay muchos tipos de registros, que contienen cada una un tipo diferente de información. En este ejemplo, queremos obtener la dirección IP de www.upv.es, por lo que pedimos al servidor de nombres autorizado el registro de dirección (registro de tipo A).

6.- Obtención del registro
El servidor local obtiene el registro A para el nombre de dominio “www.upv.es” del servidor autorizado y almacena dicho registro en su cache local. Si alguien más solicita este registro, el servidor local ya tiene la respuesta y no necesitará repetir la búsqueda. Todos los registros tienen un valor de “tiempo de vida” tras el cual el servidor necesitará solicitar una nueva copia del registro para estar seguro que su información está actualizada.

7.- Recepción de la respuesta
Una vez obtenida la respuesta, el servidor local devuelve el registro A al equipo local que ha hecho la consulta. Éste almacena la respuesta en su cache local, lee la dirección IP del registro y pasa la información al navegador. El navegador entonces abre una conexión con el servidor web y recibe la página web.

La Figura 5 muestra la representación gráfica de la secuencia de las consultas y respuestas en un caso de ejemplo de una consulta a www.ccc.ejemplo.es:

1. El equipo cliente hace una consulta recursiva pidiendo la dirección (registro A) del host “www.ccc.ejemplo.es” a su servidor DNS local.
2. El servidor DNS local hace una consulta iterativa a uno de los servidores raíz.
3. El servidor raíz responde al DNS local con las direcciones de los servidores TLD para el dominio “es”.
4. El servidor local hace una consulta iterativa a uno de los servidores TLD.
5. El servidor TLD responde con la dirección del servidor autorizado para el dominio “ejemplo.es”.
6. El servidor local hace una consulta recursiva al servidor autorizado.
7. El servidor autorizado para el dominio “ejemplo.es” hace una consulta recursiva al servidor autorizado para el dominio “ccc.ejemplo.es”.
8. El servidor autorizado para el dominio “ccc.ejemplo.es” responde con la dirección del host “www.ccc.ejemplo.es”
9. El servidor autorizado para el dominio “ejemplo.es” reenvía la dirección del host “www.ccc.ejemplo.es” al servidor local DNS que inicio la consulta.
10. El servidor local devuelve la respuesta al equipo del usuario.
11. El equipo del usuario puede ahora acceder al equipo “www.ccc.ejemplo.es”.

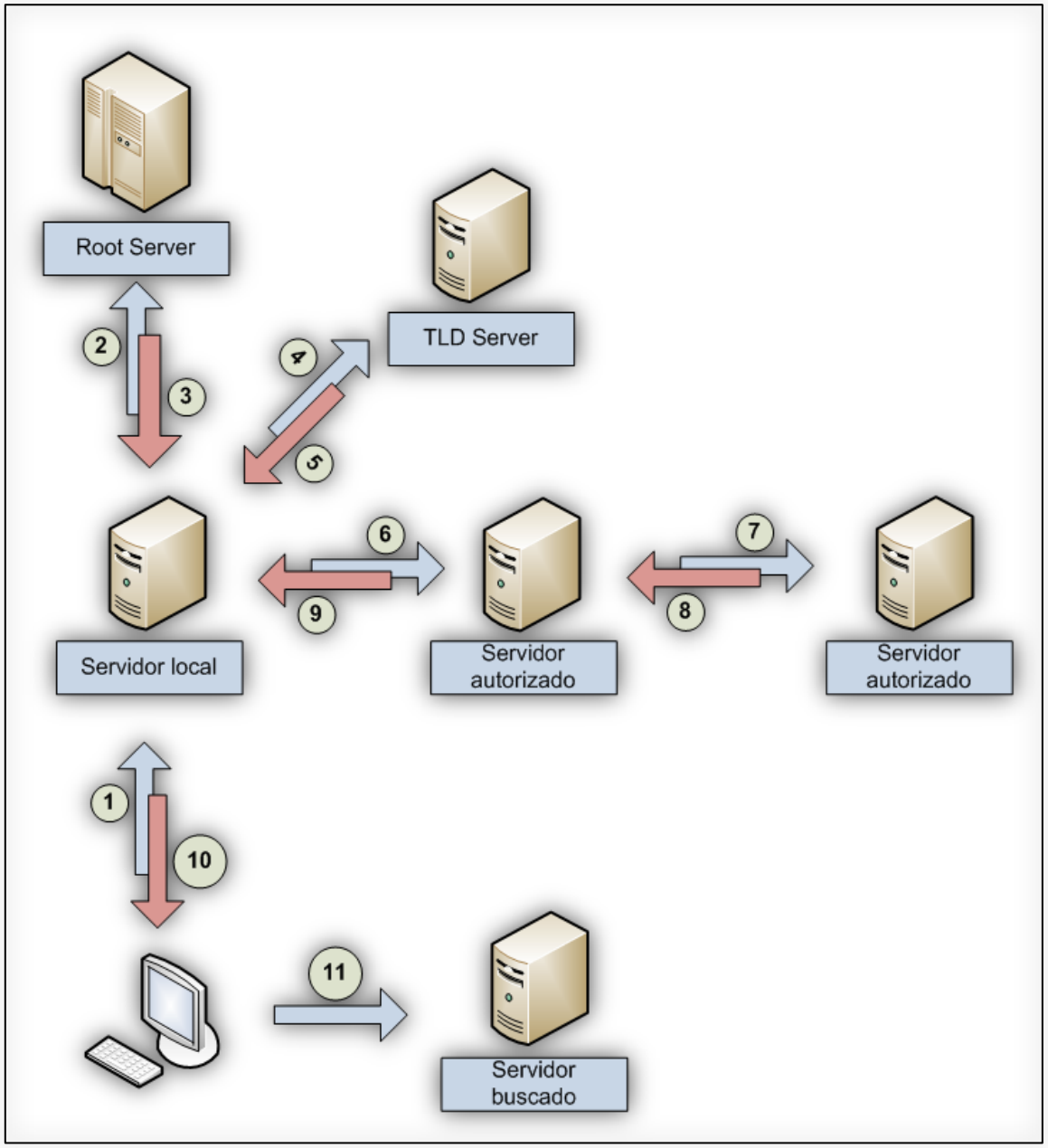



**Figura 5. Secuencia de consultas DNS.**

# 4 SOLUCIÓN PROPUESTA

En este apartado se describe el desarrollo e implementación de la herramienta web para el análisis del tráfico DSN.

La solución propuesta está basada en el desarrollo de un *applet* implementado en Java. Este *applet* permite realizar de forma gráfica, y utilizando como soporte un navegador web, consultas DNS. La herramienta implementada en este proyecto clasifica las respuestas obtenidas en función del tipo de registro DNS devuelto, indicando asimismo el servidor que ha respondido.

De esta forma, esta herramienta puede ser utilizada por usuarios que no tengan un conocimiento específico del protocolo DNS o la herramienta *nslookup*.

## 4.1 Selección del entorno de desarrollo

Para la selección del entorno de desarrollo, se han analizado dos entornos gráficos ampliamente extendidos y conocidos para el desarrollo en lenguaje Java, como son **Eclipse** y **NetBeans**.

**Eclipse** es un es un proyecto *open source* de la fundación Eclipse para el desarrollo de un entorno de desarrollo integrado de código abierto multiplataforma. Eclipse es una comunidad *open source* cuyos proyectos se centran en la construcción de una plataforma de desarrollo abierta compuesta por marcos extensibles y herramientas para la construcción, despliegue y gestión de software a través de su ciclo de vida. Esta plataforma ha sido usada típicamente para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT).

El Proyecto Eclipse fue creado originalmente por IBM en noviembre de 2001 y apoyado por un consorcio de proveedores de software. La Eclipse Foundation se creó en enero de 2004 como una organización independiente sin fines de lucro, para actuar como administrador de la comunidad Eclipse.

**NetBeans** IDE es un entorno de desarrollo Java, pero que puede servir para cualquier otro lenguaje de programación. Existe un importante número de módulos para extender las funcionalidades del IDE. NetBeans es un producto libre y gratuito sin restricciones de uso.

Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

Sun MicroSystems fundó el proyecto NetBeans en junio del 2000. NetBeans es código abierto y gratuito para uso tanto comercial como no comercial. El código fuente está disponible para su reutilización de acuerdo con la *Common Development and Distribution License* (CDDL) v1.0 y la *GNU General Public License* (GPL) v2.

En resumen, en ambos casos se trata de IDEs *open source* con características muy similares. Aunque Eclipse es un entorno más potente y extendido, no dispone de una herramienta nativa para el diseño gráfico de un *applet*. Las herramientas existentes son de terceras partes y deben de ser instaladas y configuradas tras la instalación del propio entorno. En el caso de NetBeans sí que se dispone de una herramienta nativa para el diseño del interfaz gráfico del *applet* perfectamente integrada con el resto del entorno, por lo que ésta ha sido la herramienta seleccionada.

## 4.2 Selección de librerías Java

Se han analizado dos de las librerías disponibles para la realización de consultas DNS, **JNDI** y **DNSJAVA**. En un punto anterior de este documento ya se han presentado las características de ambas librerías.

Como resumen, podemos decir que JNDI es un API en tecnología Java que proporciona la funcionalidad de asignación de nombres y directorio a aplicaciones escritas en lenguaje de programación Java. Está diseñado especialmente utilizando el modelo de objetos de Java. Usando JNDI, las aplicaciones Java puede almacenar y recuperar objetos Java de cualquier tipo. Además, JNDI proporciona métodos para realizar operaciones estándar de directorio, como la asociación de atributos con los objetos y la búsqueda de objetos usando sus atributos.

JNDI también es independiente de cualquier implementación específica de servicios de directorio o de nombres. Esto permite a las aplicaciones el acceso a diferentes servicios de nombres y directorio con una API común. Diferentes proveedores de servicio de nombres y directorio puede conectarse sin problemas a través de este API común. Esto permite a las aplicaciones basadas en tecnología Java beneficiarse de la variedad de servicios de nombres y directorio existentes, como LDAP, NDS, DNS y NIS, además de permitir a las aplicaciones coexistir con sistemas y software heredados.

Por su parte, DNSJAVA es una implementación de DNS en Java. Soporta todos los tipos definidos por el registro (incluyendo los tipos de DNSSEC), y tipos desconocidos. Puede ser utilizado para las consultas, las transferencias de zona, y las actualizaciones dinámicas. Se incluye una memoria caché que puede ser utilizado por los clientes, y una mínima implementación de un servidor. Es compatible con TSIG mensajes autenticados, comprobación parcial DNSSEC y EDNS0.

DNSJAVA proporciona una funcionalidad más allá de la de la clase *InetAddress*, ya que está escrito en Java puro. Es completamente "*threadable*"[5], y en muchos casos es más rápido que *InetAddress*.

DNSJAVA proporciona acceso de nivel alto y bajo de DNS. Las funciones de alto nivel permiten realizar consultas de registros de un determinado nombre, tipo y clase, y devolver la respuesta o la razón para el fracaso. También hay funciones similares a las de

[5] El termino *threadable* hace referencia a ser adecuado para ejecutarse en modo *multithread*. Un sistema *multithread* es capaz de ejecutar varias secuencias de ejecución (*threads*) simultáneamente.

la clase *InetAddress*. La memoria caché se utiliza para reducir el número de consultas DNS enviadas. Las funciones de bajo nivel permiten la manipulación directa de los mensajes DNS y los registros. También es posible ajustar las propiedades adicionales de resolución.

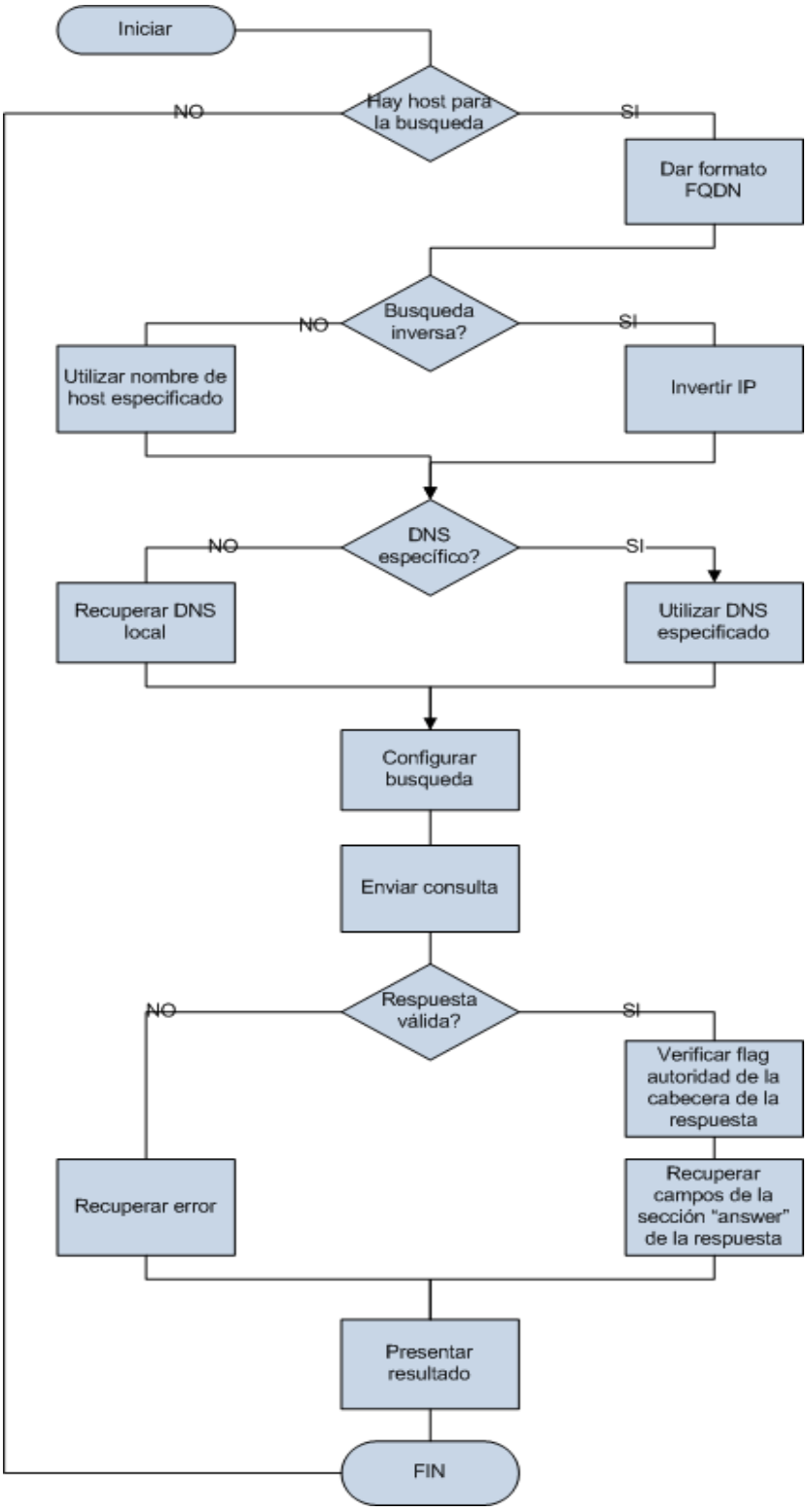
A la hora de comparar estas dos librerías con el fin de seleccionar una de ellas para el presente proyecto, podemos tener en cuenta diferentes características:

- Facilidad de implementación: JNDI proporciona funciones de más alto nivel que facilitan la implementación.
- Tiempo de ejecución: DNSJAVA realiza las consultas de forma más rápida que JNDI.
- Acceso a bajo nivel a los mensajes DNS: DNSJAVA dispone de funciones de bajo nivel que permiten la manipulación directa de los mensajes DNS.
- Especificidad: JNDI es un API para servicios de nombre y directorio en general, no únicamente para servicios DNS. DNSJAVA es una implementación específica para DNS.

Por todos estos factores y a pesar de una mayor dificultad de implementación, DNSJAVA ha sido la seleccionada para el desarrollo de la herramienta web de análisis de tráfico DNS que se presenta en este proyecto.

## 4.3 Flujo de ejecución

El siguiente diagrama presenta el flujo de ejecución del *applet*.



El funcionamiento es el siguiente. Cuando el usuario solicita una consulta, se comprueba que éste haya proporcionado un nombre de dominio o host para la consulta. Para poder utilizarlo, el nombre de dominio debe tener el formato FQDN. La diferencia entre el formato habitual y el FQDN es el punto (“.”) de terminación de la cadena que representa el dominio raíz. Por lo tanto, se añade dicho punto al final de la cadena.

En el siguiente paso se comprueba el tipo de registro DNS solicitado. Si el registro es de tipo “PTR” significa que se quiere realizar una búsqueda inversa. Cuando un cliente DNS desea conocer el nombre de dominio asociado a la dirección IP w.x.y.z realiza una pregunta inversa a z.y.x.w.in-addr.arpa. La inversión de los bytes es necesaria debido a que los nombres de dominio son más genéricos por la derecha, al contrario que ocurre con las direcciones IP. Para evitar una búsqueda exhaustiva por todo el espacio de nombres de dominio, se utiliza un dominio especial llamado *in-addr.arpa*. Por lo tanto, para realizar una búsqueda inversa procedemos a comprobar que la información introducida para la búsqueda es una dirección IP válida y a invertirla añadiéndole el sufijo ”in-addr.arpa.”, que, como se ha explicado, es el formato necesario para poder realizar búsquedas inversas.

Después, comprobamos si el usuario ha introducido un valor de un servidor DNS específico para hacer la consulta. El servidor DNS se puede especificar tanto como nombre de dominio como dirección IP. Si el usuario no ha introducido ningún servidor DNS para la consulta, entonces utilizaremos el servidor DNS por defecto, para lo que recuperamos la dirección del servidor DNS que está configurado localmente en el equipo.

Posteriormente, se configura la búsqueda con los valores recuperados de dominio, tipo de registro y servidor DNS a consultar. Tras lanzar la consulta verificamos si la respuesta contiene algún código de error que recuperaremos. En caso contrario la consulta se ha respondido correctamente.

A continuación, procedemos a recuperar la cabecera de la respuesta DNS y a analizarla para comprobar si está o no marcado el flag “AA” para determinar si la respuesta proviene o no de un servidor autorizado.

Por último accedemos a la sección “*answer*” de la respuesta y recorremos cada campo que contenga, recuperando en este proceso el tipo de registro y su valor para presentarlos por pantalla.

En el caso de que no se hubiera introducido un dominio o host para la búsqueda o una IP válida para la búsqueda inversa o el resultado de la misma devuelve un error, se presentará por pantalla el mensaje de error correspondiente.

### 4.4 Descripción del *applet*

Se ha definido el siguiente interface gráfico para la realización de las consultas:

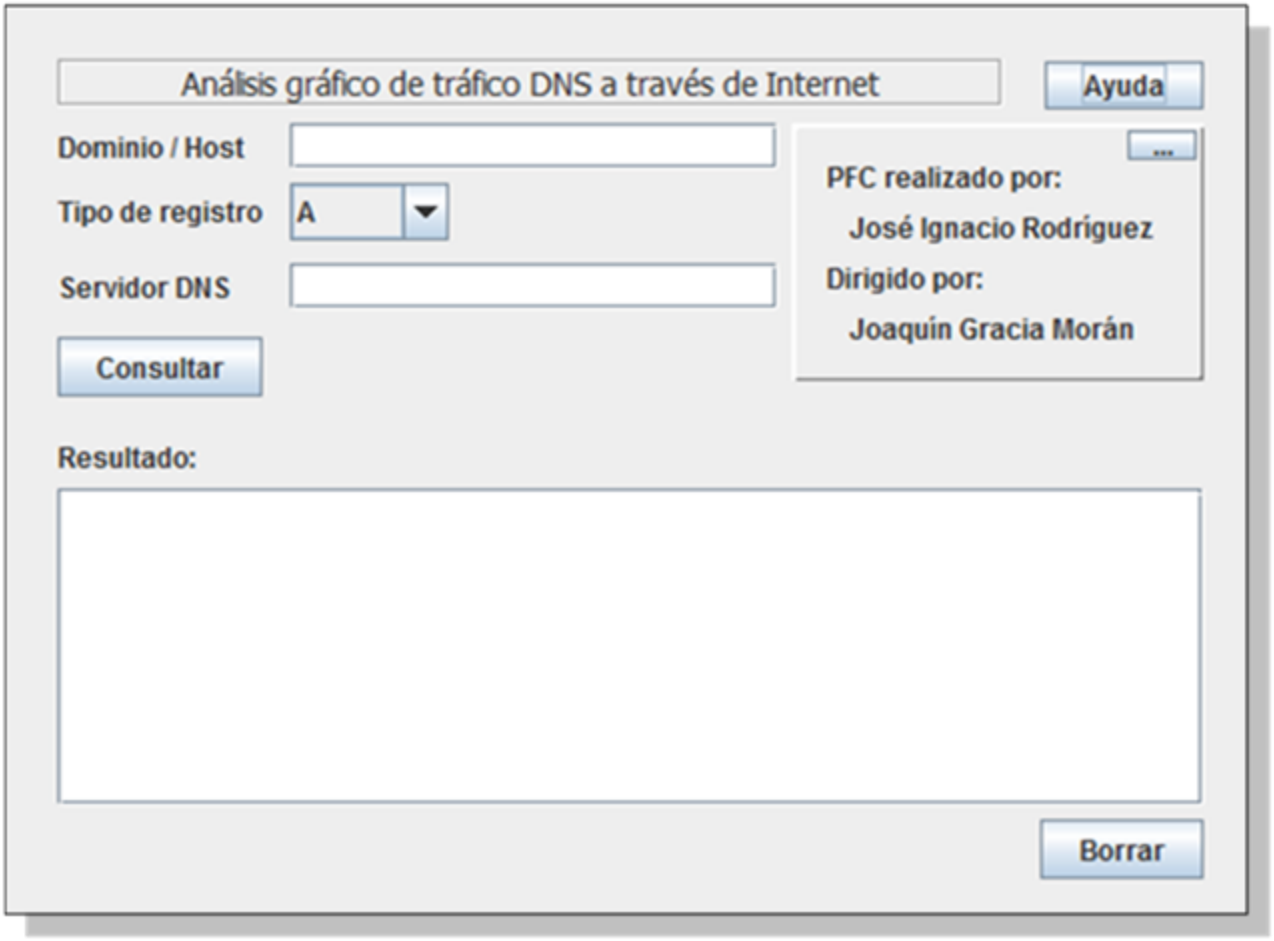



**Figura 6. Interface gráfico del *applet***

Los diferentes campos se utilizan de la siguiente forma:

**Dominio / Host**: En este campo se introduce el nombre del dominio o de host a consultar, en formato FQDN. Como se ha comentado previamente, FQND es el acrónimo de *fully qualified domain name*. Un nombre FQDN está compuesto por dos partes, un nombre de host y un nombre absoluto de dominio que termina en un TLD (*Top Level Domain*). Por ejemplo www.upv.es o vega.cc.upv.es.

**Tipo de registro**: Se utiliza para seleccionar el tipo de registro a consultar. En esta herramienta, se pueden consultar los siguientes tipos de registros:

- A (*Address*). Este registro proporciona la correspondencia entre un nombre de host y la/s dirección/es IP correspondiente/s.

- CNAME (*Canonical Name*). Este registro proporciona la correspondencia entre un alias y el nombre canónico del host.
- NS (*Name Server*). Con esta consulta obtenemos el nombre del servidor autorizado para resolver la consulta sobre la dirección IP.
- MX (*Mail Exchange*). Asocia un nombre de dominio a una lista de servidores de intercambio de correo para ese dominio.
- PTR (*Pointer*). Conocido como 'registro inverso', funciona a la inversa del registro A, es decir, traduce direcciones IP en nombres de dominio.
- SOA (*Start Of Authority*). Proporciona información sobre el servidor DNS primario de la zona.
- HINFO (*Host INFOrmation*). Descripción del host. Con este registro podemos obtener información del host, como por ejemplo el procesador o el Sistema Operativo de la máquina.
- TXT (*TeXT*). Permite a los dominios identificarse de modos arbitrarios. Este registro es un texto en formato libre que se puede usar de forma arbitraria.
- ANY. Esta opción proporciona información sobre todos los tipos de registro disponibles para el dominio.

**Servidor DNS**: Indica el servidor DNS al que se quiere hacer la consulta. Si se deja en blanco el *applet* utiliza el servidor DNS configurado por defecto en el sistema.

**Consultar**: Realiza la consulta con los parámetros introducidos los campos anteriormente descritos.

**Resultado**: Muestra el resultado de la consulta.

**Borrar**: Permite borrar el área de presentación de resultados.

## 4.5 Puesta en marcha del *applet*

### 4.5.1 Herramientas necesarias

Para poder ejecutar el *applet* necesitaremos tres herramientas:

- Un navegador web,
- Una página web que haga la llamada al *applet*
- Tener instalado el entorno de ejecución Java (JRE: *Java Runtime Environment*).


**Figura 7. Inicio de la ejecución del *applet*.**

Al abrir la página web[6] con el navegador, éste encontrará la información que le permitirá acceder a la ubicación del *applet*, descargar el *bytecode*[7] Java que lo compone y ejecutarlo en la máquina virtual Java.

En la carpeta de proyecto "\Ficheros anexos\applet\" se encuentra tanto el *applet* como un ejemplo de página web que lo llama para permitir su ejecución.

[6] En nuestro caso denominada "Analisis_trafico_DNS.html"
[7] *Bytecode* es el formato compilado para los programas de Java. Una vez que un programa Java se ha convertido en *bytecode*, puede ser ejecutado por la Java Virtual Machine (JVM). Los archivos *bytecode* en general tienen la extensión ".class".

El entorno de ejecución de Java se puede descargar de forma gratuita desde la URL: www.java.com/es/download/

Cuando se abre la página web nos pedirá confirmación para la ejecución de la aplicación, ya que en este caso el certificado con el que se ha firmado el *applet* no es conocido por el navegador y por tanto de confianza. En la Figura 7 se visualiza este proceso.

Una vez confirmada la ejecución se cargará en el navegador el interface gráfico del *applet* y permitirá introducir los datos de la búsqueda a realizar, siendo el aspecto de la página web el que se muestra a continuación en la Figura 8:

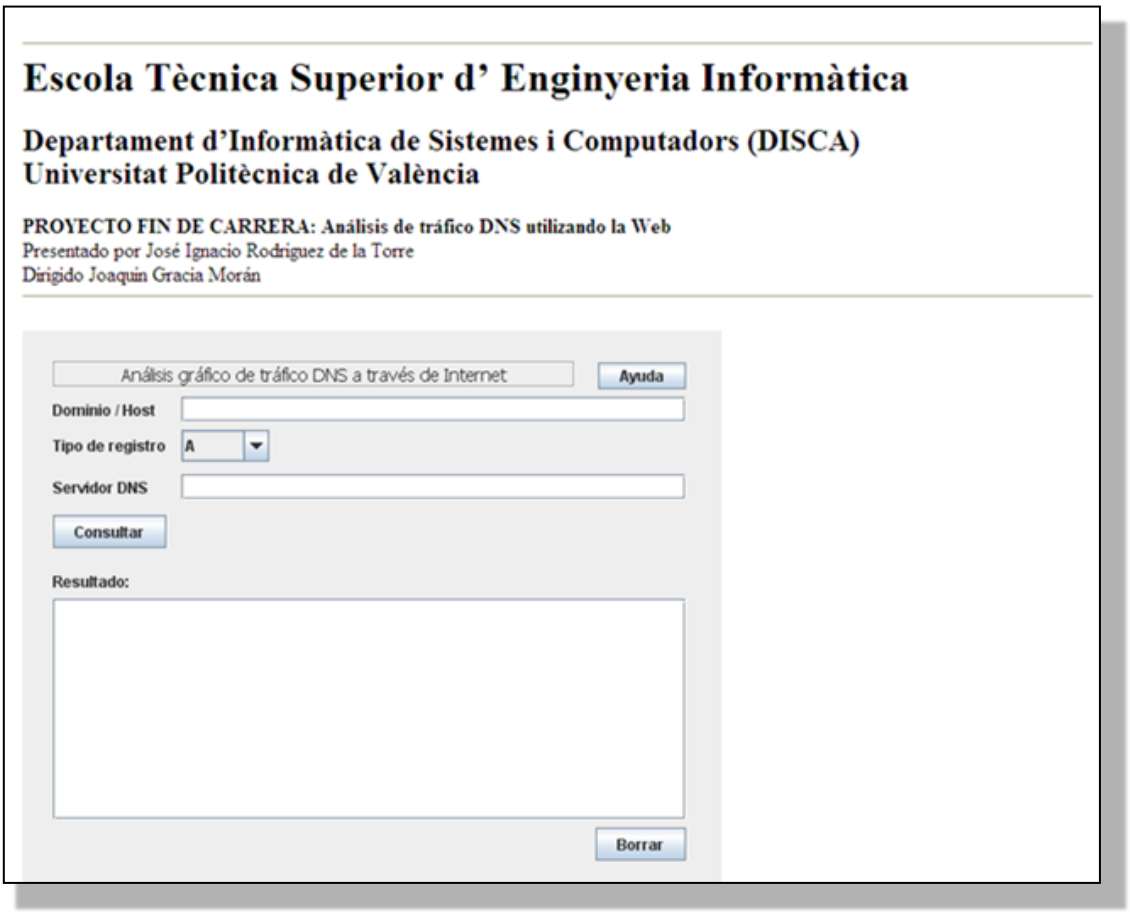



**Figura 8. Aspecto del *applet* cargado en la página web.**

Tras lanzar la búsqueda, el *applet* presentará el resultado de la misma indicando los tipos de registro recuperados y el servidor DNS que los ha proporcionado. A continuación podemos ver en la Figura 9 un ejemplo del aspecto del *applet* después de realizar una consulta.

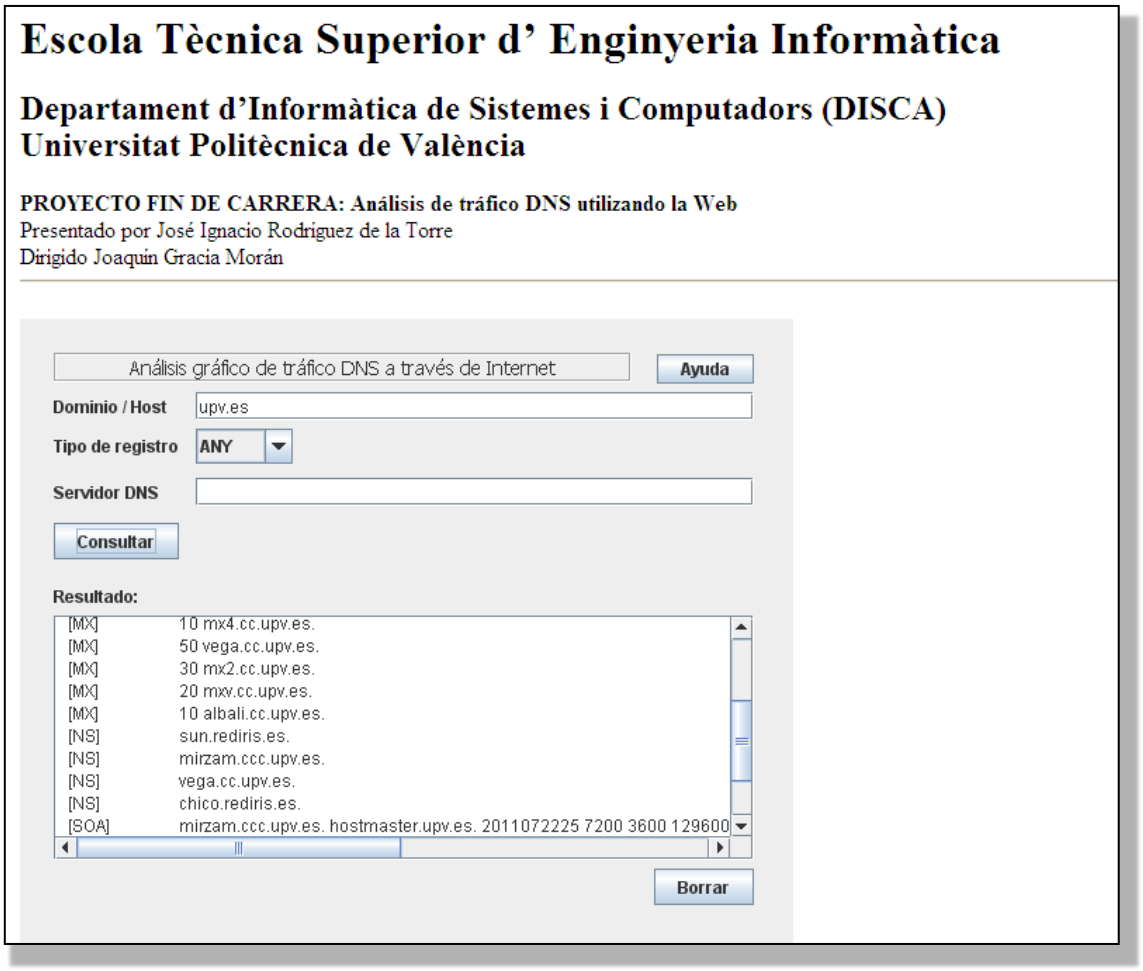



**Figura 9. Presentación de resultados en el *applet*.**

### 4.5.2 Problemas con la firma

Uno de los principales problemas al intentar ejecutar en el navegador un *applet* que accede a los recursos locales es que no se podrá ejecutar si no está firmado. En la consola de la JVM[8] aparecerá un mensaje de error parecido al siguiente:

java.security.AccessControlException: access denied

Para firmar digitalmente un *applet* es necesario un certificado, es decir, un archivo que contenga la firma. El mismo jdk nos provee de las herramientas necesarias para crearlo y posteriormente aplicarlo al archivo .jar.

Dentro de la carpeta *bin* que posee el jdk (en Windows se encuentra normalmente en C:\Archivos de programa\java\jdkx.xx.xx) existen un par de aplicaciones que nos servirán. Dichas aplicaciones son las siguientes:

- keytool: para generar los certificados
- jarsigner: para firmar los archivos .jar

Para generar un certificado que posteriormente servirá para firmar los *applets* basta ejecutar el siguiente comando desde la línea de comandos:

keytool –genkey –alias <Alias> –validity <Días> –v

Donde:

- <Alias> es el alias a utilizar.
- <Días> son los días de validez del certificado.

Una vez se tiene el certificado hay que ejecutar el siguiente comando para firmar el *applet*:

jarsigner.exe <AppletFirmado.jar> <Alias> –verbose

Con estas acciones, el *applet* queda firmado y listo para ejecutarse en el navegador.

[8] JVM: *Java Virtual Machine*

# 5 RESUMEN Y PALABRAS CLAVE

## 5.1 Resumen

En este proyecto se ha diseñado e implementado un *applet* que permite, de forma gráfica, realizar consultas DNS y obtener las respuestas correspondientes.

Las respuestas obtenidas se clasifican en función del tipo de registro DNS devuelto. Además, el *applet* indica el tipo de servidor que ha respondido.

Para la implementación de este *applet* se ha utilizado el IDE de desarrollo NetBeans y la librería DNSJAVA.

Para permitir la correcta ejecución del *applet* ha sido necesario firmarlo digitalmente, lo que se ha realizado con las herramientas del JDK de java *keytool*, la cual genera los certificados y *jarsigner*, que se utiliza para firmar los archivos .jar

## 5.2 Palabras clave

DNS, applet, Java, JNDI, nslookup.

# 6 BIBLIOGRAFÍA Y REFERENCIAS

**JNDI**
The JNDI Tutorial: http://download.oracle.com/javase/jndi/tutorial/

**Firma de Applets**
http://es.debugmodeon.com/articulo/como-y-por-que-firmar-los-applets-java

**DNS**
RFC 1034 - Domain names - concepts and facilities

RFC 1035 - Domain names - implementation and specification

RFC 1123 - Requirements for Internet Hosts -- Application and Support

RFC 2181 - Clarifications to the DNS Specification

RFC 2782 - A DNS RR for specifying the location of services (DNS SRV)

"Redes de Computadores. Un enfoque descendente", James F. Kurose & Keith W. Ross. 5ª Ed. 2010. Prentice-Hall.

http://es.wikipedia.org/wiki/Domain_Name_System#Tipos_de_registros_DNS

**Nslookup**
http://support.microsoft.com/kb/200525/es

http://en.wikipedia.org/wiki/Nslookup

**Applet java**
http://es.wikipedia.org/wiki/Applet_Java

**Función getRevName**
http://blogs.captechconsulting.com/blog/david-tiller/accessing-the-dusty-corners-dns-java

**DNSJAVA**
http://www.xbill.org/dnsjava/

**NetBeans**
http://netbeans.org/

**Eclipse**
http://www.eclipse.org/

# 7 ANEXO A: CÓDIGO FUENTE

A continuación se presenta el código fuente desarrollado comentado para su mejor entendimiento.

```
/*
 * Analisis_trafico_DNS.java
 * Created on 12-dic-2010, 13:00:47
 */

package Analisis_trafico_DNS;

/*
 * @author José Ignacio Rodríguez de la Torre
*/

import javax.naming.*;
import javax.naming.directory.*;
import java.util.*;
import org.xbill.DNS.*;

public class Analisis_trafico_DNS extends javax.swing.JApplet {
    /** Initializes the applet Analisis_trafico_DNS */
    //Esta función inicializa el applet. Se ejecuta automáticamente cuando se llama al applet
    public void init() {
        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
                    initComponents();
                    jPanelAyuda.setVisible(false);
                }
            });
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    /** This method is called from within the init() method to
```

```
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
//Esta función inicializa los componentes gráficos del applet
//Se invoca desde el procedimiento init() en el momento de inicializar el applet
private void initComponents() {
    jPanel1 = new javax.swing.JPanel();
    jLabel1 = new javax.swing.JLabel();
    lookupButton = new javax.swing.JButton();
    jComboRegistro = new javax.swing.JComboBox();
    dominioTextField = new javax.swing.JTextField();
    jLabel2 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    dnsTextField = new javax.swing.JTextField();
    ClearButton = new javax.swing.JButton();
    jLabel5 = new javax.swing.JLabel();
    jLabel6 = new javax.swing.JLabel();
    jButtonAyuda = new javax.swing.JButton();
    jPanelAyuda = new javax.swing.JPanel();
    jLabel7 = new javax.swing.JLabel();
    jLabel8 = new javax.swing.JLabel();
    jButtonSalirAyuda = new javax.swing.JButton();
    jLabel9 = new javax.swing.JLabel();
    jLabel10 = new javax.swing.JLabel();
    jScrollPane1 = new javax.swing.JScrollPane();
    jTextArea1 = new javax.swing.JTextArea();
    jLabel1.setText("Dominio / Host");
    lookupButton.setText("Consultar");
    lookupButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            lookupButtonActionPerformed(evt);
        }
```

```
        });
        jComboRegistro.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "A", "NS", "CNAME",
"SOA", "PTR", "MX", "TXT", "HINFO", "ANY" }));
        jLabel2.setText("Tipo de registro");
        jLabel3.setText("Servidor DNS");
        dnsTextField.setToolTipText("Servidor DNS al que se va a realizar la consulta.");
        dnsTextField.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                dnsTextFieldActionPerformed(evt);
            }
        });
        ClearButton.setText("Borrar");
        ClearButton.setAlignmentY(0.0F);
        ClearButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                ClearButtonActionPerformed(evt);
            }
        });
        jLabel5.setText("Resultado:");
        jLabel6.setFont(new java.awt.Font("Tahoma", 0, 14));
        jLabel6.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel6.setText("Análisis gráfico de tráfico DNS a través de Internet");
        jLabel6.setBorder(javax.swing.BorderFactory.createEtchedBorder());
        jButtonAyuda.setText("Ayuda");
        jButtonAyuda.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonAyudaActionPerformed(evt);
            }
        });
        jPanelAyuda.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
        jLabel7.setText("PFC realizado por:");
        jLabel8.setText("Dirigido por:");
        jButtonSalirAyuda.setText("X");
        jButtonSalirAyuda.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonSalirAyudaActionPerformed(evt);
```

```
        }
    });
    jLabel9.setText("José Ignacio Rodríguez");
    jLabel10.setText("Joaquín Gracia Morán");
    javax.swing.GroupLayout jPanelAyudaLayout = new javax.swing.GroupLayout(jPanelAyuda);
    jPanelAyuda.setLayout(jPanelAyudaLayout);
    jPanelAyudaLayout.setHorizontalGroup(
        jPanelAyudaLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanelAyudaLayout.createSequentialGroup()
            .addContainerGap()

.addGroup(jPanelAyudaLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanelAyudaLayout.createSequentialGroup()
                    .addGap(10, 10, 10)
                    .addComponent(jLabel10)
                    .addContainerGap())

.addGroup(jPanelAyudaLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanelAyudaLayout.createSequentialGroup()
                        .addGap(10, 10, 10)
                        .addComponent(jLabel9)
                        .addContainerGap())
                    .addGroup(jPanelAyudaLayout.createSequentialGroup()
                        .addComponent(jLabel7)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 29,
Short.MAX_VALUE)
                        .addComponent(jButtonSalirAyuda, javax.swing.GroupLayout.PREFERRED_SIZE, 30,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(jPanelAyudaLayout.createSequentialGroup()
                        .addComponent(jLabel8)
                        .addContainerGap(89, Short.MAX_VALUE)))))
    );
    jPanelAyudaLayout.setVerticalGroup(
        jPanelAyudaLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanelAyudaLayout.createSequentialGroup()

.addGroup(jPanelAyudaLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jButtonSalirAyuda, javax.swing.GroupLayout.PREFERRED_SIZE, 13,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGroup(jPanelAyudaLayout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(jLabel7)))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jLabel9)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(jLabel8)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jLabel10)
            .addContainerGap())
        );
        jTextArea1.setColumns(20);
        jTextArea1.setRows(5);
        jScrollPane1.setViewportView(jTextArea1);
        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(ClearButton)
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE, 411,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(jButtonAyuda))
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                            .addComponent(jLabel5, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(lookupButton, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(jLabel3, javax.swing.GroupLayout.Alignment.LEADING,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jLabel2, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 84, Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jComboRegistro, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(dnsTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 227,
Short.MAX_VALUE)
                    .addComponent(dominioTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 227,
Short.MAX_VALUE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jPanelAyuda, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 486, Short.MAX_VALUE))
        .addGap(384, 384, 384)
        .addComponent(jLabel4)
        .addContainerGap())
    );
    jPanel1Layout.setVerticalGroup(
      jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(jPanel1Layout.createSequentialGroup()
        .addContainerGap()
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
          .addComponent(jLabel6)
          .addComponent(jButtonAyuda, javax.swing.GroupLayout.PREFERRED_SIZE, 21,
javax.swing.GroupLayout.PREFERRED_SIZE))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
          .addComponent(jPanelAyuda, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
          .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addGroup(jPanel1Layout.createSequentialGroup()
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(jLabel1)
                        .addComponent(dominioTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(jLabel2)
                        .addComponent(jComboRegistro, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(10, 10, 10)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(jLabel3)
                        .addComponent(dnsTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addComponent(jLabel4))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(lookupButton)
                .addGap(18, 18, 18)
                .addComponent(jLabel5)))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 146,
Short.MAX_VALUE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(ClearButton)
            .addGap(21, 21, 21))
        );
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 517,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(65, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
```

```
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
        );
    }// </editor-fold>

    //Este procedimiento oculta el panel de ayuda. Se invoca al pulsar el botón de cierre del panel
    private void jButtonSalirAyudaActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        jPanelAyuda.setVisible(false);
}

    //Este procedimiento visualiza el panel de ayuda. Se invoca al pulsar el botón “Ayuda”
    private void jButtonAyudaActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        jPanelAyuda.setVisible(true);
}

    //Este procedimiento borra  el área de presentación de resultados. Se invoca al pulsar el botón “Borrar”
    private void ClearButtonActionPerformed(java.awt.event.ActionEvent evt) {
        clr();
}

    //Este procedimiento realiza la consulta DNS y presenta el resultado.
    //Se invoca al pulsar el botón “Consultar”
    private void lookupButtonActionPerformed(java.awt.event.ActionEvent evt) {
        //Guardamos la hora actual
        long tiempoInicio = System.currentTimeMillis();
        try {
            //Recuperamos el dominio/host a buscar del formulario
            String dominio=dominioTextField.getText();
            if (dominio.length()>0){
                //añadimos el "." final para que tenga formato FQDN
```

```
dominio=dominio + ".";
//Recuperamos el tipo de registro solicitado
String Tipo = (String)jComboRegistro.getSelectedItem();
// Comprobamos si la busqueda es inversa
if (Tipo.equals("PTR")){
    // Verificamos si se ha introducido una IP válida
    if (esIP(dominio)){
        //invertimos la IP y añadimos "in-addr.arpa." al final
        dominio=getRevName(dominio);
    }else{
        // Si no hay una IP válida se informa del error
        prt("Para realizar búsquedas inversas se debe introducir una dirección IP válida");
    }
}
//Comprobamos si se ha introducido un servidor DNS para la consulta
//Si no se ha configurado un DNS específico recuperamos el del sistema
String dnsServer=dnsTextField.getText();
if (dnsServer.length()==0){
    ResolverConfig r= new ResolverConfig ();
    dnsServer = r.server();
    DNSsrv = r.servers();
}
Resolver res = new ExtendedResolver(srv);
//introducimos el dominio y el tipo de registro y realizamos la consulta
int dclass = DClass.IN;
int type = Type.value(Tipo);
org.xbill.DNS.Name name = org.xbill.DNS.Name.fromString(dominio);
Record rec = Record.newRecord(name, type, dclass);
Message query = Message.newQuery(rec);
Message response = res.send(query);
//Presentamos el resultado de la consulta
prt ("-------------------------------------------------------");
//Presentamos el servidor que ha respondido a la consulta
prt ("Servidor DNS: " + dnsServer );
//Verificamos que la respuesta no sea erronea
int error=response.getRcode();
```

```
        if (error==0){
            //Recuperamos la cabecera de la respuesta
            Header hd=response.getHeader();
            //Comprobamos el flag RA para saber si la respuesta es o no autorizada
            if (hd.getFlag(Flags.AA)){
                prt("Respuesta autorizada ");
            }else{
                prt("Respuesta no autorizada ");
            }
            prt ("-------------------------------------------------------");
            prt ("   Tipo\t Nombre");
            prt ("-------------------------------------------------------");
            //Recupramos las secciones de la respuesta
            //Y recuperamos los campos rdata de la secciones
          Record[] answers = response.getSectionArray(Section.ANSWER);
          for (int l=answers.length;l>0;l--){
              prt("   [" + Type.string(answers[l-1].getType()) + "]\t" + answers[l-1].rdataToString());
          }
          answers = response.getSectionArray(Section.AUTHORITY);
          for (int l=answers.length;l>0;l--){
              prt("   [" + Type.string(answers[l-1].getType()) + "]\t" + answers[l-1].rdataToString());
          }
          answers = response.getSectionArray(Section.ADDITIONAL);
          for (int l=answers.length;l>0;l--){
              prt("   [" + Type.string(answers[l-1].getType()) + "]\t" + answers[l-1].rdataToString());
           }
        }else {
            //Si la consulta no se ha realizado correctamente presentamos el código del error
            prt("Error: " + Rcode.TSIGstring(response.getRcode()));
        }
    }else{
        // Si no hay ninguna valor para la búsqueda que realizar se informa del error
        prt("Introduzca un el host o dominio para realizar la consulta.");
    }
} catch(Exception e) {
    prt(e.getMessage());
```

```
        }
        long totalTiempo = System.currentTimeMillis() - tiempoInicio;
        //prt ("-----------------------------------------------------");
        prt("");
        // Presentamos cuál ha sido la duración de la consulta
        prt("Respuesta en: " + totalTiempo + " miliseg");
        prt ("-----------------------------------------------------");
}

    static String getRevName(String ipAddr) {
    //Este procedimiento invierte la IP y la prepara para la búsqueda inversa
                String revName = null;
                String[] quads = ipAddr.split("\\.");
                ipAddr = "";
                for (int i = quads.length - 1; i >= 0; i--) {
                    ipAddr += quads[i] + ".";
                }
                ipAddr += "in-addr.arpa.";
                revName = ipAddr;
                return revName;
    }

    static boolean esIP (String ipAddr){
        //Este procedimiento verifica que la cadena es una dirección IP válida
        String octetos[] = ipAddr.split("\\.");
        if (octetos.length == 4) {
            return
(esNumeroIP(octetos[0])&&esNumeroIP(octetos[1])&&esNumeroIP(octetos[2])&&esNumeroIP(octetos[3]));
        }else{
            return false;
        }
    }

    static boolean esNumeroIP (String cadena){
        // Este procedimiento verifica si la cadena es un número entre 0 y 255
        try{
```

```
        int num = Integer.parseInt(cadena);
        if (num>=0&&num<=255)
            return true;
        else
            return false;
    }catch(Exception e){
        return false;
    }
}

void prt (String cadena){
    //Este procedimiento escribe en el área de resultados
    jTextArea1.append (cadena + "\n");
}

void clr (){
    //Este procedimiento borra el área de resultados
    jTextArea1.setText("");
}

// Variables declaration - do not modify
private javax.swing.JButton ClearButton;
private javax.swing.JTextField dnsTextField;
private javax.swing.JTextField dominioTextField;
private javax.swing.JButton jButtonAyuda;
private javax.swing.JButton jButtonSalirAyuda;
private javax.swing.JComboBox jComboRegistro;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
```

```
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanelAyuda;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextArea1;
    private javax.swing.JButton lookupButton;
    // End of variables declaration

}
```

--- FIN DEL DOCUMENTO ---