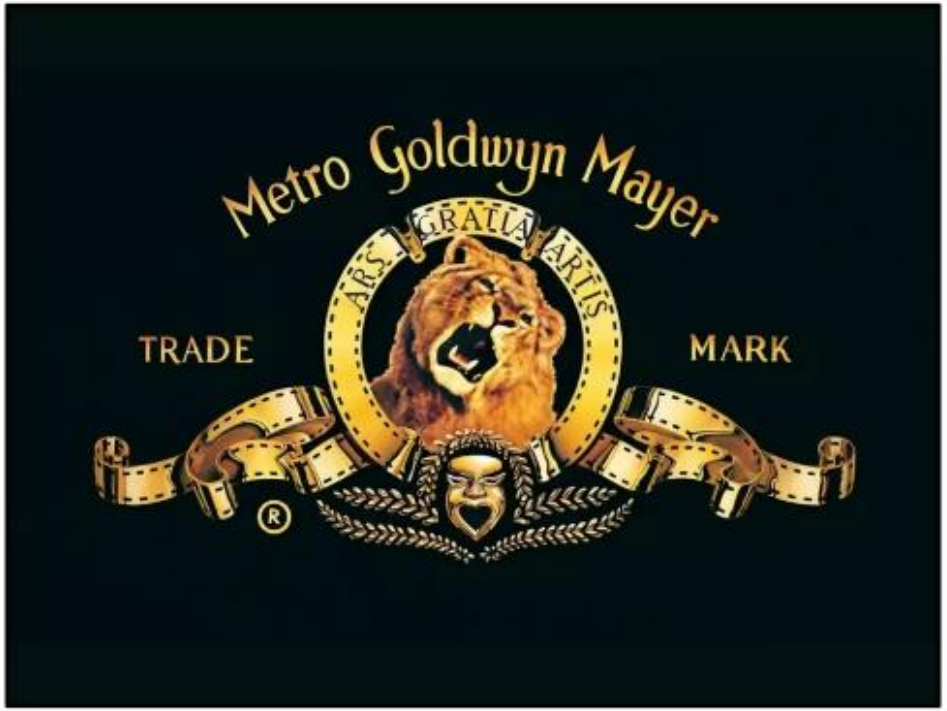
Programación 2

Introducción a las APIs



APPLICATION PROGRAMING INTERFACE

Conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

=

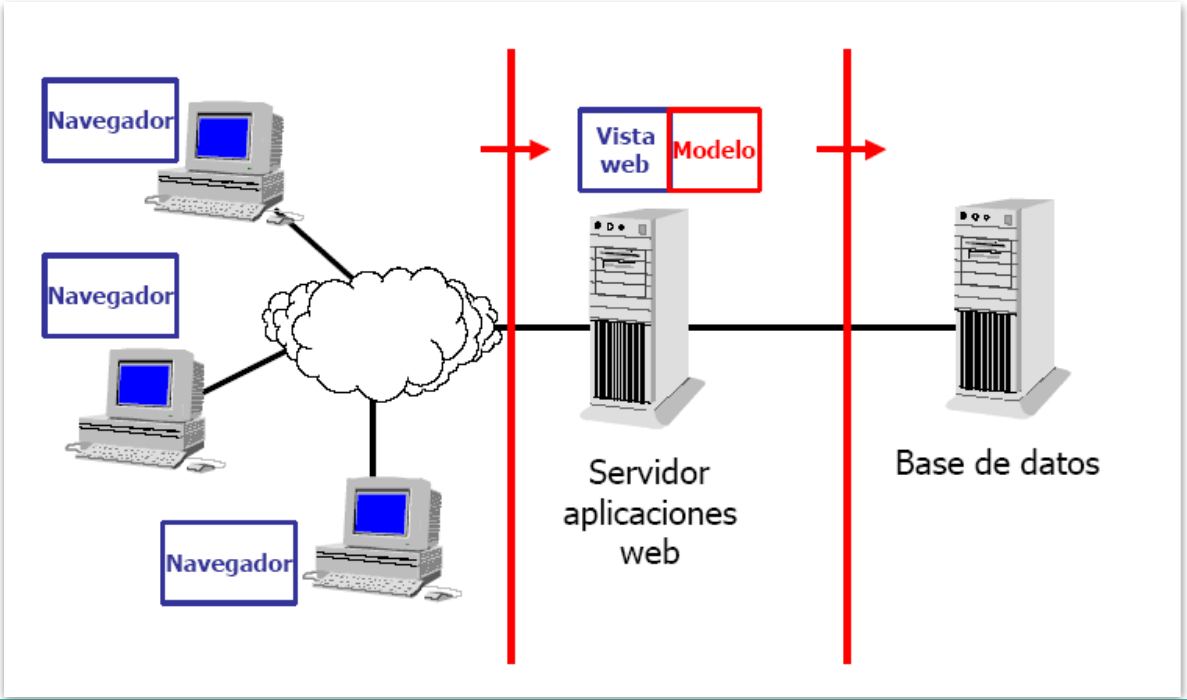
La cara de una plataforma hacia el mundo exterior.

Nos abstraemos de las capas más internas de la plataforma.

LO QUE IMPORTA ES EL QUÉ, NO EL CÓMO

Funcionamiento de las aplicaciones web

Antes de ver qué es una API, veamos cómo funcionan las aplicaciones web.



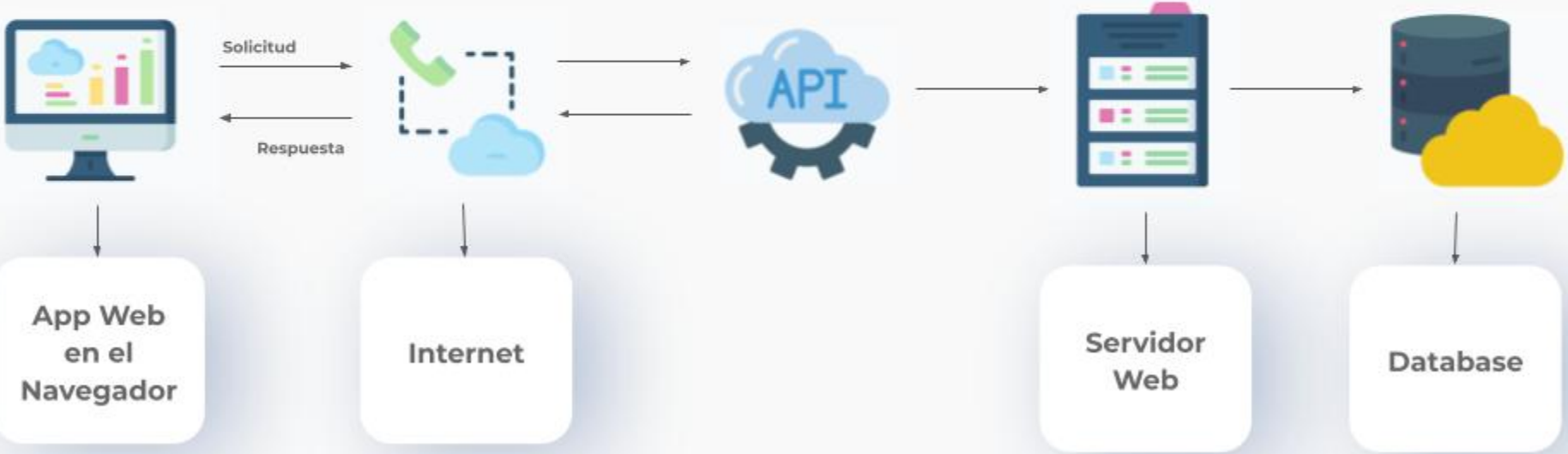
The easiest way to understand frontend, backend and API

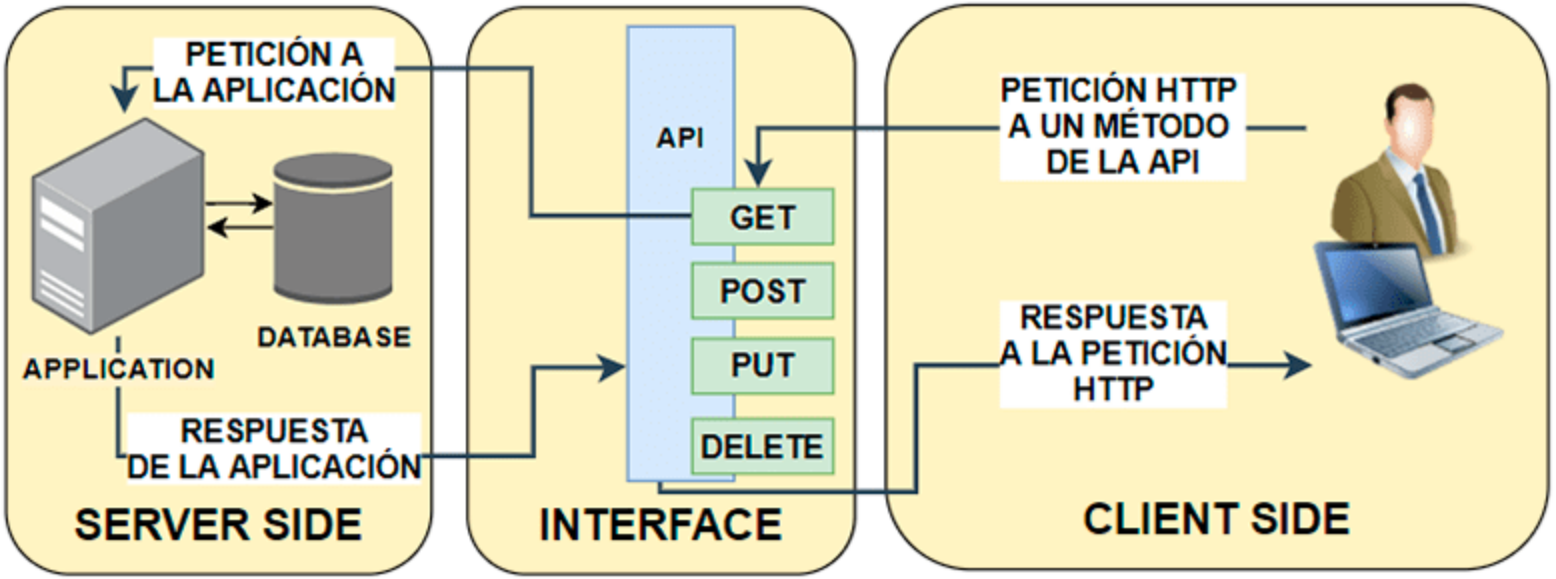
Definición de API.

API significa Interfaz de Programación de Aplicaciones en inglés, Application Programming Interface. Es un conjunto de reglas y definiciones que permite que diferentes aplicaciones informáticas se comuniquen entre sí. En otras palabras, una API especifica cómo los componentes de software deben interactuar.

Las APIs se utilizan comúnmente para permitir la integración entre diferentes sistemas y aplicaciones, lo que permite que programas desarrollados en diferentes lenguajes de programación y en diferentes plataformas se comuniquen y compartan datos entre sí.

¿Cómo trabaja un API?




PETICIÓN A
LA APLICACIÓN
APPLICATION
DATABASE
RESPUESTA
DE LA APLICACIÓN
SERVER SIDE
API
GET
POST
PUT
DELETE
INTERFACE
PETICIÓN HTTP
A UN MÉTODO
DE LA API
RESPUESTA
A LA PETICIÓN
HTTP
CLIENT SIDE

Importancia de las APIs en el desarrollo de software.

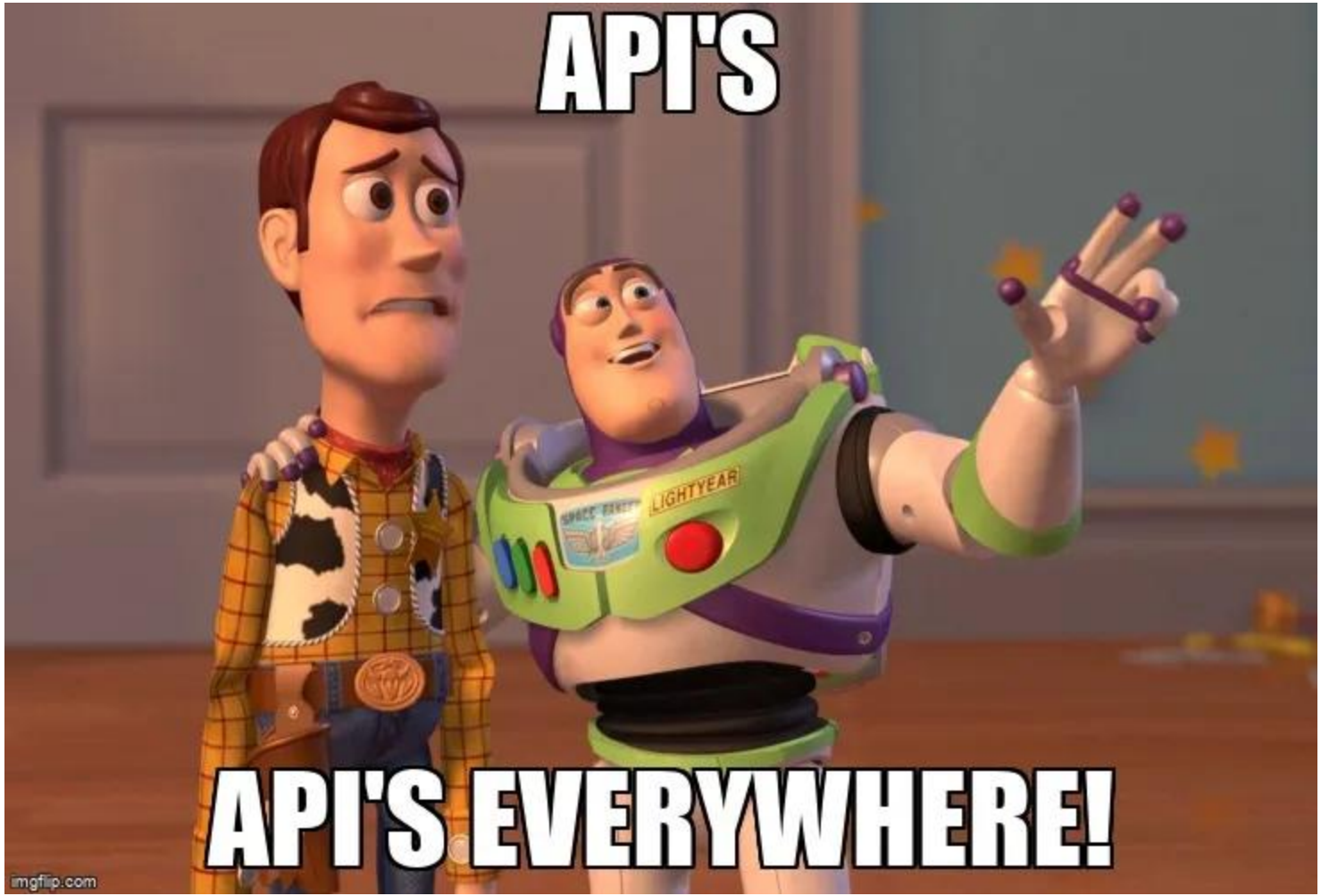
Una de las principales funciones de las API es poder facilitarle el trabajo a los desarrolladores y ahorrarles tiempo y dinero. Por ejemplo, si estamos creando una aplicación que es una tienda online, no necesitaremos crear desde cero un sistema de pagos u otro para verificar si en nuestro proveedor hay stock disponible de un producto. Podremos utilizar la API de un servicio de pago ya existente, por ejemplo MercadoPago o PayPal, y pedirle al distribuidor una API que te permita saber el stock que ellos tienen.

Con ello, no será necesario tener que reinventar la rueda con cada servicio que se crea, ya que podremos utilizar piezas o funciones que otros ya han creado. Imaginemos que cada tienda online tuviera que tener su propio sistema de pago, para los usuarios normales es mucho más cómodo poder hacerlo con los principales servicios que casi todos utilizan.

Ejemplos de uso cotidiano de APIs

Por ejemplo, si queremos crear una aplicación que se conecte a las publicaciones que hay en X (Twitter), tendremos que conectar la aplicación al servicio mediante la API que Twitter tiene disponible para los desarrolladores. Y si queremos que esta aplicación se comunique con la barra de notificaciones de un sistema operativo, entonces también necesitará otra API del sistema operativo.

Otro ejemplo: cuando vas a comprar una entrada a través de la web de una sala de cine. Cuando pones la información de tu tarjeta, la web utiliza una API para enviarle esa información de forma remota a otro programa que verifica si tus datos son correctos o es una tarjeta inventada. Una vez se verifica, este programa remoto le dice a la web que todo está en orden, y esta ya te emite tus entradas.


API'S
LIGHTYEAR
API'S EVERYWHERE!
imgflip.com

Tipos de API

API de SOAP: Estas API utilizan el protocolo simple de acceso a objetos. El cliente y el servidor intercambian mensajes mediante XML. Se trata de una API poco flexible que era más popular en el pasado.

API de RPC: Estas API se denominan llamadas a procedimientos remotos. El cliente completa una función (o procedimiento) en el servidor, y el servidor devuelve el resultado al cliente.

API de WebSocket: La API de WebSocket es otro desarrollo moderno de la API web que utiliza objetos JSON para transmitir datos. La API de WebSocket admite la comunicación bidireccional entre las aplicaciones cliente y el servidor. El servidor puede enviar mensajes de devolución de llamada a los clientes conectados, por lo que es más eficiente que la API de REST.

API de REST: Estas son las API más populares y flexibles que se encuentran en la web actualmente. El cliente envía las solicitudes al servidor como datos. El servidor utiliza esta entrada del cliente para iniciar funciones internas y devuelve los datos de salida al cliente. Veamos las API de REST con más detalle a continuación.

Generalidades del protocolo HTTP

HTTP, de sus siglas en inglés: "Hypertext Transfer Protocol", es el nombre de un protocolo el cual nos permite realizar una petición de datos y recursos, como pueden ser documentos HTML. Es la base de cualquier intercambio de datos en la Web, y un protocolo de estructura cliente-servidor, esto quiere decir que una petición de datos es iniciada por el elemento que recibirá los datos (el cliente), normalmente un navegador Web. Así, una página web completa resulta de la unión de distintos sub-documentos recibidos, como, por ejemplo: un documento que especifique el estilo de maquetación de la página web (CSS), el texto, las imágenes, vídeos, scripts, etc...

HTTP

HTTP es sencillo

Incluso con el incremento de complejidad, que se produjo en el desarrollo de la versión del protocolo HTTP/2, en la que se encapsularon los mensajes, HTTP esta pensado y desarrollado para ser leído y fácilmente interpretado por las personas, haciendo de esta manera más facil la depuración de errores, y reduciendo la curva de aprendizaje para las personas que empiezan a trabajar con él.

HTTP es extensible

Presentadas en la versión HTTP/1.0, las cabeceras de HTTP han hecho que este protocolo sea fácil de ampliar y de experimentar con él. Funcionalidades nuevas pueden desarrollarse, sin más que un cliente y su servidor, comprendan la misma semántica sobre las cabeceras de HTTP.

HTTP

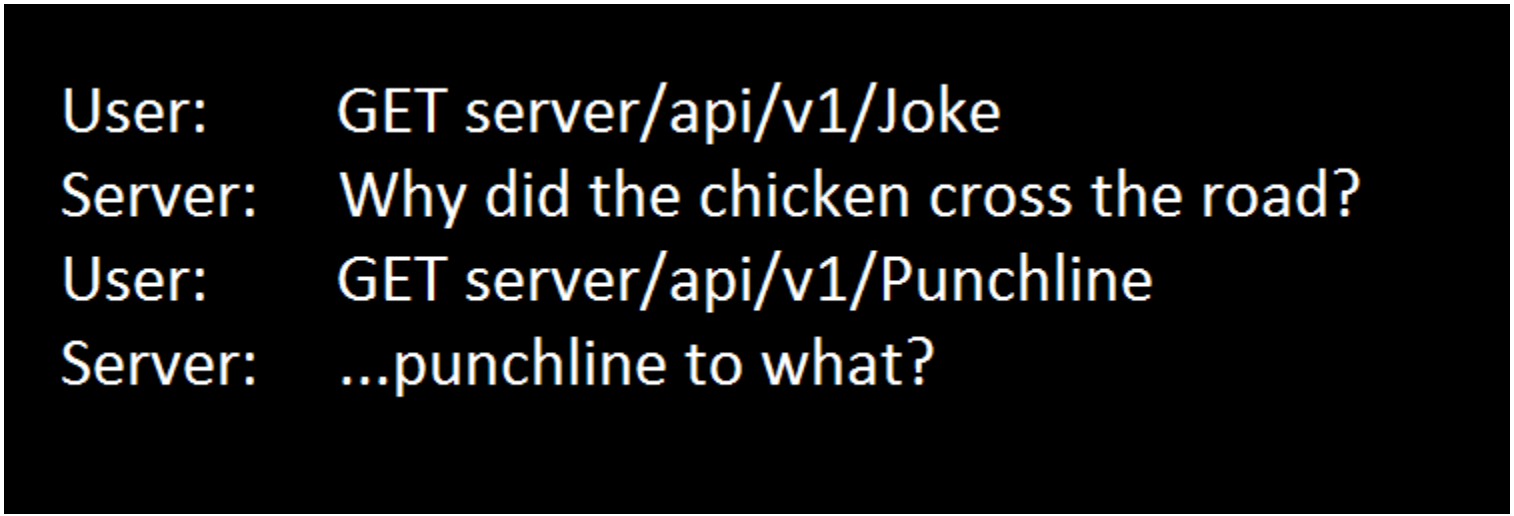
HTTP es un protocolo con sesiones, pero **sin estados**.

¿Que implica que HTTP sea un protocolo sin estado?

Significa que no guarda ningún dato entre dos peticiones en la misma sesión. Esto crea problemáticas, en caso de que los usuarios requieran interactuar con determinadas páginas Web de forma ordenada y coherente, por ejemplo, para el uso de "cestos de compra" en páginas que utilizan en comercio electrónico. Pero, mientras HTTP ciertamente es un protocolo sin estado, el uso de HTTP cookies, sí permite guardar datos con respecto a la sesión de comunicación. Usando la capacidad de ampliación del protocolo HTTP, las cookies permiten crear un contexto común para cada sesión de comunicación.

HTTP

HTTP es un protocolo con sesiones, pero **sin estados**.



¿Qué es un servidor Web?

Un servidor web se define como un software que hace uso del HTTP (Hypertext Transfer Protocol) para almacenar los archivos que forman los sitios web y mostrarlos a los usuarios cuando estos lo solicitan.

Por lo tanto, cualquier tipo de sitio web requiere de un servidor de este tipo. Algunas empresas, generalmente de gran tamaño, cuentan con un servidor web propio. No obstante, en la gran mayoría de ocasiones, tanto particulares como compañías, optan por contratar este servicio a un proveedor de alojamiento web como Amazon, Hostinger, etc.

Servidor Web

El protocolo utilizado para la transmisión de archivos es HTTP o HTTPS (ofrece un nivel extra de seguridad gracias al cifrado). Este protocolo se basa a su vez en los protocolos de red IP y TCP. Así, un servidor web puede mostrar el contenido de un sitio web de forma simultánea a varios navegadores web.

La cantidad de solicitudes, además de la velocidad con la que pueden ser procesadas, depende de varios factores: hardware, número de solicitudes realizadas por los usuarios al mismo tiempo, etc.

Así, es importante escoger el servidor web que mejor se adapte a las necesidades de cada persona o empresa. El principal objetivo es evitar sobrecargas en el servidor que puedan ralentizar la velocidad de carga del sitio web. HTTP engloba una serie de métodos de petición que permiten indicar la acción a realizar para un determinado recurso.

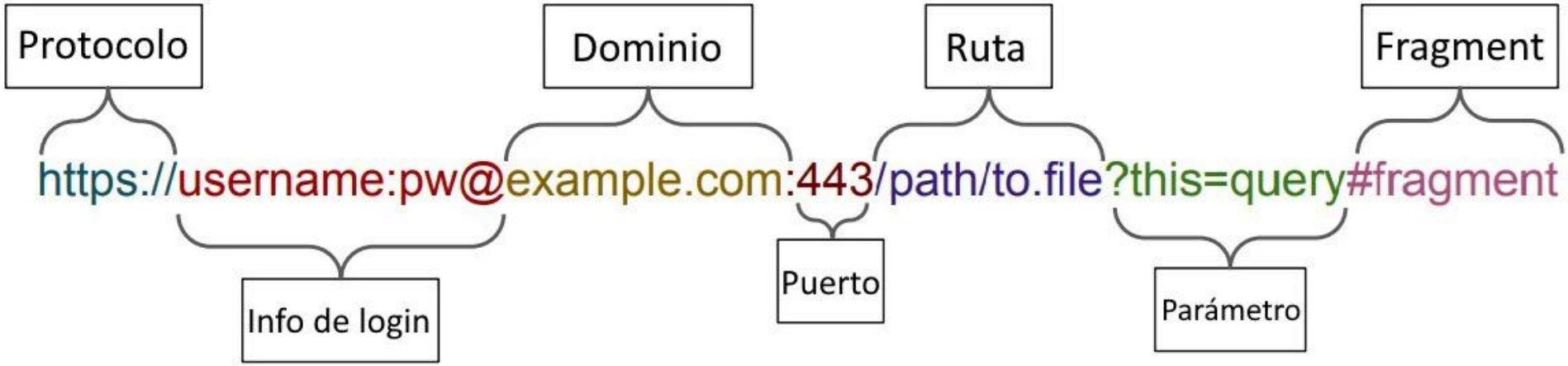
¿Qué es una URL?

Junto con el Hipertexto y HTTP, las URL son uno de los conceptos claves de la Web. Es el mecanismo usado por los navegadores para obtener cualquier recurso publicado en la web.

URL significa Uniform Resource Locator (Localizador de Recursos Uniforme). Una URL no es más que una dirección que es dada a un recurso único en la Web. En teoría, cada URL válida apunta a un único recurso. Dichos recursos pueden ser páginas HTML, documentos CSS, imágenes, etc. En la práctica, hay algunas excepciones, siendo la más común una URL apuntando a un recurso que ya no existe o que ha sido movido. Como el recurso representado por la URL y la URL en sí son manejadas por el servidor Web, depende del dueño del servidor web manejar ese recurso y su URL asociada adecuadamente.

Ejemplo: www.google.com

Partes de una URL



Protocolo: Indica al navegador cómo debe conectarse al servidor. No es lo mismo "http://" que su versión segura ("https://") —aunque ambas se conectan a páginas web—, o que "mailto://" (para direcciones de e-mail), "ftp://", etc. Uso obligatorio, aunque los navegadores ya no siempre lo muestran.
Información de login: Podemos indicar sólo el nombre de usuario o incluir también la contraseña. Uso opcional y cada vez menos frecuente.

Dominio: Es el eje en torno al cual gira una URL, le dice al navegador a qué máquina de Internet conectarse (con la ayuda de las DNS). Uso obligatorio. En algunos casos puede aparecer dividido por un punto, indicando un 'subdominio' (primercurso.colegio.es).

Puerto: Cada protocolo tiene un puerto o puertos vinculados por defecto, pero en algunos casos podemos querer indicar al navegador que se conecte, por ejemplo, a un servidor web secundario instalado en la misma máquina que otro: en esos casos, lo habitual es recurrir a especificar un puerto distinto. Uso opcional.

Ruta: Cuando no queremos acceder a la página principal, sino a un subdirectorio de sitio web. Uso opcional, pero muy frecuente.

Parámetro: También conocido como 'query', permite pasar valores para ser utilizados por el servidor.

Fragment: Para indicar al navegador que no muestre la página deseada desde el principio, sino que se localice en un punto concreto de dicha página.

Tipos de peticiones

Peticiones GET: El método GET se utiliza cuando se solicita la representación de un recurso en concreto. Está dirigido a la recuperación de datos. Así, se trata de recuperar cualquier tipo de información identificada por el Request-URI. De esta manera, si el Request-URI hace referencia a un proceso de producción de datos, son los datos generados los que se devuelven a modo de respuesta, no el texto fuente del proceso.

Peticiones POST: Este es el segundo tipo de petición HTTP más utilizado a nivel global. Los datos que se envían al servidor se engloban el cuerpo de la propia petición con las cabeceras HTTP correspondientes a la misma. En la gran mayoría de ocasiones guarda relación con los formularios web, en los que los datos son cifrados para ser enviados al servidor de una forma totalmente segura.

¿Que es HTTP Request y Response?

¿Que es HTTP Request y Response?

Un sitio web que empieza con la URL http:// es entrado en un navegador web de la computadora del cliente. El navegador puede ser Chrome, Firefox, Brave, Edge, Opera, o cualquier otro, no importa cuál.

El navegador envía un **request** (solicitud) al servidor web que está hospedando el website.

El servidor web regresa una respuesta como un página de HTML, o algún otro formato de documento al navegador (puede ser un mp4, mp3, pdf, doc, entre otros soportados por el navegador)

El navegador despliega **el response** (respuesta) del servidor al usuario. Por supuesto esto dependerá de los formatos que soporte el navegador.

HTTP Request Structure from Client

Un simple mensaje de request de un cliente tiene los siguientes componentes:

Una línea de request para obtener el recurso, por ejemplo un request con el método GET /content/page1.html está requiriendo el recurso llamado /content/page1.html del servidor

Encabezados. Indican cosas como el lenguaje, codificación, tipo de datos (XML,JSON, etc). (Por ejemplo– Accept-Language: EN).

Un cuerpo del mensaje que es **opcional**. Entre aplicaciones esta es la parte más importante. Por ejemplo, yo puedo enviar un XML o un JSON a otra máquina, y el servidor web interpretará la información que yo le mando.

HTTP Response Structure from Web Server

Un simple response del servidor web contiene lo siguiente:

HTTP Status Code (Por ejemplo HTTP/1.1 301 Movido Permanentemente, significa que el recurso requerido fue movido permanentemente y redirigido a otro recurso).

Encabezados. Igual que en el request, describen el contenido del response (Ejemplo – Content-Type: html)

Un cuerpo de mensaje, que es **opcional**. Cuando trabajamos con aplicaciones, aquí puede venir la respuesta en XML u otro formato. Cuando es una página del navegador, contiene el código HTML que forma nuestra página.

Códigos de estado de respuesta HTTP

Respuestas informativas (100–199),

Respuestas satisfactorias (200–299),

Redirecciones (300–399),

Errores de los clientes (400–499),

y errores de los servidores (500–599).

Ejemplo: Respuesta 200 OK

La solicitud ha tenido éxito. El significado de un éxito varía dependiendo del método HTTP

2XX

3XX

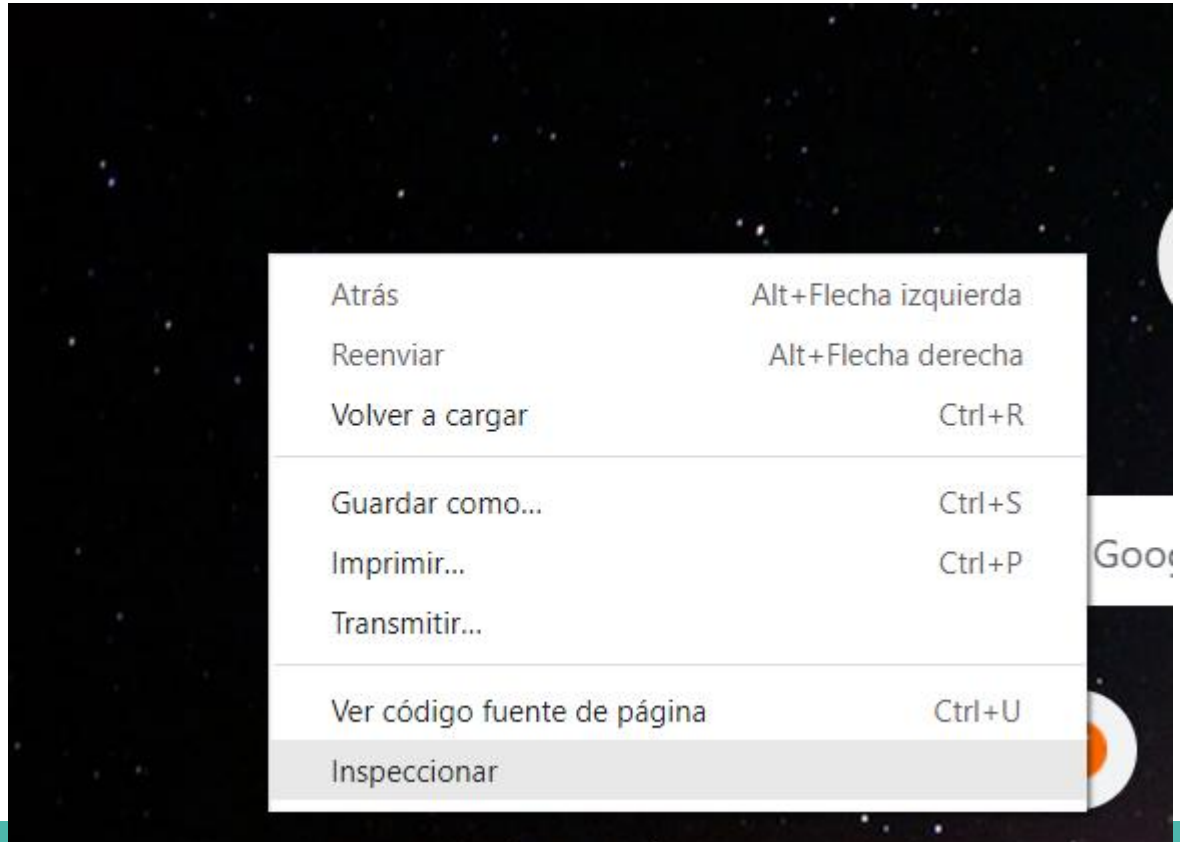
4XX

5XX

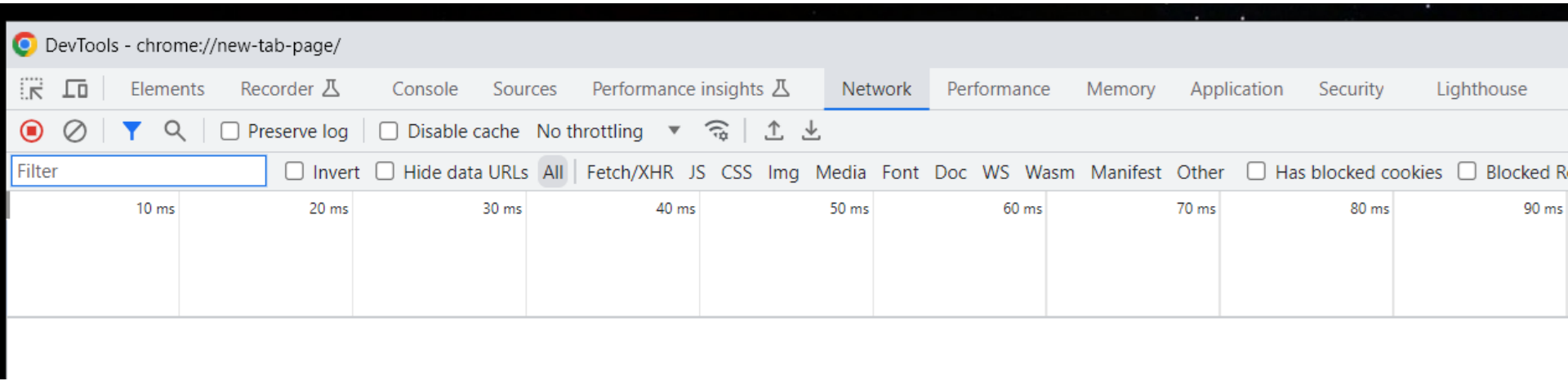
500

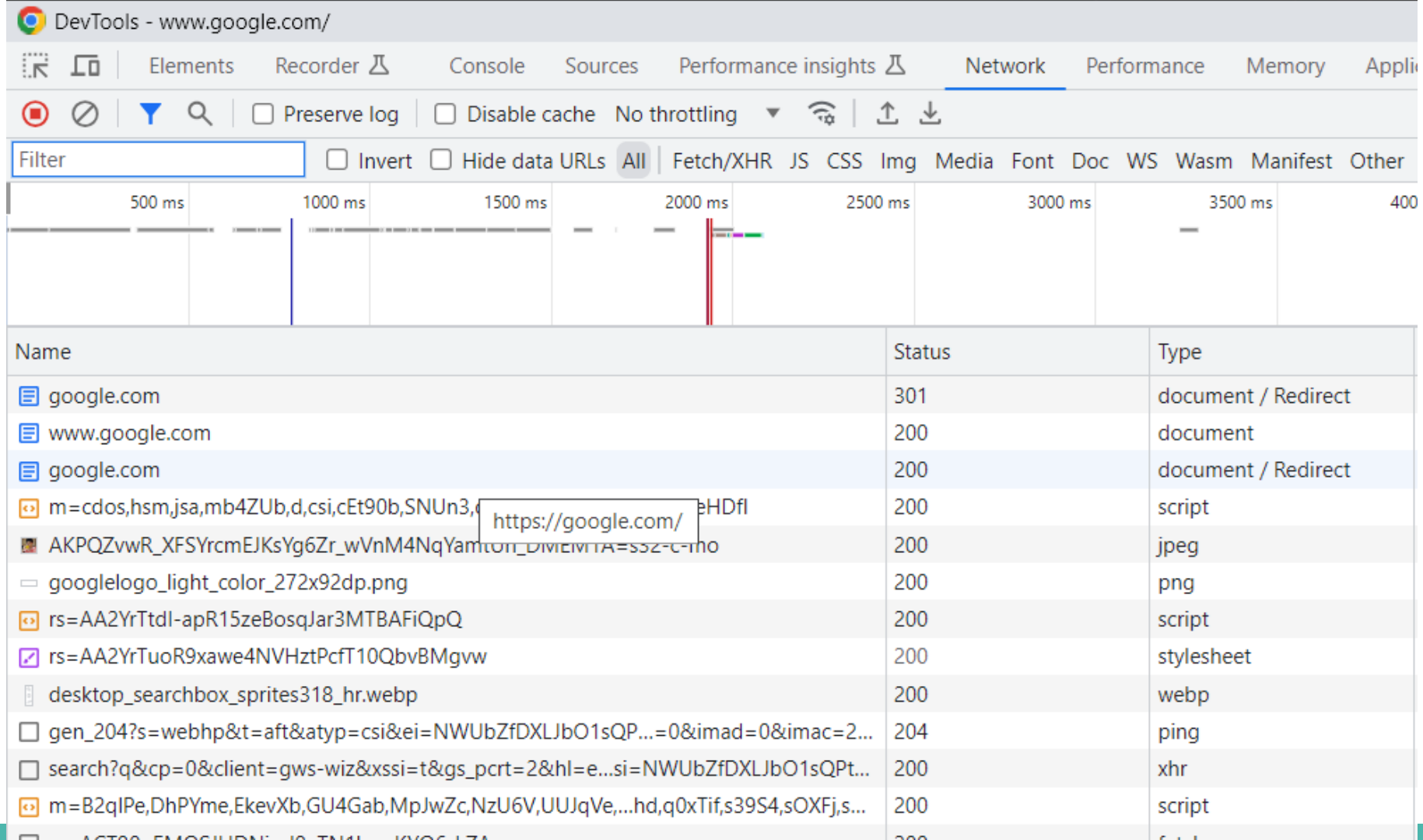
Internal Server Error

¿Como puedes ver las request? (chrome)



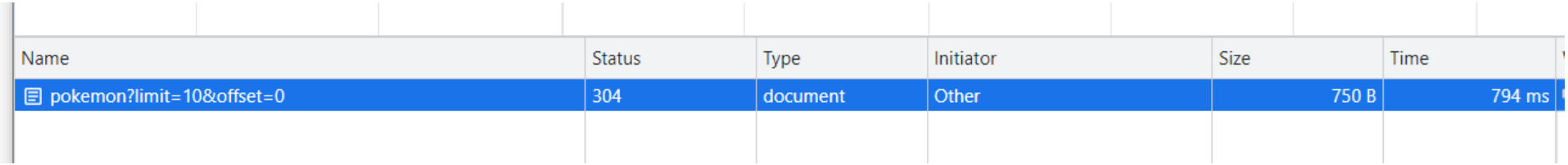
DevTools de chrome




DevTools - www.google.com/
Elements
Recorder
Console
Sources
Performance insights
Network
Performance
Memory
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
All
Fetch/XHR
JS
CSS
Img
Media
Font
Doc
WS
Wasm
Manifest
Other
500 ms
1000 ms
1500 ms
2000 ms
2500 ms
3000 ms
3500 ms
Name
Status
Type
google.com
301
document / Redirect
www.google.com
200
document
google.com
200
document / Redirect
m=cdos,hsm,jsa,mb4ZUb,d,csi,cEt90b,SNUn3,
HDfl
200
script
https://google.com/
AKPQZvwR_XFSYrcmEJKsYg6Zr_wVnM4NqYam
200
jpeg
googlelogo_light_color_272x92dp.png
200
png
rs=AA2YrTtdI-apR15zeBosqJar3MTBAFiQpQ
200
script
rs=AA2YrTuoR9xawe4NVHztPcfT10QbvBMgvw
200
stylesheet
desktop_searchbox_sprites318_hr.webp
200
webp
gen_204?s=webhp&t=aft&atyp=csi&ei=NWUbZfDXLJbO1sQP...=0&imad=0&imac=2...
204
ping
search?q&cp=0&client=gws-wiz&xssi=t&gs_pcrt=2&hl=e...si=NWUbZfDXLJbO1sQPt...
200
xhr
m=B2qlPe,DhPYme,EkevXb,GU4Gab,MpJwZc,NzU6V,UUJqVe,...hd,q0xTif,s39S4,sOXFj,s...
200
script

Ejemplo con api publica https://pokeapi.co/

https://pokeapi.co/api/v2/pokemon?limit=3&offset=0



Name	Status	Type	Initiator	Size	Time
pokemon?limit=10&offset=0	304	document	Other	750 B	794 ms


```
{"count":1292,"next":"https://pokeapi.co/api/v2/pokemon?offset=3&limit=3","previous":null,"results":[{"name":"bulbasaur","url":"https://pokeapi.co/api/v2/pokemon/1/"},{"name":"ivysaur","url":"https://pokeapi.co/api/v2/pokemon/2/"},{"name":"venusaur","url":"https://pokeapi.co/api/v2/pokemon/3/"}]}
```