

# Servidor DNS

Un DNS (Domain Name System) es un sistema que traduce nombres de dominio (como por ejemplo google.com) en direcciones IP (10.3.45.23). Traduce una dirección fácil de recordar en su dirección IP real donde realmente está el servidor.

Los servidores DNS almacenan y sirven distintos tipos de registros DNS:

## A

Asocian un nombre de dominio a una dirección  
example.com -> 93.184.216.34

## MX (mail exchanger)

Indica qué servidores reciben correo electrónico del dominio. Incluyen una prioridad aunque en esta práctica no se tendrá en cuenta.  
example.com -> mail1.example.com

## CNAME

Crea un alias de un dominio a otro. Es útil cuando varios subdominios apuntan a un mismo servidor.  
www.example.com -> example.com

Cuando se hace una solicitud a un servidor DNS, el servidor contesta primero con el código de respuesta y luego el resultado de la petición. Los códigos pueden ser los siguientes:

Código	Significado	Ejemplo
200	Éxito	200 <mensaje>
404	No encontrado	404 Not Found
400	Petición incorrecta	400 Bad Request
500	Error de servidor	500 Server error
150	Inicio de listado	150 Inicio listado <registros...> 226 Fin listado
226	Fin de listado	

## Fase 1: Servidor DNS mínimo. Sólo acepta una conexión simultánea y un único tipo de registro

En un archivo, almacena un conjunto de pares dominio-IP. El servidor debe leer este archivo y mantener un diccionario con el contenido del fichero. Para que te resulte sencillo los siguientes pasos, te aconsejo que definas el diccionario como `HashMap<String, List<Registro>>`, donde Registro debe almacenar el tipo (A/CNAME/MX) y el valor.

Ejemplo del fichero:

```
google.com A 172.217.17.4
educa.madrid.org A 193.146.123.119
example.com A 20.33.123.44
chatgpt.com A 172.64.155.209
```

El servidor debe ser capaz de:

1. Esperar una conexión
2. Recibe una línea con el siguiente formato:

```
LOOKUP <tipo> <dominio>
```

```
LOOKUP A google.com
```

3. Si existe, responde:  
**200** 172.217.17.4

Si no existe, responde:  
**404** Not Found

Si durante el procesamiento de la petición se produce alguna excepción, siempre debe contestar:  
**500** Server error

4. Si recibe **EXIT** -> cierra la conexión
5. Si recibe cualquier otra cosa que no respete el formato contesta:  
**400** Bad request

## Fase 2: Añadir más comandos y validaciones

Implementa el comando LIST. Devuelve todas las entradas.

Cliente manda:

```
LIST
```

Servidor contesta:

**150** Inicio listado

google.com A 172.217.17.4

educa.madrid.org A 193.146.123.119

example.com A 20.33.123.44

chatgpt.com A 172.64.155.209

**226** Fin listado

## Fase 3: Soportar varios tipos de registro DNS

Guarda en el archivo más tipos de registros DNS.

```
google.com A 172.217.17.4
educa.madrid.org A 193.146.123.119
example.com A 20.33.123.44
chatgpt.com A 172.64.155.209
wikipedia.org A 185.15.58.226
wikipedia.org MX mx1.mail.com
wikipedia.org MX mx2.mail.com
```

Ejemplo:

El cliente solicita LOOKUP A wikipedia.org	El servidor contesta <b>200</b> 185.15.58.226
-----------------------------------------------	--------------------------------------------------

El cliente solicita LOOKUP MX chatgpt.org	El servidor contesta <b>404</b> Not found
----------------------------------------------	----------------------------------------------

El cliente solicita LOOKUP MX wikipedia.org	El servidor contesta <b>200</b> mx1.mail.com <b>200</b> mx2.mail.com
------------------------------------------------	----------------------------------------------------------------------------

## Fase 4: Registro dinámico

Permite al cliente que introduzca nuevos registros en el servidor. El servidor debe permitir recibir un nuevo comando:

```
REGISTER <dominio> <tipo> <valor>
```

Ejemplo:

```
REGISTER curso.local A 192.14.32.55
```

Servidor:

```
200 Record added
```

El servidor debe añadir el nuevo registro a su diccionario y también debe escribirlo en el fichero.

## Fase 5: Servidor DNS multihilo (multicliente)

Modifica el código para permitir hasta 5 clientes simultáneos. Cuando un cliente finalice con EXIT, el servidor estará de nuevo disponible para una nueva conexión.

## IMPORTANTE: Indicaciones de entrega

1. Crear una cuenta de GitHub.
2. Crea un nuevo repositorio con el siguiente formato de nombre:  
dns-simplificado-nombre-alumno
3. Cada vez que trabajes y avances en el proyecto, realiza un commit:  
git add .  
git commit -m "Descripción del cambio"  
git push
4. Comparte el repositorio con el profesor como colaborador:
  - a. En tu repositorio, ve a Settings
  - b. En el menú de la izquierda, selecciona Collaborators
  - c. Pulsa "Add people"
  - d. Introduce al usuario: soniaalmodovarvialas
5. Crea un tag para cada fase  
git tag v1.0-fase1

Al final del proyecto tendrás que tener las tags que se describen a continuación:

Fase	Descripción	Nombre del tag
1	Servidor simple, 1 conexión, solo A	v1.0-fase1
2	Comando LIST	v1.1-fase2
3	Soporte de tipos DNS (CNAME, MX)	v1.2-fase3
4	REGISTER dinámico	v1.3-fase4
5	Multicliente	v1.4-fase5