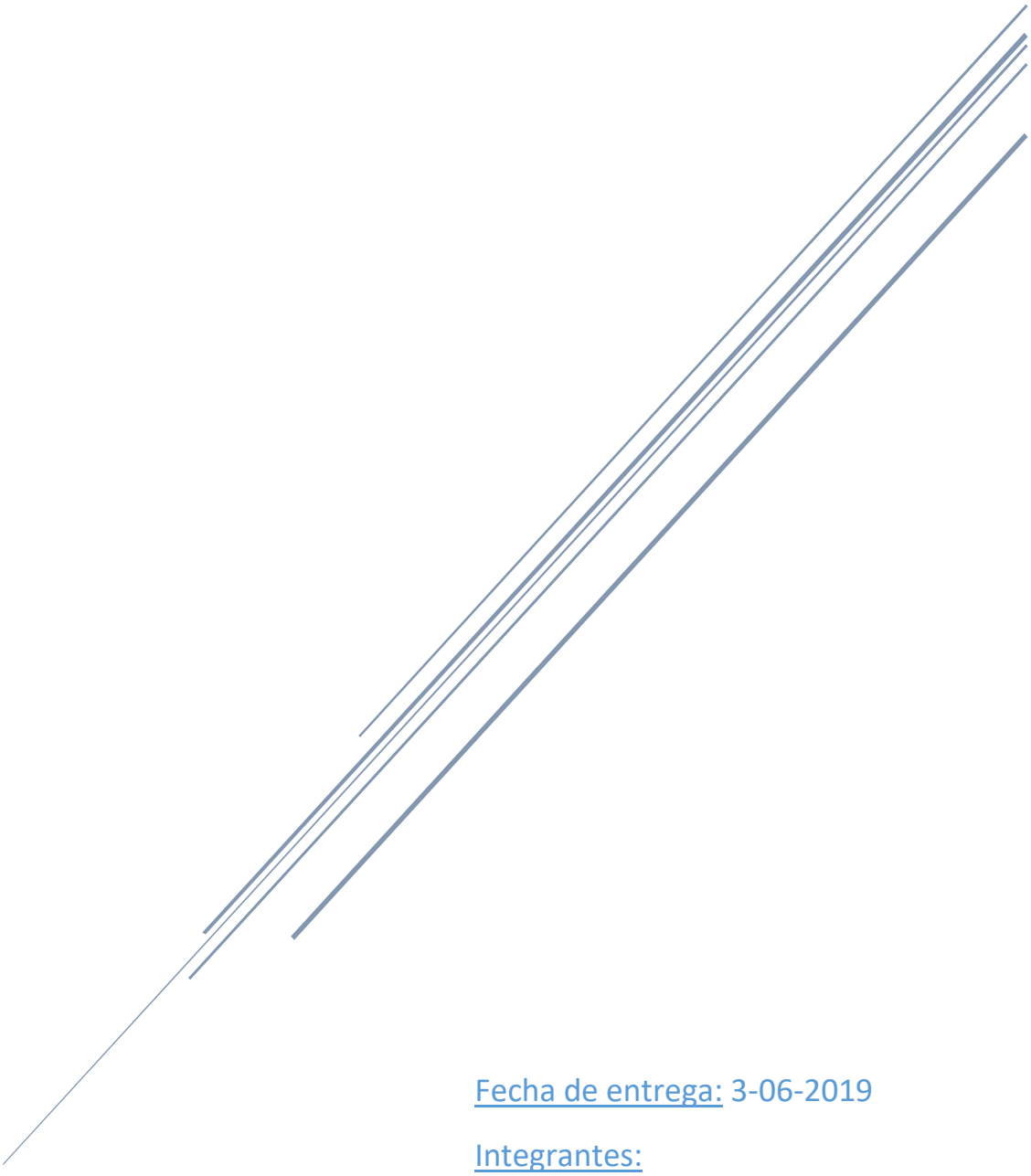
# REDES DE COMPUTADORAS

## PROYECTO Nº1



Fecha de entrega: 3-06-2019

Integrantes:


- ARTOLA, Bianca (111396)
- SUPERI, Agustina (110020)

# Índice

# Definiciones y especificación de requerimientos

## Definición general del proyecto de software

- **Idea general:** Empleando el lenguaje de programación C, se implementa un programa con el propósito de servir de herramienta de consulta y exploración de la jerarquía DNS bajo el sistema operativo GNU/Linux.
  El mismo se ejecuta desde la consola e interactúa con uno o más servidores DNS, respetando siempre el intercambio de mensajes estipulado en los RFC 1034 y RFC 1035.
- **Objetivos:** Lograr un funcionamiento similar al de la herramienta **dig**, interactuando con servidores DNS, permitiendo al usuario realizar las consultas que desee.
- **Usuarios:** Los usuarios del sistema serán el Asistente y/o los Ayudantes de la cátedra Redes de Computadoras de la Universidad Nacional del Sur. También puede utilizar el sistema cualquier persona que lo adquiera y que pueda seguir los pasos de obtención e instalación del mismo, detallados en la especificación del proyecto.

## Especificación de requerimientos del proyecto

- **Requisitos generales:** El proyecto de software realizado debe cumplir diferentes requerimientos a la hora de evaluar la entrada del usuario, ya que no toda entrada ingresada al sistema es capaz de resolverse.
  Hay diferentes opciones para que la entrada sea válida:

  **query consulta @servidor[:puerto] [-a | -mx | -loc] [-r | -t] [-h]**

  El significado de los diversos parámetros es el siguiente (recordar que las opciones entre corchetes resultan opcionales (los cuales pueden aparecer en cualquier orden) y las separadas por la barra vertical son alternativas excluyentes):
  - **consulta**: este argumento, el cual debe estar siempre presente salvo que se esté invocando a la ayuda a través del parámetro opcional –h, es la consulta que se desea resolver (en general, la cadena de caracteres denotando el nombre simbólico que se desea mapear a un IP).
  - **@servidor**: el cliente debe resolver la consulta suministrada contra el servidor DNS que se especifique con este argumento. Caso contrario, de no estar presente este argumento, se debe resolver la consulta suministrada contra el servidor DNS por defecto.
  - **[:puerto]**: este parámetro opcional, el cual sólo puede ser especificado si también se especificó un servidor, permite indicar que el servidor contra el cual se resolverá la consulta no está ligado al puerto DNS estándar. Caso contrario, al no indicarse el puerto que se deba usar se asumirá que las consultas serán dirigidas al puerto estándar del servidor (ya sea especificado a través del parámetro anterior o bien el configurado por defecto).
  - **[-a | -mx | -loc]:** estos parámetros opcionales, excluyentes entre sí, denotan el tipo de consulta que se está realizando. Si se especifica el parámetro -a, la consulta se trata de un nombre simbólico y se desea conocer su correspondiente IP numérico asociado. Si se especifica el parámetro -mx, se desea determinar el servidor a cargo de la recepción de correo electrónico para el dominio indicado en la consulta. Finalmente, si se especifica el parámetro -loc, se desea recuperar la información relativa a la ubicación geográfica del dominio indicado en la consulta. En caso de no indicarse el tipo de la consulta, se debe asumir por defecto que se trata de una consulta sobre los registros DNS de tipo A (es decir, el cliente se comporta como si se hubiese especificado el parámetro opcional –a.
  - **[-r | -t]**: estos parámetros opcionales, excluyentes entre sí, denotan la manera en la cual se desea resolver la consulta. En caso de especificarse el parámetro -r, se está solicitando que la consulta contenga el bit recursion desired activado (es decir, puede pasar una de dos cosas, o bien el servidor acepta la consulta recursiva y retorna la respuesta definitiva, o bien rechaza ese tipo de consulta, por lo que no se podrá obtener la respuesta definitiva bajo esas condiciones). Caso contrario, al especificarse el parámetro -t, se está solicitando que la consulta se resuelva iterativamente, mostrando una traza con la evolución de la misma (esto es, mostrando las distintas respuestas parciales que se van obteniendo al efectuar una consulta no recursiva hasta dar con la respuesta final, similar a la funcionalidad provista por el comando: dig con la opción --trace). Por último, en caso de no especificarse ni uno ni otro, se asumirá por defecto que la consulta debe ser resuelta de manera recursiva (es decir, retornando directamente la respuesta final).

- **[-h]:** si se especifica el parámetro opcional -h, el programa sólo deberá mostrar una pequeña ayuda por pantalla, la cual consiste en listar las opciones de invocación del programa (sintaxis) y cuál es su propósito. En caso de estar presente esta opción no se deberá iniciar el cliente, tan sólo se muestra el texto de ayuda.

## Procedimientos de instalación y prueba

### Procedimientos de desarrollo

Para la realización de este proyecto se han utilizado las siguientes herramientas:

- VMware Workstation 15 player
- Máquina virtual otorgada por la cátedra de Redes de Computadoras de la Universidad Nacional del Sur
- Visual Studio Code
- Terminal

Pasos realizados a lo largo de la resolución del proyecto:

- Comenzamos con la lectura y el análisis de múltiples fuentes de información. Se utilizó el archivo RFC-1035 para mayor entendimiento de la estructura de cada registro DNS.
- Cada nueva funcionalidad creada fue testeada a fin de verificar su funcionamiento individual; luego integrada al programa de base y nuevamente probada
- Ya avanzada la etapa de codificación, se comenzó a escribir la documentación.
- Cabe destacar que mientras escribíamos el código fuente del programa realizabamos los correspondientes comentarios a fin de aumentar la legibilidad del mismo.
- La técnica utilizada fue trabajo en equipo con división de tareas.

### Procedimientos de instalación y prueba

1. Obtener los archivos consulta.c, ayuda.c, dnsquery.c, dns.c, socket.c, DNS-service.c , consultaLOC.c, con sus correspondientes archivos .h , ayuda.txt y el archivo script.sh.
2. Ingresar a Terminal, y una vez ubicados en el directorio donde se encuentran los archivos del paso 1) [Dentro de la carpeta “Archivos.C”] ingresar por consola: **sh script.sh**
3. Ejecutar el programa ingresando **./dns <*consultaDNSquery*>**

#### *Aclaraciones importantes*

- El archivo **script.sh** ejecuta aquellos archivos **.c** utilizados para el funcionamiento del programa.
- Recordar que **consultaDNSquery** deberá contener el formato especificado anteriormente en la sección “Requisitos Generales”.

# Arquitectura del sistema

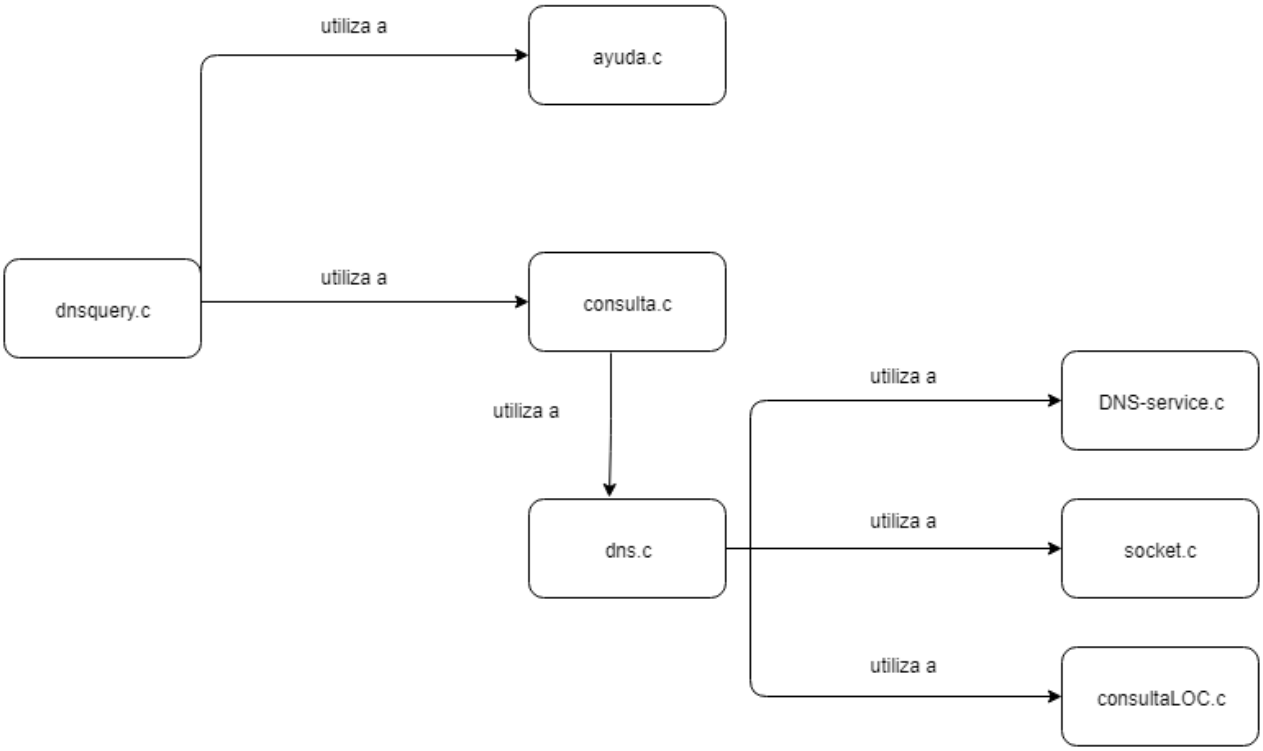
La lógica del programa fue separada en módulos con el fin de obtener un mejor entendimiento del mismo. Los distintos archivos se organizan en tres carpetas diferentes las cuales contienen los módulos:

- **Archivos.C:** ayuda.c, consultaLOC.c, consultas.c, DNS-service.c, dns.c, dnsquery.c, socket.c
- **Archivos.H:** ayuda.h, consultaLOC.h, consultas.h, DNS-service.h, dns.h, socket.h
- **Archivos.TXT:** ayuda.txt

Dentro de la carpeta Archivos.C, se encuentra el archivo con el nombre script.sh, ya mencionado anteriormente.

La interacción entre los mismo puede observarse en el siguiente diagrama:

## Diagrama de interacción




## Descripción individual de ayuda

Este módulo tiene como objetivo obtener un mensaje de ayuda desde un archivo y mostrarlo por pantalla. Se utiliza en los casos en los que el usuario ingresó mal la entrada al programa.

## Descripción individual de consultaLOC

Este módulo realiza la consulta de tipo LOC que fue pedida por el usuario. Se encarga de imprimir la información obtenida en un formato entendible por los humanos.

## Descripción individual de consultas

Este módulo contiene toda lógica relacionada a la evaluación de la entrada del usuario: se encarga de verificar que la entrada tenga un formato como el mencionado en la sección “Especificación de requerimientos del proyecto”.

## Descripción individual de DNS-service

Este módulo le provee diferentes servicios a la clase DNS, como por ejemplo asignar las distintas propiedades DNS al correspondiente struct. Todos los servicios que brinda se encuentran explicados en cada método de la clase.

## Descripción individual de dns

Este módulo es el principal de nuestra aplicación, ya que es el encargado de realizar toda la lógica (y derivar parte de ella) de la consulta DNS. Se encarga del manejo de los distintos servidores, realizando consultas en los mismos. También muestra los resultados en pantalla

## Descripción individual de dnsquery

Este módulo es nuestro módulo inicial. Desde aquí comienza toda la aplicación: se encarga de llamar al módulo "consultas.c" para evaluar si la consulta es correcta [Luego el módulo consultas.c por su parte se encarga de llamar a DNS para que siga el funcionamiento]. Si el resultado que le brinda "consultas.c" indica que la consulta fue errónea, este módulo se encarga de mostrar la ayuda [mediante la llamada a la clase "ayuda.c".

## Descripción individual de socket

Este módulo se encarga de la creación del socket: evalúa que este se cree de manera correcta, y si no lo hace, muestra el mensaje de error. Además, se encarga de asignarle un timer al envío y receptor del socket: con esto nos aseguramos de que, si el usuario ingreso un servidor o puerto erróneo, el programa no se quede a la espera infinitamente.

## Servicios externos que el software utiliza

Para la realización de nuestro proyecto hemos utilizado diversas librerías de c:

- **<stdio.h>:** Contiene tipos, macros y funciones para la realización de tareas de E/S.
- **<stdlib.h>:** Contiene tipos, macros y funciones para la conversión numérica, generación de números aleatorios, búsquedas y ordenación, gestión de memoria y tareas similares.
- **<string.h>:** Contiene tipos, macros y funciones para la manipulación de cadenas de caracteres.
- **<sys/socket.h>:** Permite utilizar la estructura **sockaddr.**
- **<arpa/inet.h>:** Permite utilizar la estructura **in_addr.**
- **<netinet/in.h>:** Permite utilizar el método htons().
- **<unistd.h>:** Librería que contiene funciones para el manejo de directorios y archivos.
- **<arpa/nameser.h>:** Permite utilizar las variables relacionadas al tipo de registro DNS, como ser, A-MX-LOC-SOA-NS, y la variable para indicar la clase de consulta, en este caso ns_c_in el cual indica internet.
- **<sys/types.h>:** Librería que contiene funciones de búsqueda y ordenamiento de directorios y manipulación de archivos.
- **<sys/time.h>:** Contiene tipos, macros y funciones para la la manipulación de información sobre fechas y horas.
- **<netdb.h>:** Permite utilizar el método gethostbyname().
- **pragma:** la función de pragma es instruir al compilador a empaquetar los miembros de un *struct* con una alineación particular. Tiene la siguiente sintaxis:

  #pragma pack( [ show ] | [ push | pop ] [, identifier ] , n )

  En nuestro proyecto utilizamos las funciones push y pop de la siguiente manera:

  - #pragma pack(push, 1)
  - #pragma pack(pop)

  PUSH: Empuja el valor de alineación de empaquetamiento actual en la pila del compilador interno y establece el valor de alineación de empaquetamiento actual en n. Si no se especifica n, se empuja el valor actual de alineación del empaque.

  POP: Elimina el registro de la parte superior de la pila del compilador interno. Si n no se especifica con pop, entonces el valor de empaquetado asociado con el registro resultante en la parte superior de la pila es el nuevo valor de alineación de empaquetamiento.

Pequeño ejemplo del funcionamiento:
Para un struct como el siguiente:

```
struct Test {
        char AA;
        int BB;
        char CC;
};
```

El compilador haria algo asi:

```
|   1   |   2   |   3   |   4   |

| AA(1) | pad.................. |
| BB(1) | BB(2) | BB(3) | BB(4) |
| CC(1) | pad.................. |
```

Con #pragma pack(1), La estructura anterior se presentaría de esta manera:

```
|   1   |

| AA(1) |
| BB(1) |
| BB(2) |
| BB(3) |
| BB(4) |
| CC(1) |
```

Con #pragma pack(2), se presenta así:

```
|   1   |   2   |

| AA(1) | pad.. |
| BB(1) | BB(2) |
| BB(3) | BB(4) |
| CC(1) | pad.. |
```

## Aspectos técnicos/ tecnologías empleadas

A la hora de crear este proyecto de software, utilizamos el lenguaje de programación C.

Se utiliza este lenguaje de programación ya que cuenta con la posibilidad de utilizar registros y punteros, lo cual, para lo que fue especificado a la hora de realizar el proyecto, nos facilita la realización del mismo.

## Diseño del modelo de datos

La entidad principal de nuestro sistema es el usuario que ingresa la expresión al sistema.

El usuario ingresa una expresión en el sistema por consola. Al ingresar la misma, el sistema comienza a trabajar sobre la misma: comienza evaluando si es válida. En caso de que no lo sea, se indica por pantalla un mensaje de ayuda, y se termina la ejecución de la misma.

**Mensaje de ayuda**

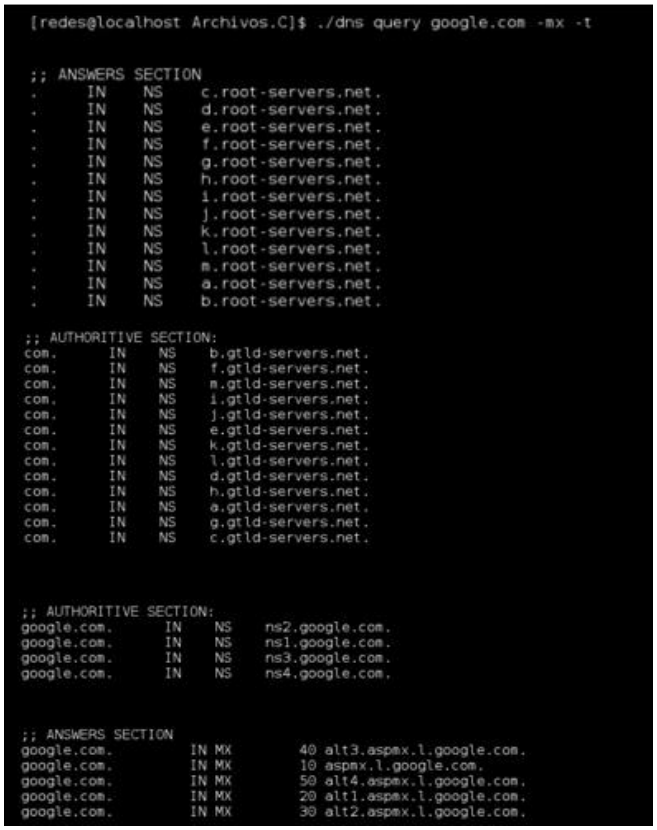
```
El formato de la consulta es:

--> query consulta @servidor[:puerto] [-a | -mx | -loc] [-r | -t] [-h]

Los parametros opcionales que se pueden ingresar son:

--> consulta: Consulta que se desea resolver.

-->@servidor: Servidor DNS por el cual se desea resolver la consulta.
-->[:puerto]: Permite indicar que el servidor contra el cual se resolvera la con
sulta
no esta ligado al puerto DNS estandar. Solo se debe especificar si se especifico
 un servidor.

-->-a: La consulta se trata de un nombre simbolico. Se desea conocer su correspo
ndiente IP numerico asociado.
-->-mx: Se desea determinar el servidor a cargo de la recepcion de correo electr
onico para el dominio indicado en la consulta.
-->-loc: Se desea recuperar la informacion relativa a la ubicacion geografica de
l dominio indicado en la consulta.

-->-r: Se solicita que la consulta obtenga el bit recursion desired activad.
-->-t: Se solicita que la consulta se resuelva iterativamente, mostrando una tra
za con la evolucion de la misma.

IMPORTANTE: Recordar que las opciones entre corchetes ([]) resultan opcionales,
y las separadas por la barra vertical (|) son alternativas excluyentes.
```

El “-h” puede ingresarse en cualquier parte de la consulta. Por ejemplo, las siguientes entradas son consideradas como válidas:

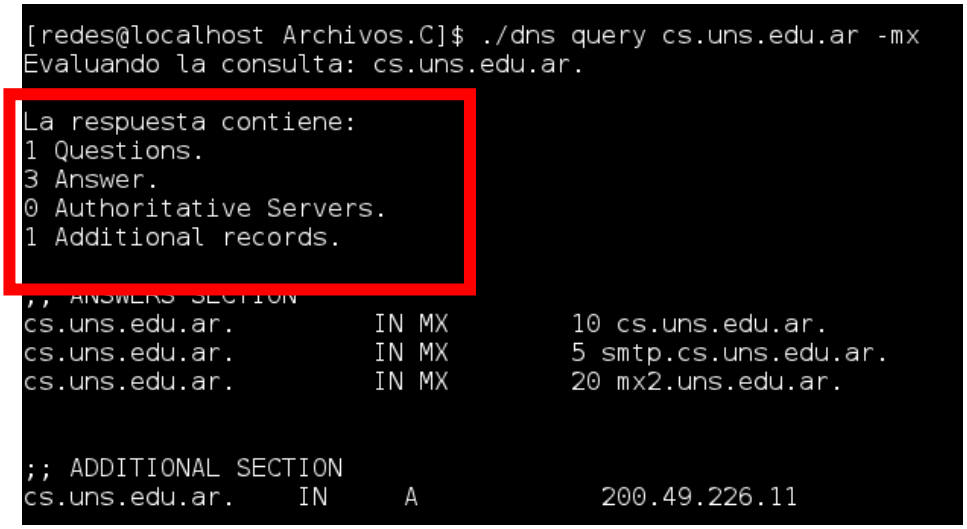
- ./dns query -h
- ./dns query www.google.com -h
- ./dns query www.google.com -a -h
- ./dns query www.google.com @8.8.8.8 -h

En dichos casos, SOLO se mostrará la ayuda del programa, ignorando la consulta realizada por el usuario, ya que se asume que el mismo está pidiendo ayuda para realizarla.

En el caso de que la expresión sea válida, se podrá observar por pantalla el resultado de la misma.
Por ejemplo, si el usuario ingresa la siguiente consulta: **query google.com -mx -t,** la salida será la siguiente:



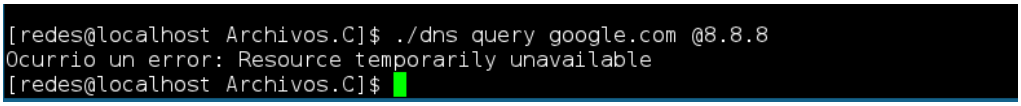

En la respuesta a consultas de tipo recursivas, nuestro programa muestra la siguiente información:




Donde:

- Questions indica la cantidad de consultas realizadas por el usuario. en nuestro caso, siempre será igual a 1.
- Answers indica la cantidad de respuestas de tipo “Answer” que el programa encontró para esa consulta.
- Authoritive indica la cantidad de respuestas de tipo “Authoritive” que el programa encontró para esa consulta.
- Additional indica la cantidad de respuestas de tipo “Additional” que el programa encontró para esa consulta.

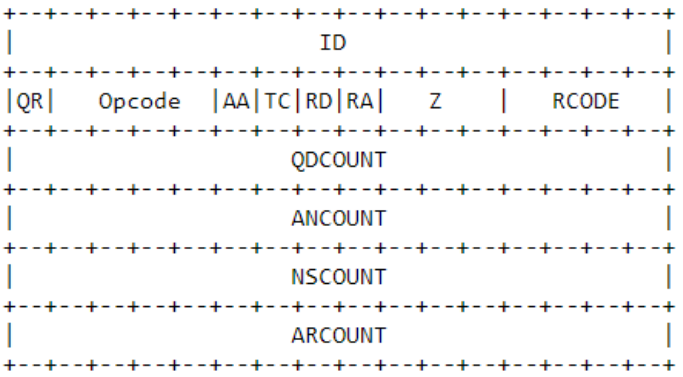
Cuando el usuario ingresa un servidor o puerto erróneo, se muestra el siguiente resultado por pantalla:



# Descripción de variables, struct, define, registros ofrecidos por el sistema.

## Struct

### Formato del header DNS

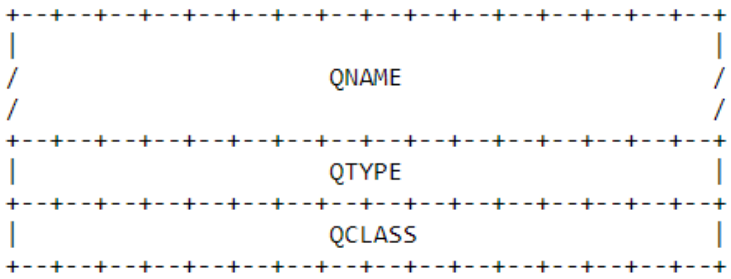



- **ID:** identificador de 16 bits asignado por el programa que genera cualquier tipo de query.
- **QR:** campo de un bit que especifica si el mensaje es una query (0) o una respuesta (1).
- **Opcode:** campo de cuatro bits que especifica el tipo de query en el mensaje. Los posibles valores son:
  - **0** query estándar (QUERY)
  - **1** query inversa (IQUERY)
  - **2** request de estado del servidor (STATUS)
  - **3-15** reservado para uso futuro
- **AA - Authoritive Answer:** este bit es válido en respuestas, y especifica que el servidor de respuesta es autoritativo para el nombre de dominio en la consulta.
- **TC - TrunCation:** especifica que este mensaje se truncó debido a una longitud mayor que la permitida en el canal de transmisión
- **RD - Recursion Desired:** bit de recursión. Si se establece RD, indica al servidor de nombres que busque la consulta de forma recursiva. El soporte de consultas recursivas es opcional.
- **RA (Recursion Available):** Indica si el servidor de nombres soporta consultas recursivas.
- **Z:** reservado para uso futuro. Debe ser 0 en todas las queries y respuestas.
- **RCODE - Response Code:** Es un campo de 4 bits. Sus valores tienen la siguiente interpretación:
  - 0 No hay condición de error
  - 1 Error de formato - El servidor de nombres fue incapaz de interpretar la consulta.
  - 2 Falla del servidor - El servidor de nombres fue incapaz de procesar la consulta debido a un problema con el servidor de nombre.
  - 3 Error de nombre
  - 4 No implementado - El servidor de nombres no soporta ese tipo de consulta
  - 5 Rechazado
- **QDCOUNT:** Entero no signado de 16 bits especificando el número de entradas en la sección *question.*
- **ANCOUNT:** Entero no signado de 16 bits especificando el número de registros en la seccion *answer. En nuestro código lo llamamos "ans_count"*
- **NSCOUNT:** Entero no signado de 16 bits especificando el número de name servers en la sección de registros *authoritives. En nuestro código lo llamamos "auth_count"*
- **ARCOUNT:** Entero no signado de 16 bits especificando el número de registros en la sección de registros *additional. En nuestro código lo llamamos "add_count"*

En nuestro código agregamos dos campos más:

- CD: Checking Disabled
- AD: Authenticated Data

### Formato de la sección de preguntas: struct QUESTION



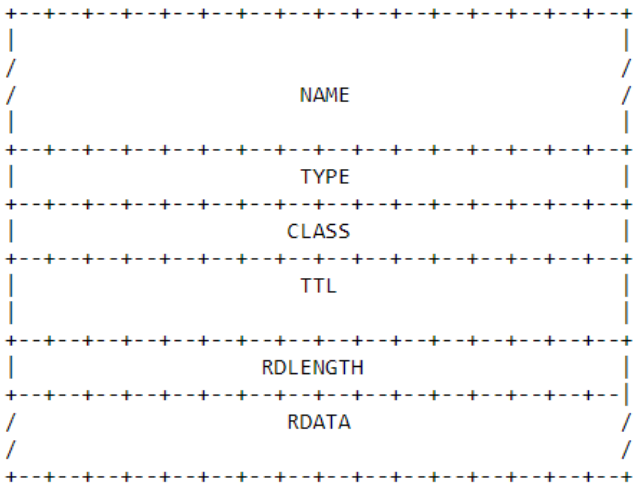

Donde

- **QNAME:** Un nombre de dominio representado como una secuencia de etiquetas, donde cada etiqueta consta de un octeto de longitud seguido de ese número de octetos. El nombre de dominio termina con el octeto de longitud cero para la etiqueta nula de la raíz.
- **QTYPE:** Especifica el tipo de la consulta.
- **QCLASS:** Especifica la clase de la consulta. Por ejemplo, el campo IN para Internet.

En nuestro caso solo utilizamos los parámetros qtype y qclass.

### Formato de registro de recursos

Las secciones *answer, additional y authoritive* comparten el mismo formato. Cada registro de recursos tiene el siguiente formato:




- **NAME:** un nombre de dominio al que pertenece este registro de recursos.
- **TYPE:** dos octetos que contienen uno de los códigos de tipo RR.
- **CLASS:** dos octetos que especifican la clase de los datos en el campo RDATA.
- **TTL:** un entero sin signo de 32 bits que especifica el intervalo de tiempo (en segundos) que el registro de recursos puede almacenarse en caché antes de que se deba descartar
- **RDLENGTH:** un entero de 16 bits sin signo que especifica la longitud en octetos del campo RDATA.
- **RDATA:** una cadena de longitud variable de octetos que describe el recurso.

## Define

### C_RECURSIVA

Se le asigna el número 1 indicando que la consulta se resuelve de manera recursiva. Se indica a través de la variable **-r.**

### C_ITERATIVO

Se le asigna el número 0 indicando que la consulta se resuelve de manera iterativa. Se indica a través de la variable **-t.**

### PUERTO_DEFECTO

En caso de que el usuario no ingrese un número de puerto, se le asigna a la consulta el puerto DNS 53.

### TIPOCONSULTA_DEFECTO

En caso de que el usuario no ingrese un tipo de consulta, se asignará por defecto el registro **-a.**

### TIPORESOLUCION_DEFECTO

En caso de que el usuario no ingrese un tipo de resolución de consulta, se asignará por defecto el recursivo **-r.**

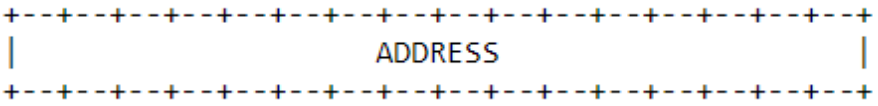
## Registros DNS

Los registros DNS que serán utilizados en el sistema son:

### Tipo A

Este registro se usa para traducir nombres de servidores de alojamiento a direcciones IPv4.

**Formato de A RDATA**




Donde

- **ADDRESS** es una dirección de Internet de 32 bits. Los hosts que tienen varias direcciones de Internet tendrán varios registros A.

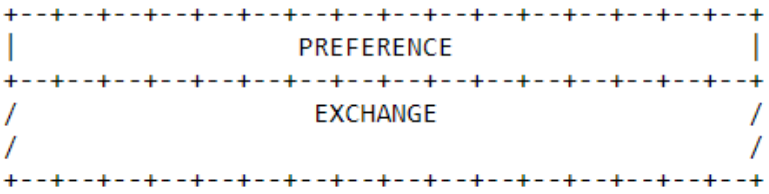
Un registro no causa procesamiento adicional.

Dicho registro es utilizado mediante la variable **ns_t_a,** la cual se encuentra definida en arpa/nameser.h.

### Tipo MX

Intercambio de correo (*mail exchange*). Asocia un nombre de dominio a una lista de servidores de intercambio de correo para ese dominio.

**Formato de MX RDATA**




Donde

- **PREFERENCE:** Un entero de 16 bits que especifica la preferencia dada a este RR entre otros del mismo propietario. Se prefieren valores más bajos.
- **EXCHANGE:** Un <nombre_de_dominio> que especifica un host que desea actuar como un intercambio de correo para el nombre del propietario.

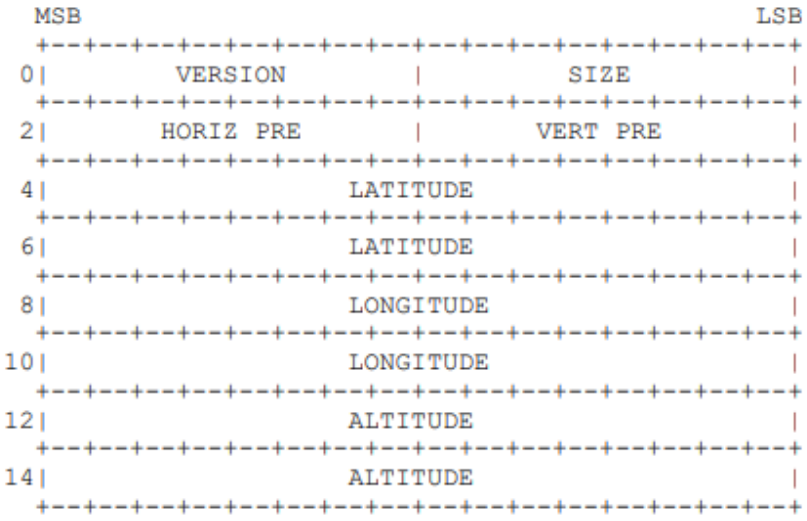
Los registros MX provocan el procesamiento de una sección adicional para el host especificado por EXCHANGE.

Dicho registro es utilizado mediante la variable **ns_t_mx,** la cual se encuentra definida en arpa/nameser.h.

### Tipo LOC

Especifica una ubicación geográfica asociada con un nombre de ámbito.

**Formato RDATA**




Donde

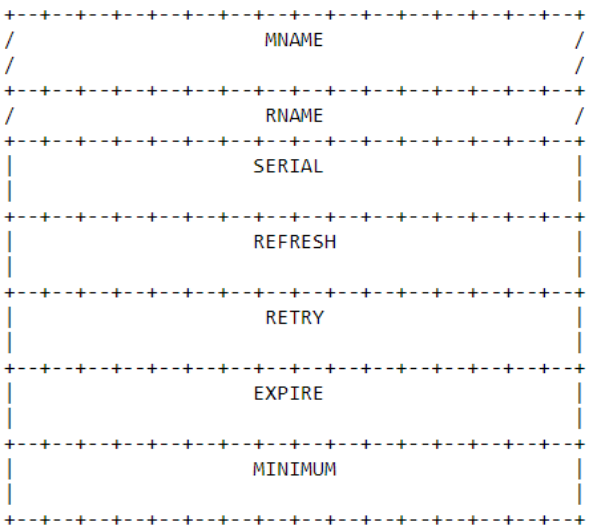
- **VERSION:** Número de versión de la representación
- **SIZE.**
- **HORIZ PRE:** la precisión horizontal de los datos en centímetros, expresado usando la misma representación que *size*
- **VERT PRE:** la precisión vertical de los datos en centímetros, expresado usando la misma representación que *size*.
- **LATITUDE.**
- **LONGITUDE.**
- **ALTITUDE.**

Dicho registro es utilizado mediante la variable **ns_t_loc,** la cual se encuentra definida en arpa/nameser.h.

### Tipo SOA

Autoridad de la zona (*start of authority*). Proporciona información sobre el servidor DNS primario de la zona.

**Formato RDATA**



Donde

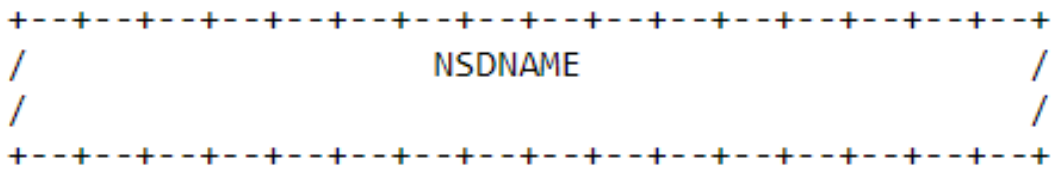
- **MNAME:** El **<nombre-de-dominio>** del servidor de nombres que fue la fuente de datos original o principal para esta zona.
- **RNAME:** Un **<nombre_de_dominio>** que especifica el buzón de la persona responsable de esta zona.
- **SERIAL:** El número de versión no signado de 32 bits de la zona. Las transferencias de zona conservan este valor.
- **REFRESH:** Un intervalo de tiempo de 32 bits antes de la zona se debe actualizar.
- **RETRY**: Se debe reintentar un intervalo de tiempo de 32 bits que debe transcurrir antes de una actualización fallida.
- **EXPIRE:** Un valor de tiempo de 32 bits que especifica el límite superior en el intervalo de tiempo que puede transcurrir antes de que la zona ya no tenga autoridad.
- **MINIMUM:** El campo TTL mínimo de 32 bits no signado que se debe exportar con cualquier RR de esta zona.

Dicho registro es utilizado mediante la variable **ns_t_soa,** la cual se encuentra definida en arpa/nameser.h.

Para dicho registro se creó un struct con los parámetros serial, refresh, retry, expire y minimum para obtener los datos de la dicha respuesta.

### Tipo NS

Servidor de nombres (name server). Define la asociación que existe entre un nombre de dominio y los servidores de nombres que almacenan la información de dicho dominio. Cada dominio se puede asociar a una cantidad cualquiera de servidores de nombres.

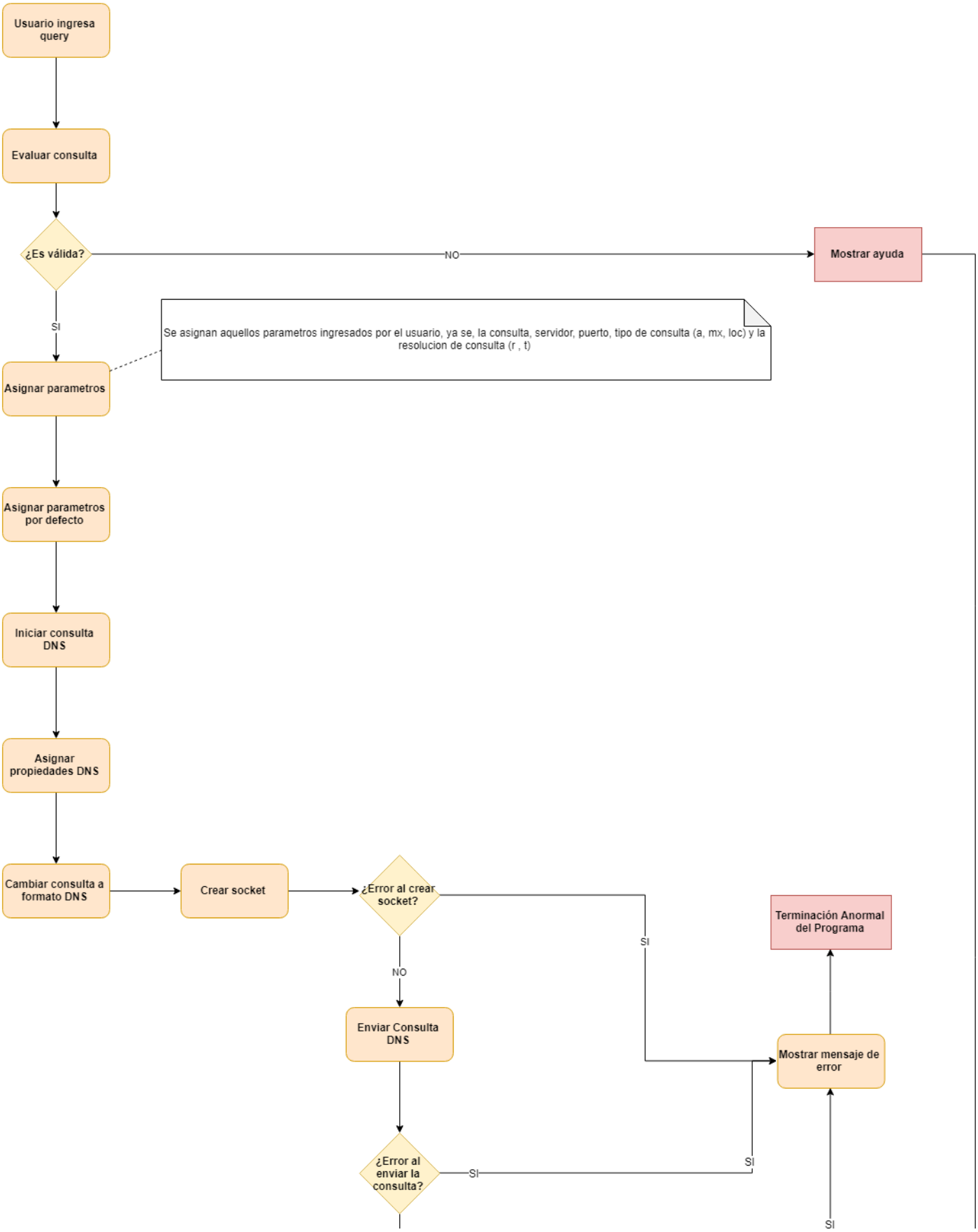


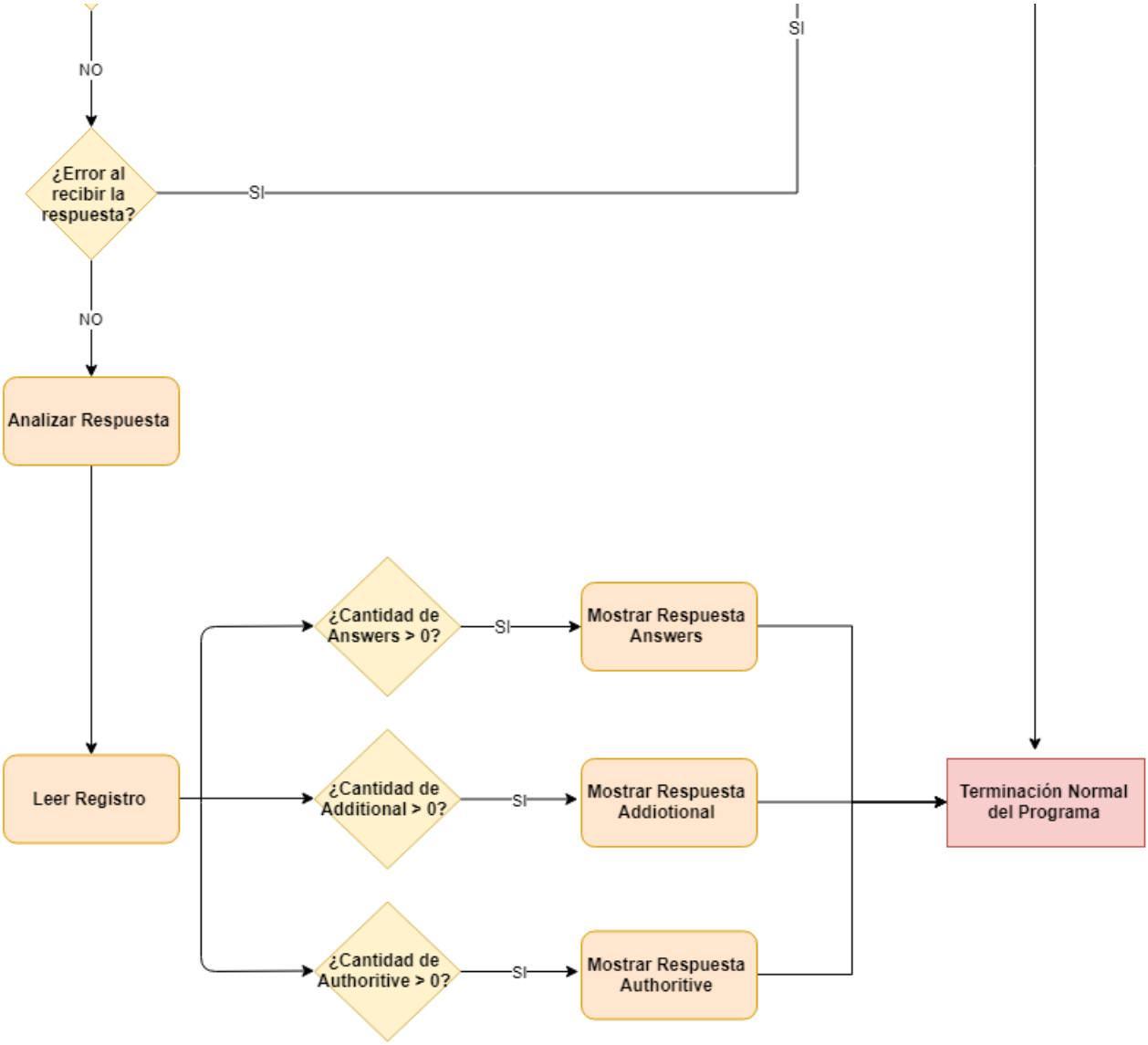

Donde

- **NSDNAME:** Un <nombre_de_dominio> que especifica un host que debe tener autoridad para la clase y el dominio especificados.

Dicho registro es utilizado mediante la variable **ns_t_ns,** la cual se encuentra definida en arpa/nameser.h.

# Diagrama en alto nivel correspondiente al funcionamiento del programa






Para mayor comprensión se adjunta la imagen correspondiente al diagrama.

## Decisiones de diseño

1. Cuando el usuario especifica un servidor y/o puerto el programa debe evaluar si puede realizar la consulta con esos datos. Para esto, a la hora de evaluar la consulta el programa contiene un timer con una duración de 30’ para crear el socket con los correspondientes datos. Por lo que, cuando la respuesta supera el límite de tiempo especificado, el programa será abortado debido a un ingreso incorrecto de los mismos.

## Conclusiones

Con la resolución de este proyecto, logramos obtener un mayor conocimiento sobre el manejo de servidores DNS, así como también sobre el manejo de sockets, sus funciones y sus limitaciones.

Además, profundizamos nuestros conocimientos en el lenguaje de programación C, ya que habíamos trabajado anteriormente en el mismo, por lo cual ya teníamos una base del mismo, pero el proyecto nos permitió aprender algunas nuevas funciones y librerías.

Por último, pudimos conocer en profundidad el funcionamiento de la herramienta **dig**, y logramos comprender los resultados de la misma.