NIC

[r_i, r_f]

asigno ips

(df_j, r_i)

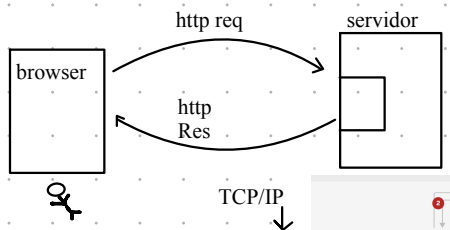
ips dinámica -> clientes
ips fijas -> servidores

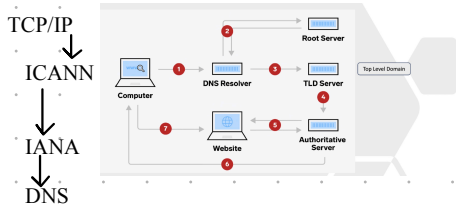
(df_j, r_i)

TCP/IP

emol

localhost es 127.0.0.1 es home





Conceptos Clave

- HTTP – URL – HTML tecnologías base
- IP v4 y v6, las direcciones de los recursos en WWW
- ICANN-IANA, la organización de entrega de rangos IP en el mundo
- DNS, arquitectura en forma de "bosque" para encauzar el tráfico
- TCP, UDP y QUIC, los protocolos de transporte (a nivel de red)
- Versiones de HTTP (1.1., 2 y 3) arquitectura y problemas que resuelven
- DNS y como llegar de un punto a otro

HTTP

HyperText Transfer Protocol

Es EL protocolo que permite intercambio de información en la Web

Hipertexto

Acceso recursos (documentos) mediante vínculos (links)

Link: texto que, según un formato específico, contiene referencia a un recurso/documento

get es el default

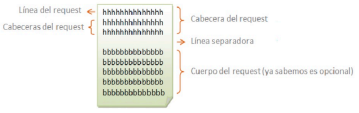
REQUEST (Solicitud)

Para HTTP hay dos tipos de mensajes: La solicitud y la respuesta (así de simple)

Partamos por la Solicitud, o mejor conocida como REQUEST

La sintaxis de request: metodo_del_request URI versión-HTTP

- *Método del request: El protocolo HTTP tiene varios métodos de request: GET, POST, HEAD, otros*
- *URI (Uniform Resource Identifier): Especifica el recurso que se requiere (URI no es URL)*
- *versión-HTTP*



https://…com?..&…&…

parámetros

URI

URL URN

¿Hay "body" en un request?

https://www.java67.com/2013/01/difference-between-url-uri-and-urn.html

MÉTODOS PARA REQUEST

Cada uno puede verse en el RFC respectivo.... Veamos los RFC

- **GET** : El cliente manda la solicitud de obtener (recibir) el recurso solicitado al servidor.
- **HEAD** : En este caso se solicita la cabecera que la envíe lo que el GET hubiese obtenido (esto permite revisar cache comparando fechas en la cabecera)
- **POST** : Solicitud de hacer un POST de datos (información) hacia el servidor Web (formularios/login acá)
- **PUT** : La solicitud de almacenar datos en el servidor (insert acá).
- **PATCH**: La solicitud de actualizar (update) la información)
- **DELETE** : Borrar datos del servidor.
- **OPTIONS** : Se pide un listado con los métodos de request que el servidor soporta/implementa.
- **CONNECT** : Esto se usa a menudo para establecer conexiones seguras vía SSL. Se solicita realizar la conexión hacia el servidor y se retorna información pero no se hace nada realmente con ella.

Códigos de respuesta

RESPONSE (Respuesta)

Línea de estado (Status): HTTP-version código explicación

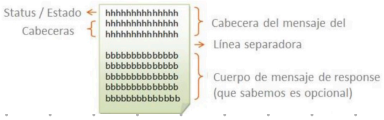
- Código de estado/status: un número de tres dígitos (típico 404).
- Explicación: Pequeña explicación del código enviado.

Ejemplo de línea de status: HTTP/1.1 200 OK o bien, el clásico: HTTP/1.0 404 Not Found

Las cabeceras de respuesta vienen del estilo *llave:valor, llave:valor…*

Ejemplo: : Content-Type: text/html Content-Length: 35

El cuerpo de la respuesta contiene el recurso/información solicitada.



- **Códigos de cien : 1xx (de Información): Solicitud recibida.**
- **Códigos de doscientos 2xx (de éxito): Solicitud recibida exitosamente.**
- **Códigos de trescientos 3xx (de redireccionamiento): Para completar la solicitud se debe tomar algún tipo de acción.**
- **Códigos de cuatrocientos 4xx (Error lado del cliente): Hasta acá no más llegamos. Hay errores desde el origen de la solicitud.**
- **Códigos de quinientos 5xx (Error lado del servidor): El servidor no pudo responder lo que parece ser una solicitud válida.**

Los recursos

Uniform Resource Locator (URL)

La URL Se usa para identificar de forma inequívoca un "lugar" o recurso en la Web (sitio, página).
Una URL típica tiene la siguiente forma (sintaxis):

protocolo://hostname:puerto/ruta-archivo

1.Protocolo: Puede ser FTP, HTTP y/o telnet.

2.Hostname: Un nombre www.milugar.cl (dado por un DNS) o una dirección IP (127.0.0.1).

Los recursos

3.Puerto: Es un número que identifica un lugar físico donde el servidor en el host "escucha" para una petición en particular.

Los puertos disponibles van del 1 al 65535 (en Linux al menos). Los puertos del 1 al 1023 son puertos entre reservados y con privilegios, es decir se requiere ser root para escuchar por ellos.

4.Ruta-archivo: El nombre y la ubicación del recurso solicitado.

Paths

Convención para orden jerárquico

http://algun.sitio.edu:3000/general/especifico?q=keyword&page=3

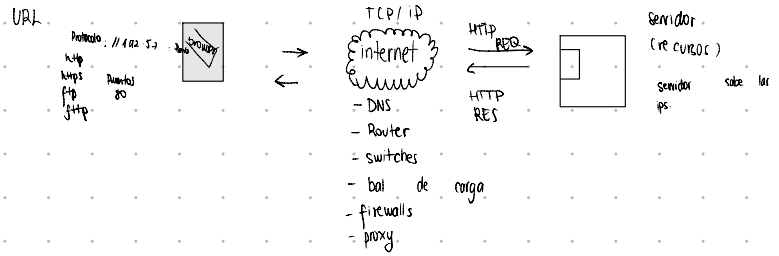
- SCHEME — Protocolo
- AUTHORITY — (dominio o IP)
- PATH — Secuencia jerárquica
- QUERY — Parámetros key=value

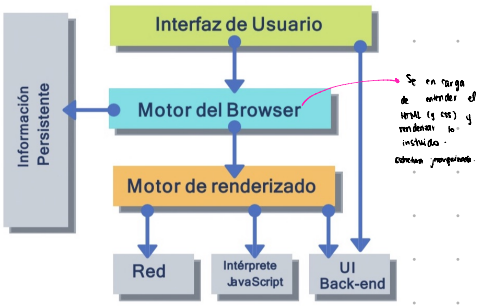
`/users` → Colección

`/users/:id` → Un elemento de la colección

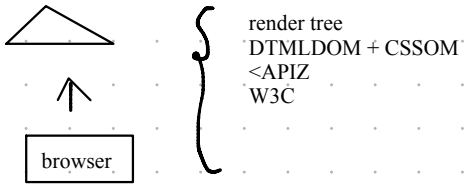
`/users/:id/posts` → Colección como sub-recurso

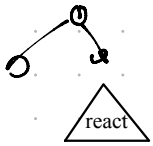
`/users/:userId/posts/:id` → sub-recurso





La APIR recibe cosas y devuelve HTML
Misma estructura que se la doy la API.
REACT tiene apis que hacen eso, es un framework REACT





componentes