

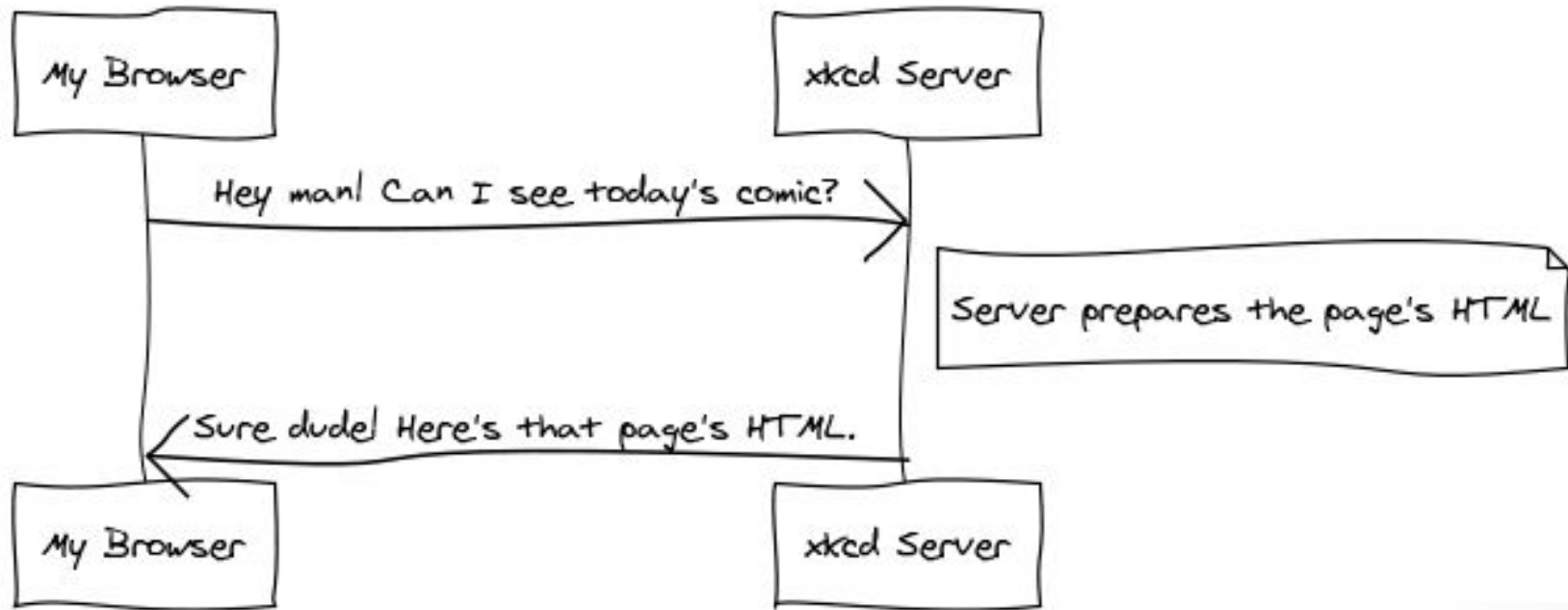
Arquitectura cliente-Servidor

LAS BASES DE INTERNET

DEV.F
DESARROLLAMOS(PERSONAS);

dev





Hace peticiones, Recibe respuestas.

(Navegador)

Cliente

DEV.F

dev

El cliente pide los recursos al servidor, recibe la respuesta del servidor y se la muestra al usuario de manera adecuada.



Recibe peticiones, envía respuestas.

Servidor

Diferentes tipos de servidores.

- Servidores de base de datos.
- Servidores de correo electrónicos.
- Servidores de imágenes.
- Servidores WEB.



Servidor



Servidor de Correo



Servidor FTP



Servidor Web



Servidor Proxy



Servidor Base de Datos



Servidor Audio/Video



Servidor Chat



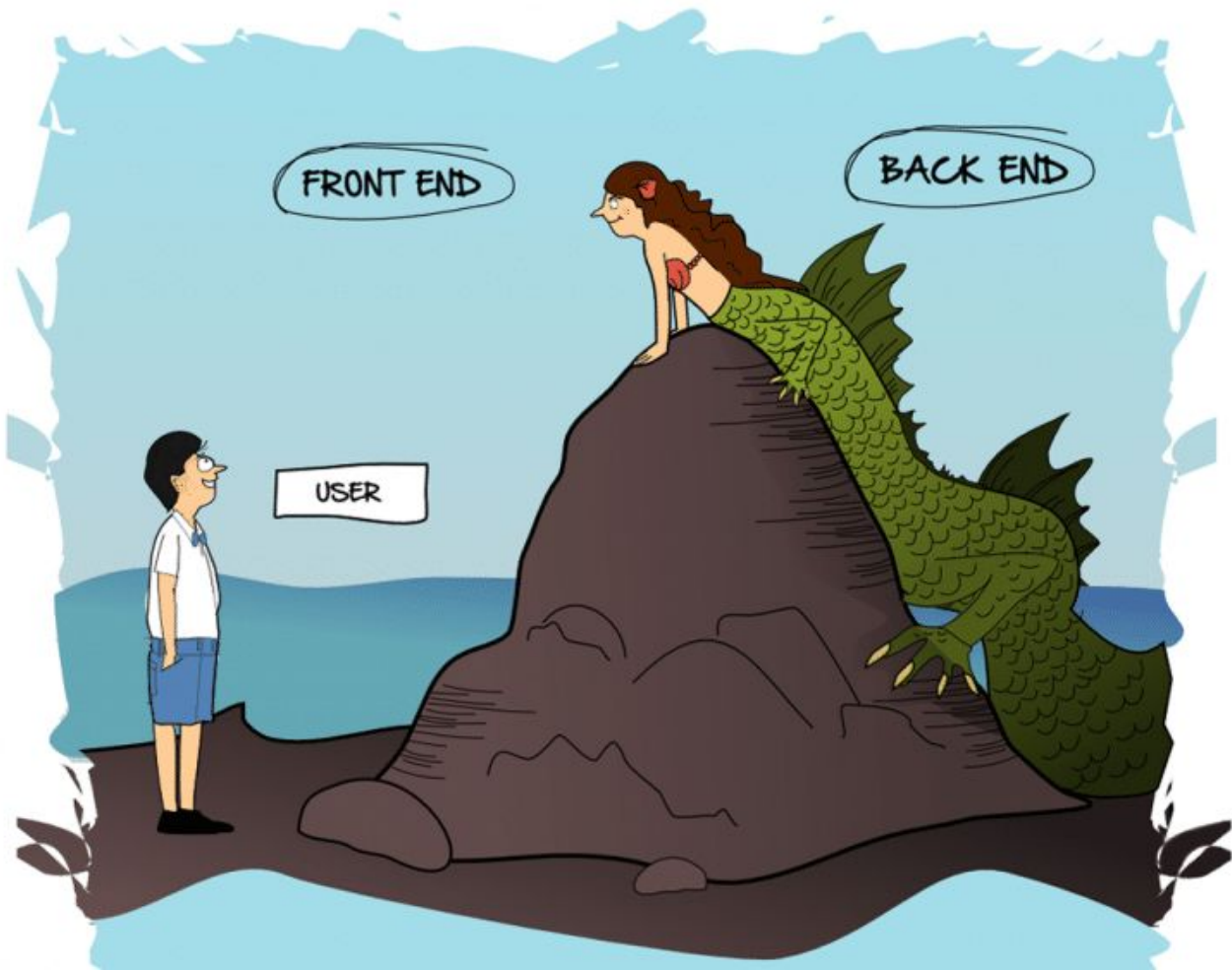
Servidor Groupware



Cluster de Servidores





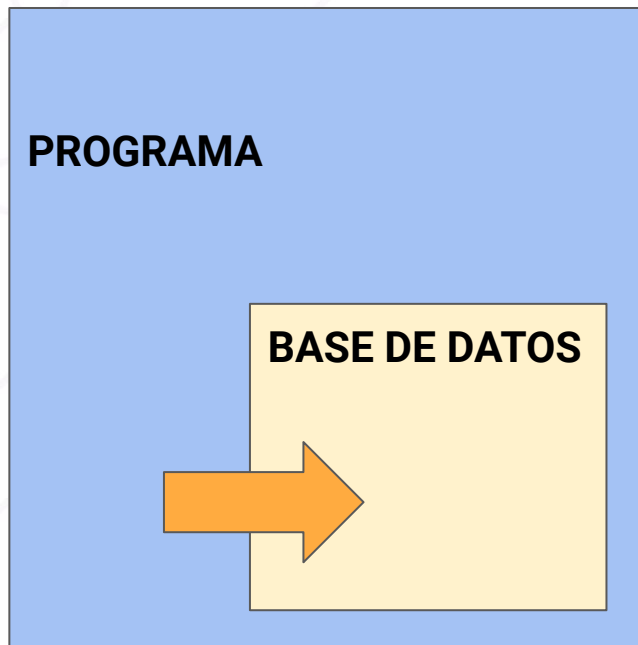


Enviar datos al servidor

DEV.F
DESARROLLAMOS(PERSONAS);

dev

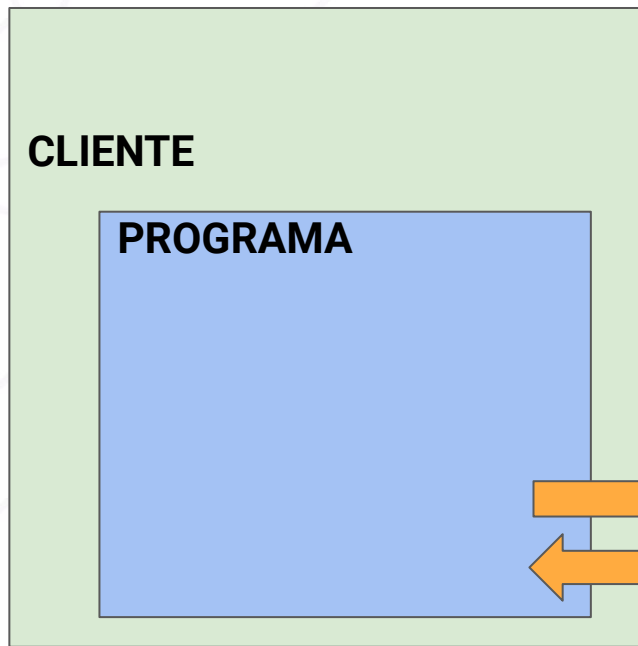
Evolución de Acceso a la Información (1)



En los inicios, la aplicación convivía muy de cerca con su base de datos y ejecutándose en un mismo equipo.

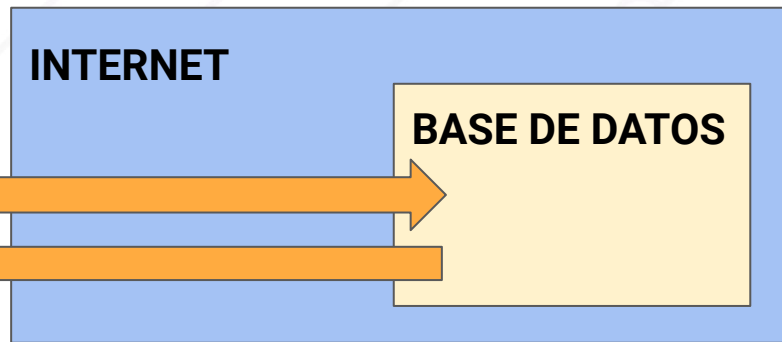


Evolución de Acceso a la Información (2)



Los programas cada día comenzaban a necesitar cada vez más reglas de negocios.

Las bases de datos son primariamente para guardar y recuperar información, pero no para aplicar reglas de negocio.



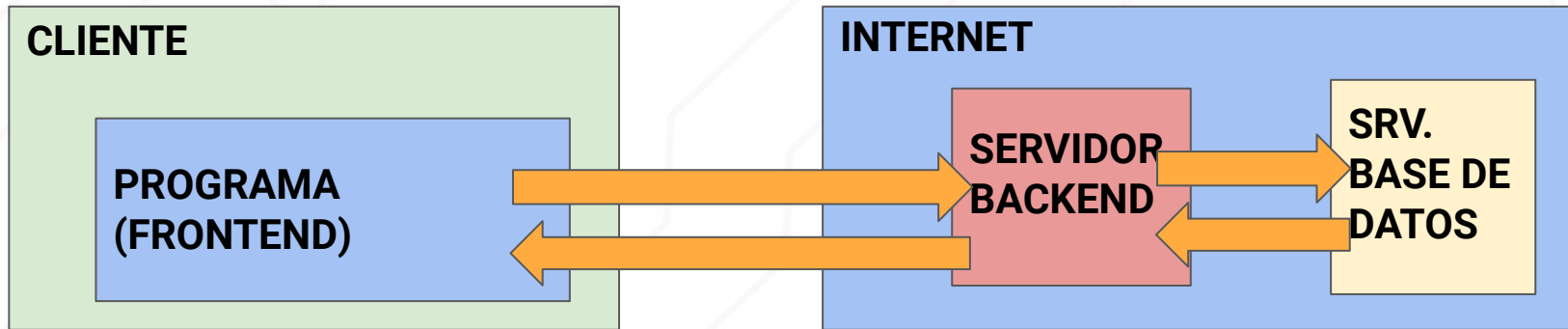
Sin embargo es poco seguro que el Cliente a través de internet acceda directamente a la base de datos.

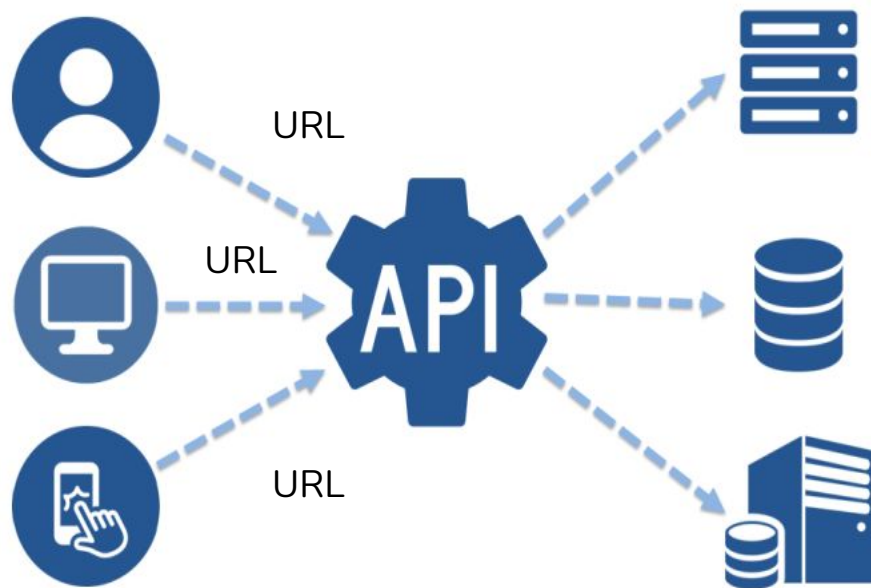
Evolución de Acceso a la Información (3)

Una solución fue escribir un programa **Backend** que acceda a la base de datos.

Normalmente **las bases de datos y el backend están servidores independientes**, y se configura la base de datos para que solo reciba instrucciones del backend (por ip).

De esta forma ahora el Backend es el programa encargado de validar la lógica de negocios y el acceso a la información de la base de datos.





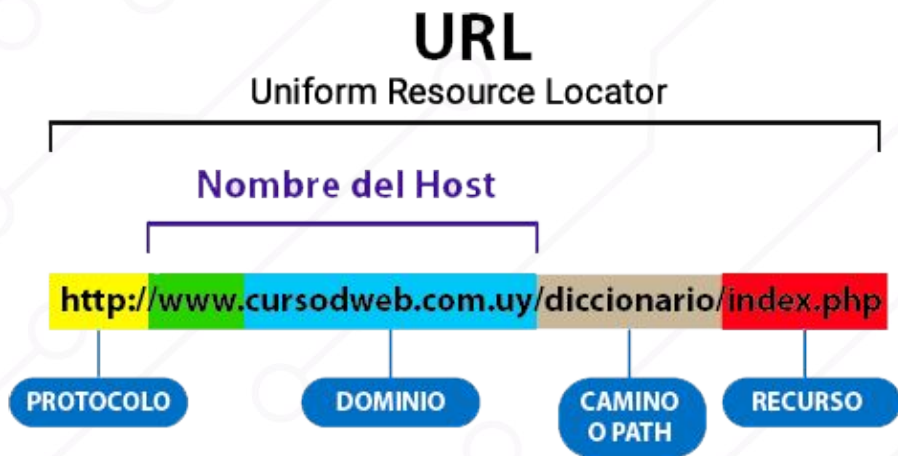
API

(Application Programming Interface)

Una interfaz de programación de aplicaciones (API) es un conjunto de herramientas, definiciones y protocolos que se utiliza para integrar los servicios y el software de aplicaciones.

Es lo que permite que sus productos y servicios se comuniquen con otros, sin tener que diseñar permanentemente una infraestructura de conectividad nueva.

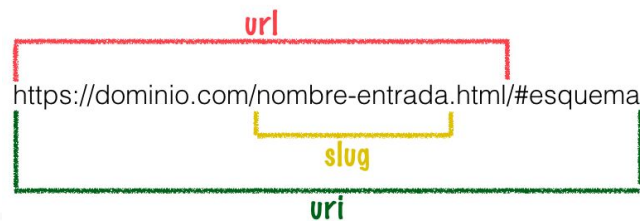
URL



- Protocolo:** http, https, ftp, file...
- Subdominio**
- Dominio:** .com, .org, .edu... ccTLD, .com.uy, .com.ar, .es
- Path o camino al directorio del curso**
- Nombre de archivo:** .php, .htm, .html, .asp, .jsp...

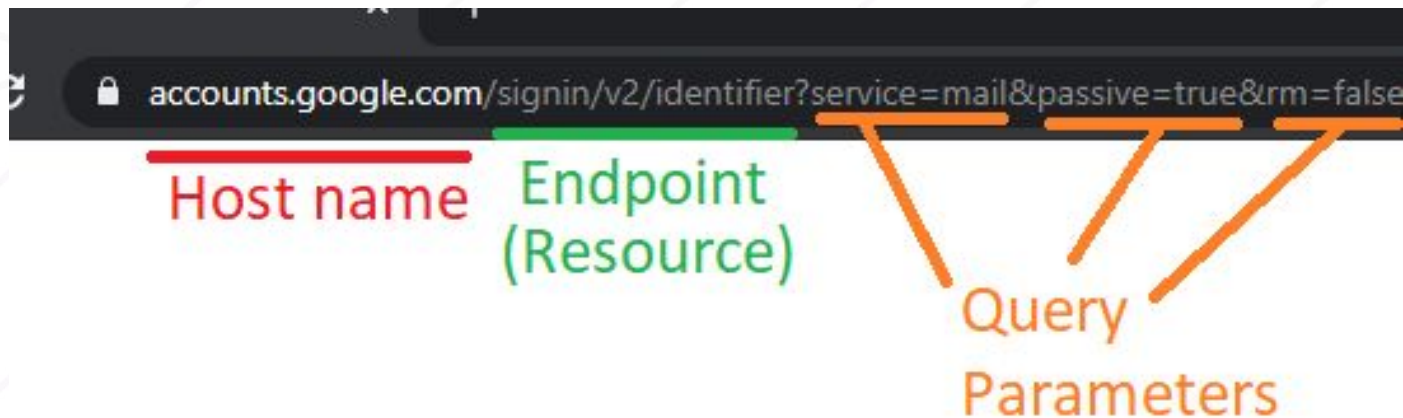
Las URL son términos para el frontend. Si lo que devuelve es información y es para el Backend nos referimos a estos como Endpoints.

Endpoints: URI que hacen referencia a una API.



Query Parameters

Un query param es la clave valor (ejemplo: service=mail) que vemos al final de la URL, y como regla, siempre deberán estar después del símbolo de interrogación. Además, una URL puede tener N query params, separadas por &, cómo el siguiente ejemplo:

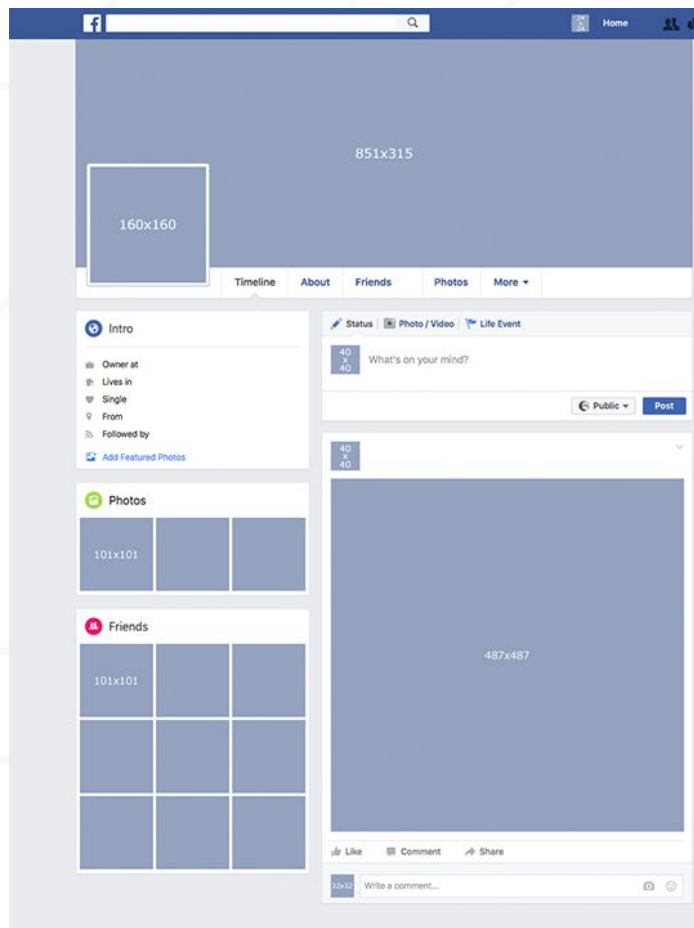


JSON

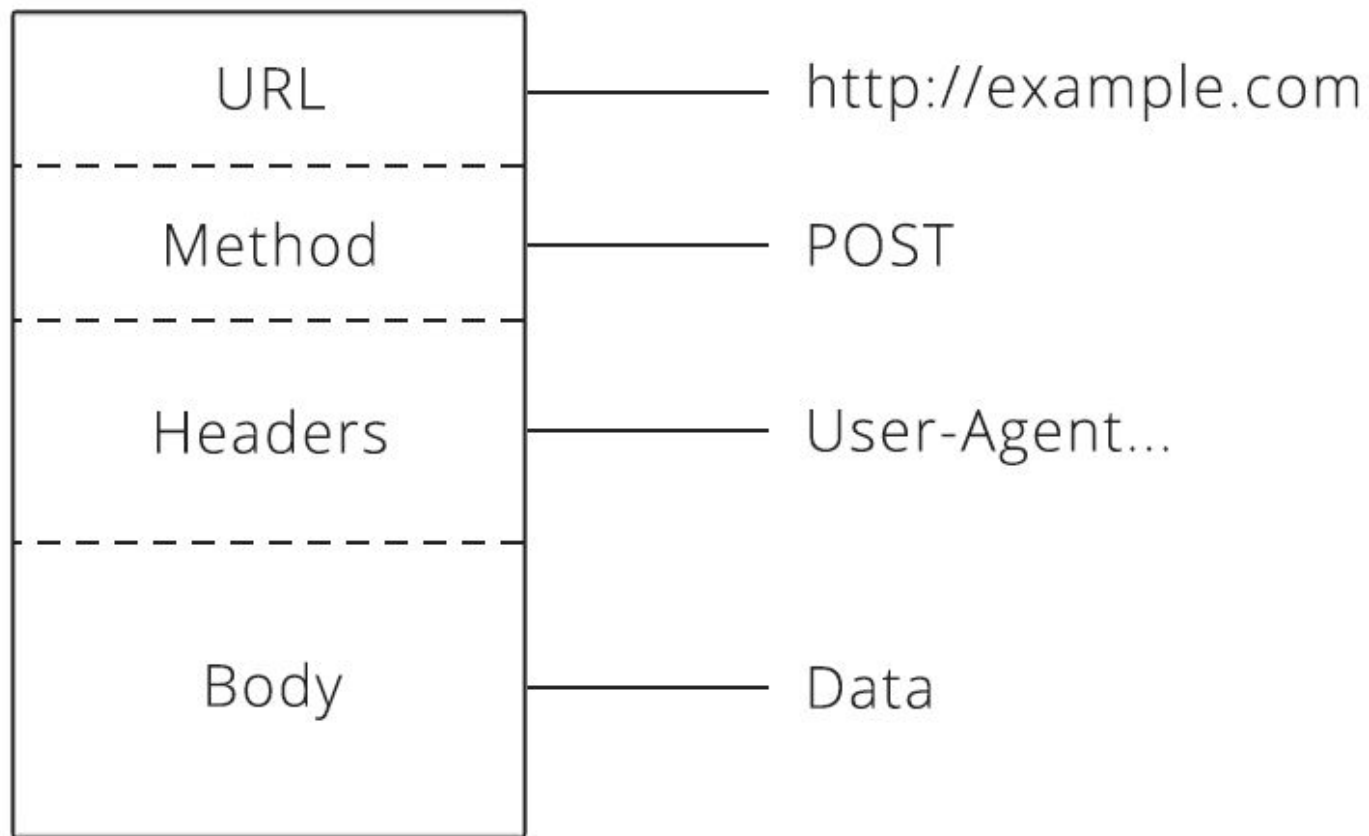
```
1 {  
2   "name": "Christopher Co",  
3   "age": 29,  
4   "level": 7,  
5   "gender": "M",  
6   "status": "good"  
7 }
```

JSON es hoy en día el método estándar de comunicación para enviar/recibir datos del servidor. Tiene un formato similar a un objeto en JavaScript.

JSON = JavaScript Object Notation



```
{
  "_id": {
    "$oid": "5c3fd50c680caf5787cd11bb"
  },
  "is_important": false,
  "applied_codes": [],
  "is_active": true,
  "first_name": "Raul",
  "last_name": "Prueba Pruebita",
  "phone": "7821076070",
  "email": "dagorik@gmail.com",
  "approaches": [{}],
  "programs": [],
  "createdAt": {
    "$date": "2019-01-28T23:28:11.470Z"
  },
  "updatedAt": {
    "$date": "2019-01-28T23:28:11.470Z"
  },
  "__v": 1
}
```



Request

GET

Pedir datos al servidor

POST

Mandar datos al servidor

PUT

Actualizar datos en el servidor

PATCH

Actualizar datos en el servidor

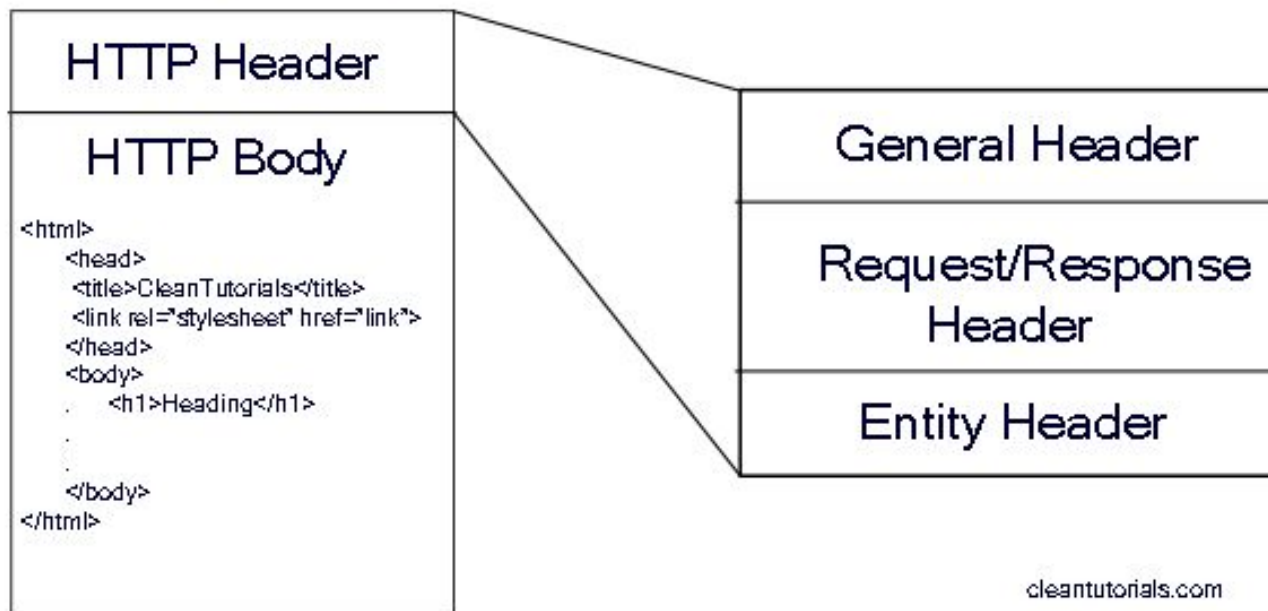
DELETE

Eliminar datos del servidor

Headers

Son meta-información relevante para que el cliente o el servidor conozca más acerca de la petición.

HTTP Request/response



cleantutorials.com