



## **Clase 1**

### **WEB. Arquitectura, tecnologías y herramientas**



## **Arquitectura cliente-servidor**

- **Cliente:**
  - Consumidor de recursos externos
- **Servidor:**
  - Comparte recursos externos
- **Mayoría de servicios de internet:**
  - Email, Web, DNS, etc.
  - Son cliente-servidor

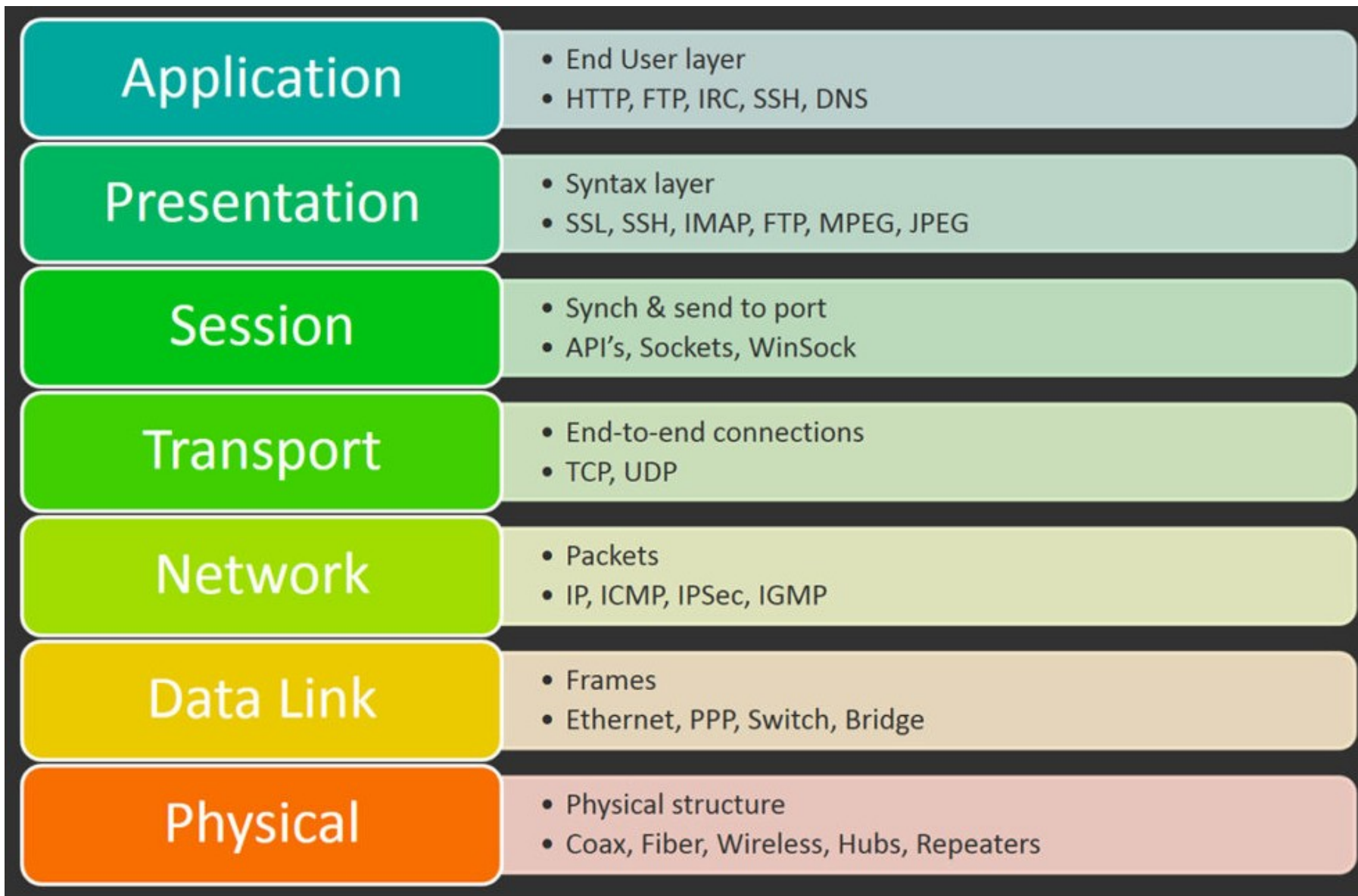


## **Arquitectura cliente-servidor**

- **Protocolo TCP/IP**
  - **Cliente/Servidor**
  - **Alguien escucha una comunicación**
  - **Alguien inicia una comunicación**
- **Protocolo HTTP**
  - **Sobre TCP/IP**

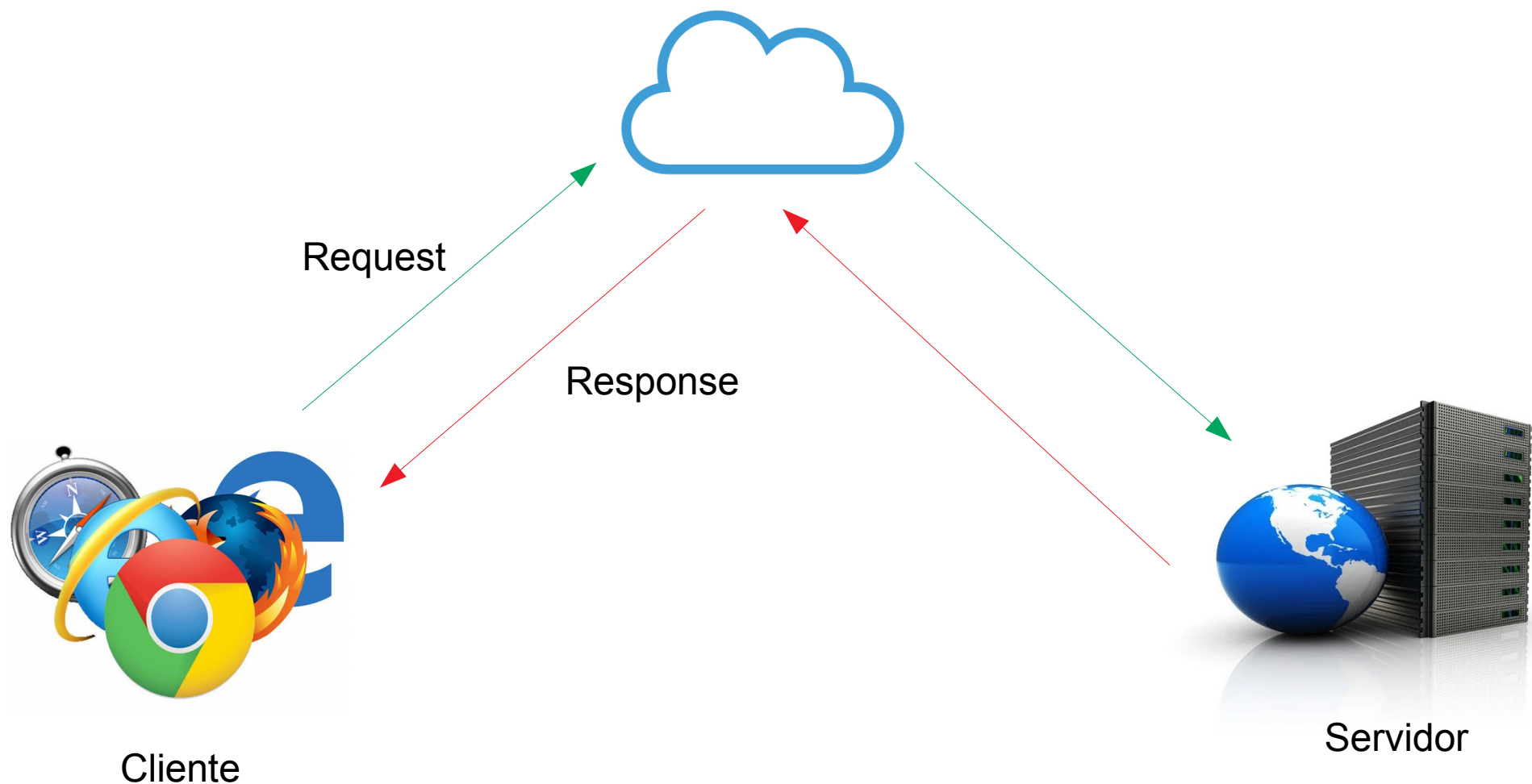


## Modelo de Capas (OSI)



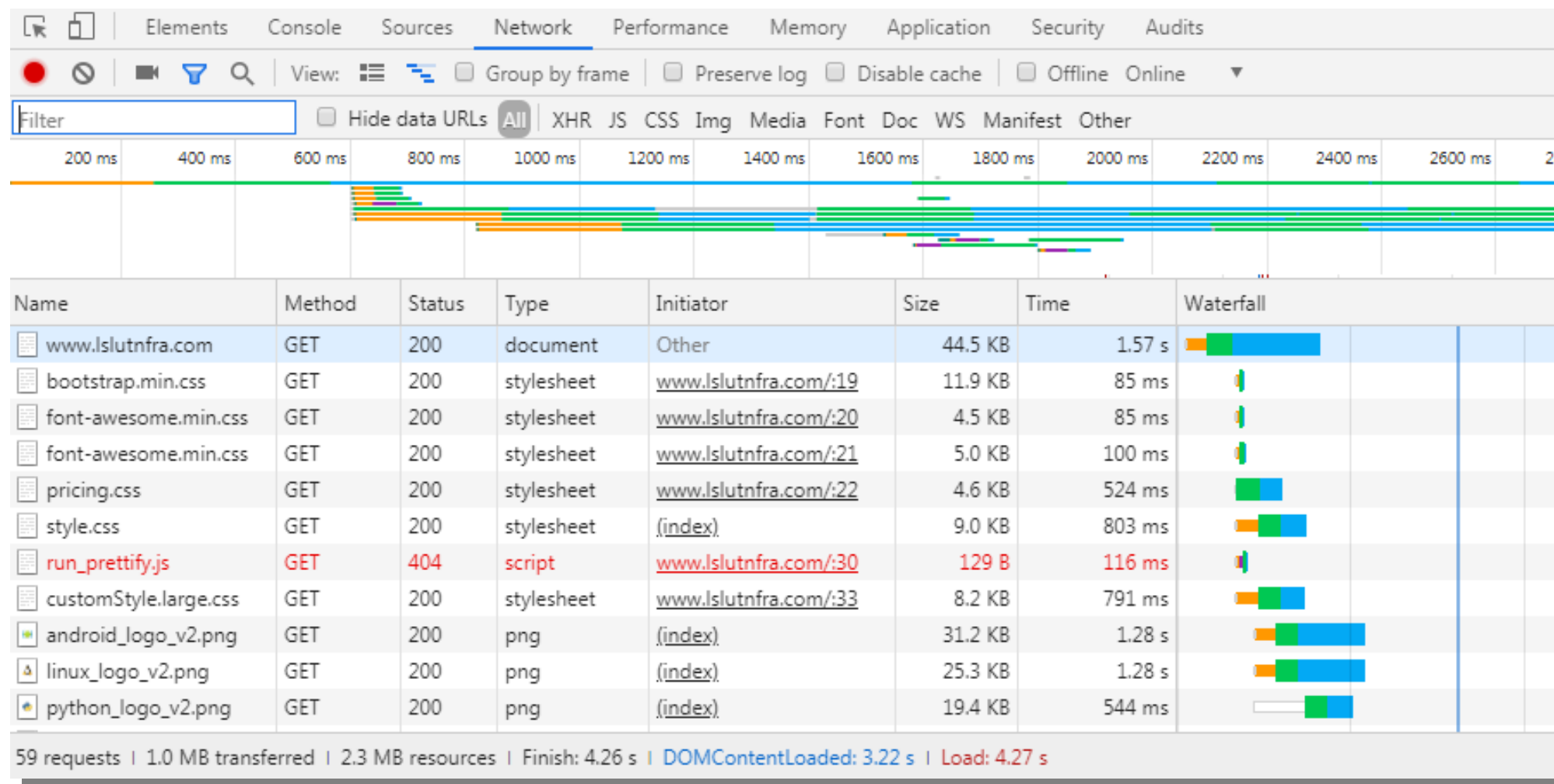


## Arquitectura cliente-servidor





## Arquitectura cliente-servidor







## Arquitectura cliente-servidor

- Protocolo HTTP: Hypertext Transfer Protocol
  - Protocolo de request-response
  - Sus recursos se identifican con URLs
  - Posee un *Header*
  - Se transmite texto plano



## **Arquitectura cliente-servidor**

- **Request:**
  - Dirección (URL).
  - Versión del protocolo.
  - Campos (Por Ej. "Accept-Language: en").
  - Cuerpo de mensaje (opcional).
  - Método (GET,POST,DELETE,etc.)
    - Definen la acción a realizar
    - Las acciones se definen en el server.





## **Arquitectura cliente-servidor**

- **Response:**
  - Dirección (URL).
  - Versión del protocolo.
  - Campo estado (200,404, etc.)
  - Campos (por Ej. "Content-Type: text/html")
  - Cuerpo de mensaje (opcional).



## Tipos de métodos

HTTP method ↕	RFC ↕	Request has Body ↕	Response has Body ↕
GET	<a href="#">RFC 7231</a>	Optional	Yes
HEAD	<a href="#">RFC 7231</a>	Optional	No
POST	<a href="#">RFC 7231</a>	Yes	Yes
PUT	<a href="#">RFC 7231</a>	Yes	Yes
DELETE	<a href="#">RFC 7231</a>	Optional	Yes
CONNECT	<a href="#">RFC 7231</a>	Optional	Yes
OPTIONS	<a href="#">RFC 7231</a>	Optional	Yes
TRACE	<a href="#">RFC 7231</a>	No	Yes
PATCH	<a href="#">RFC 5789</a>	Yes	Yes



## Arquitectura REST

URi: <http://www.misistema.com/usuarios/58>

- **GET:** Pido información del usuario con ID 58 al server
- **DELETE:** Borro al usuario con ID 58 del sistema
- **PUT:** Modifico el usuario con ID 58 en el sistema (los datos van en el body)



## Arquitectura REST

URi: <http://www.misistema.com/usuarios>

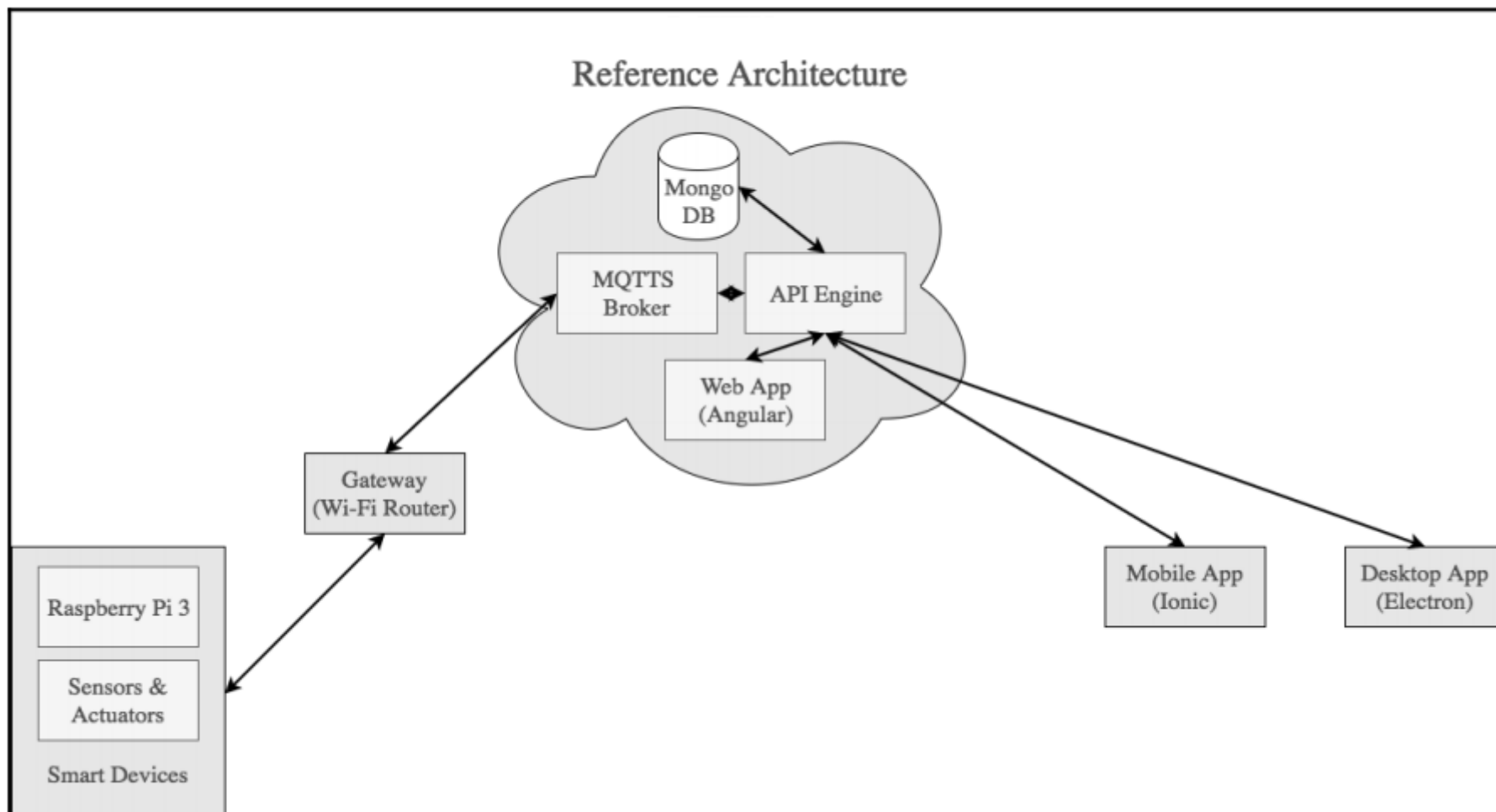
- **GET:** Pido la lista de usuarios al server
- **POST:** Creo un nuevo usuario en el sistema (los datos van en el body)

### Reglas para crear las direcciones

- Usar sustantivos y no verbos
  - Evitar "getUser", "newUser", etc.
- Indicar jerarquía con "/"
- No usar "/" al final
- Usar "-" para que sea más fácil de leer



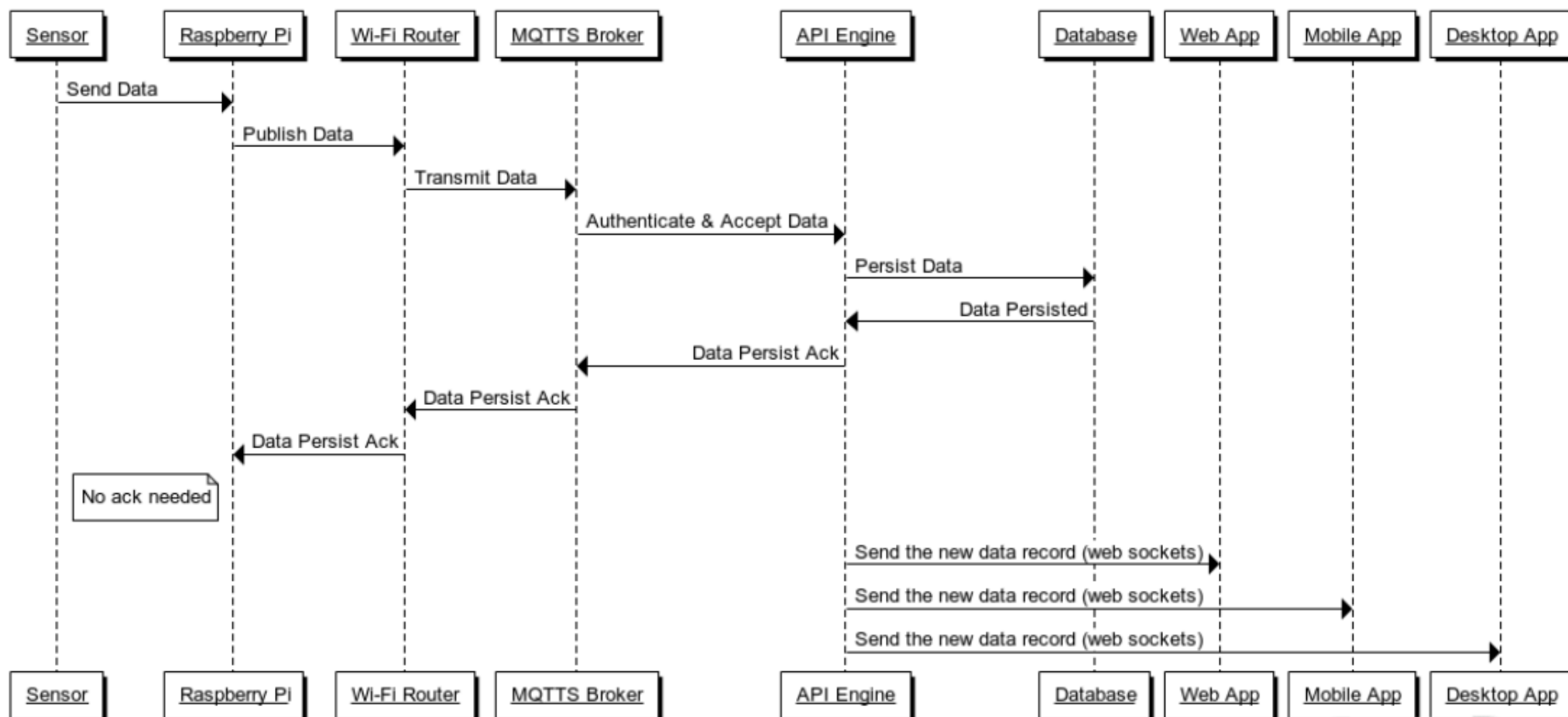
## Arquitectura cliente-servidor: Ejemplo





## Arquitectura cliente-servidor: Ejemplo

Data Flow from Sensor to Apps

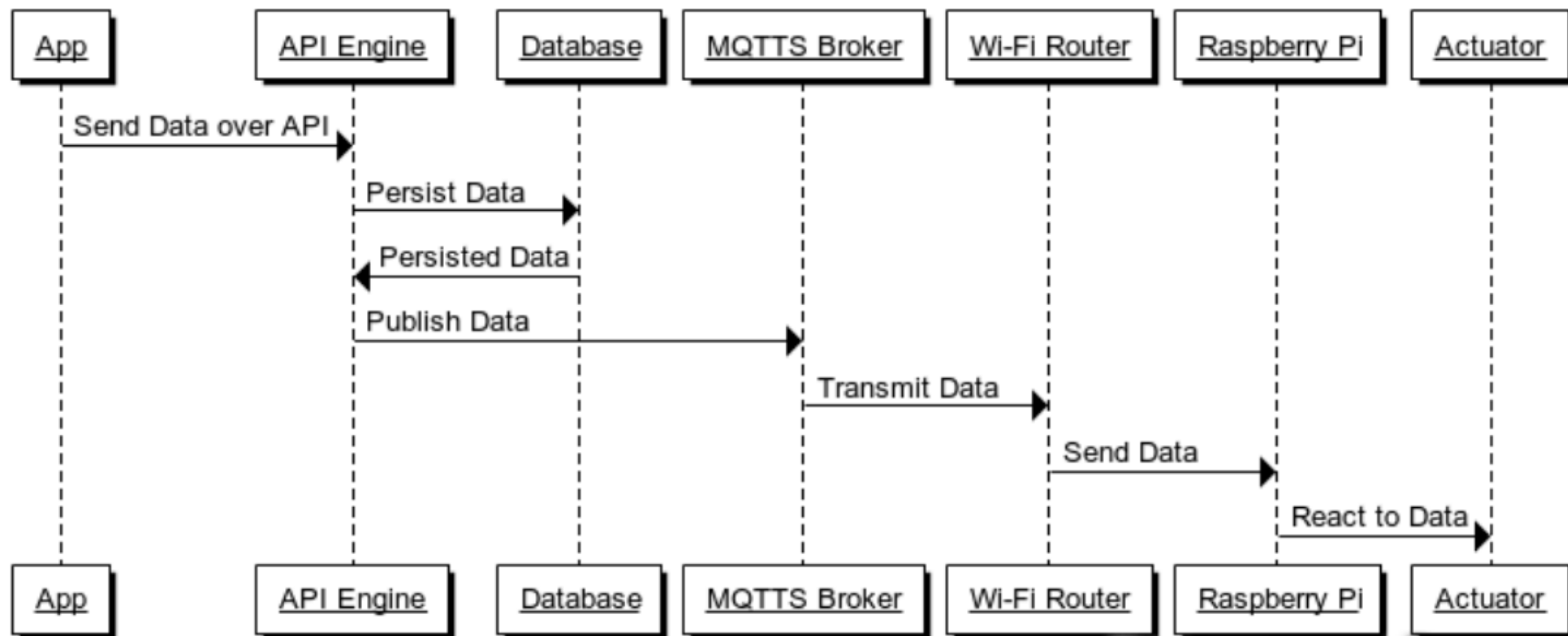






## Arquitectura cliente-servidor: Ejemplo

Data Flow from Apps to Actuators

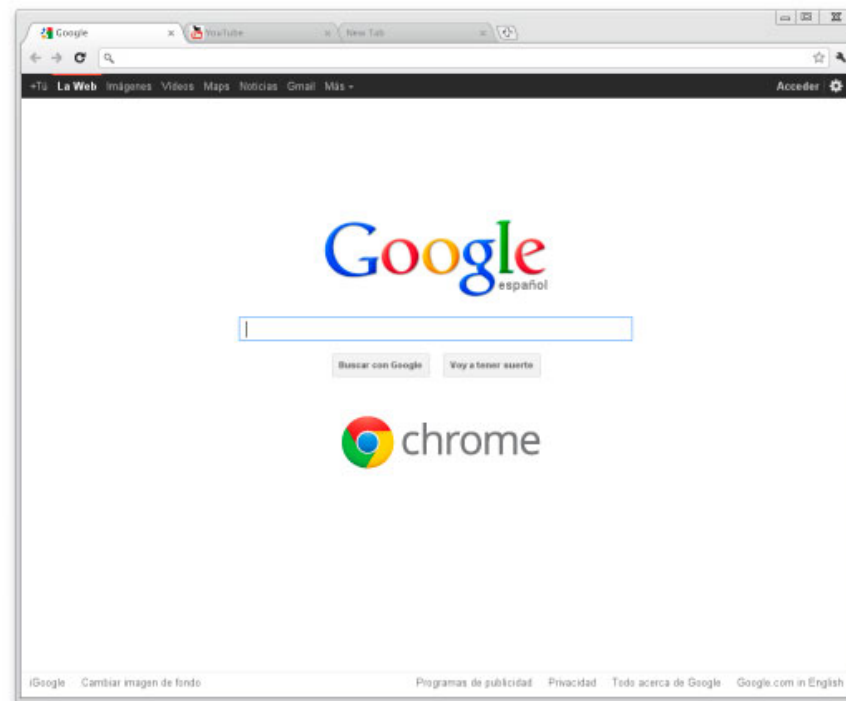




## Tecnologías: Lado cliente

- HTML
- CSS
- Javascript

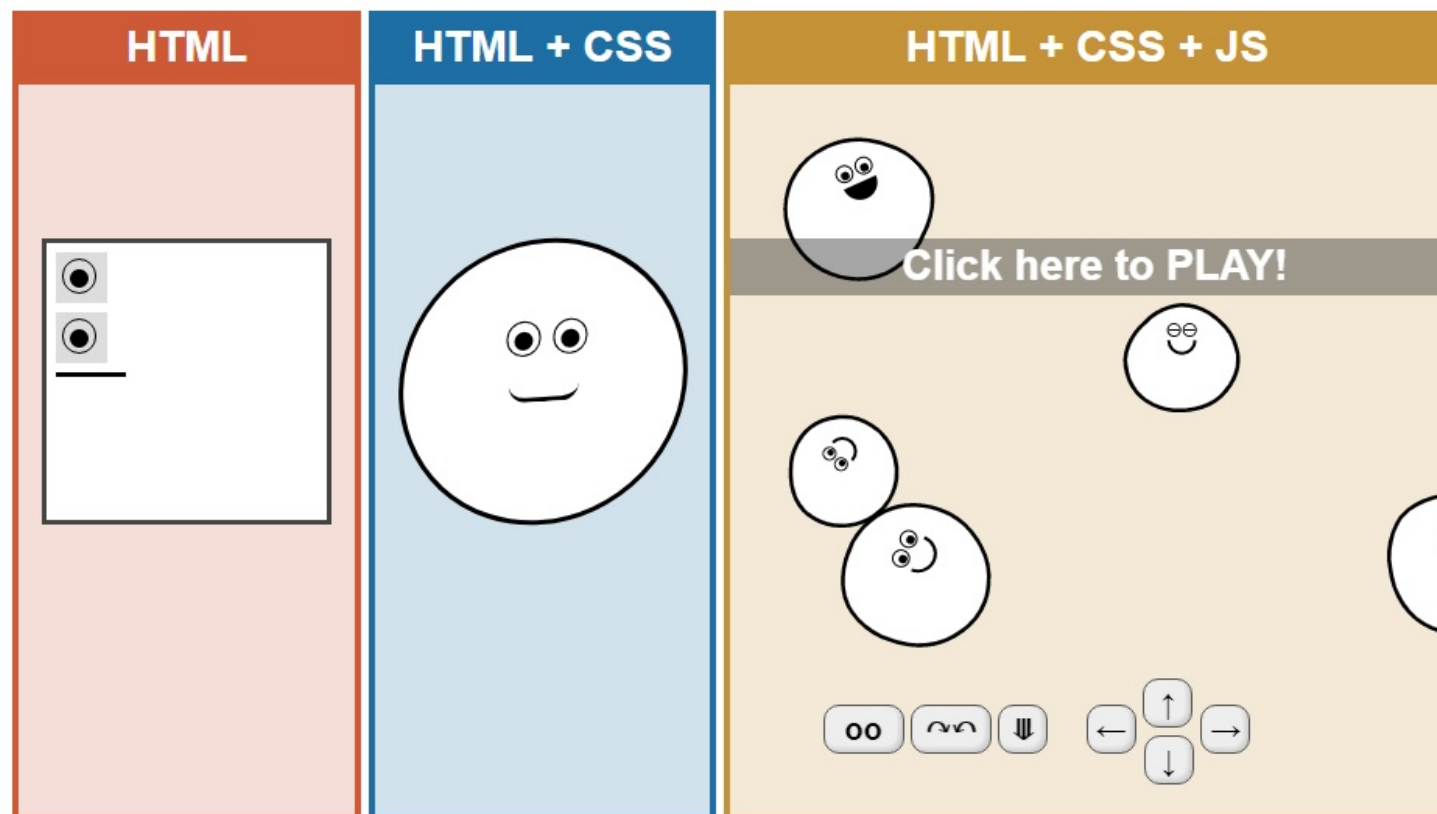
```
<!DOCTYPE html>
<html lang="es">
  <head>
    <link href="bootstrap.min.css" rel="stylesheet">
    <script src="run_prettify.js"></script>
  </head>
  <body>
    ...
```





## Tecnologías: Lado cliente

- HTML
- CSS
- Javascript





## **Tecnologías: Lado cliente**

- **CSS**
  - Style.css (manual)
- Bootstrap
- Materialize
- Etc.
  - Sistema de grillas
  - Estilos para tablas y componentes (botones, formularios, etc.)



## **Tecnologías: Lado cliente**

- **CSS compilers**
  - SASS
  - LESS



## **Tecnologías: Lado cliente**

- **Javascript**
  - **Boilerplate.js (manual)**
- **jQuery**
- **React**
- **Angular**
- **Vue.js**
- **Etc.**





## Tipos de aplicaciones Web

- **Static web app:** El contenido lo genera el servidor y siempre es el mismo (Por ej. Un server de archivos). No se ejecuta código en el server. Cada contenido tiene una URL/página diferente.
- **Dynamic web app:** El contenido lo genera el servidor ejecutando código que construye el contenido. Cada contenido tiene una URL/página diferente.
- **JS web app:** El servidor genera cierto contenido y el resto lo genera el código JS que realiza requests extras al server y consume HTML o JSON que agrega al contenido. Cada contenido tiene una URL/página diferente.

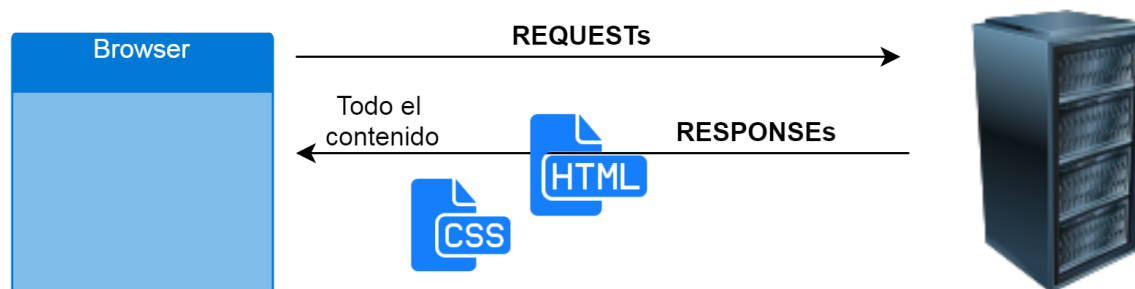


## Tipos de aplicaciones Web

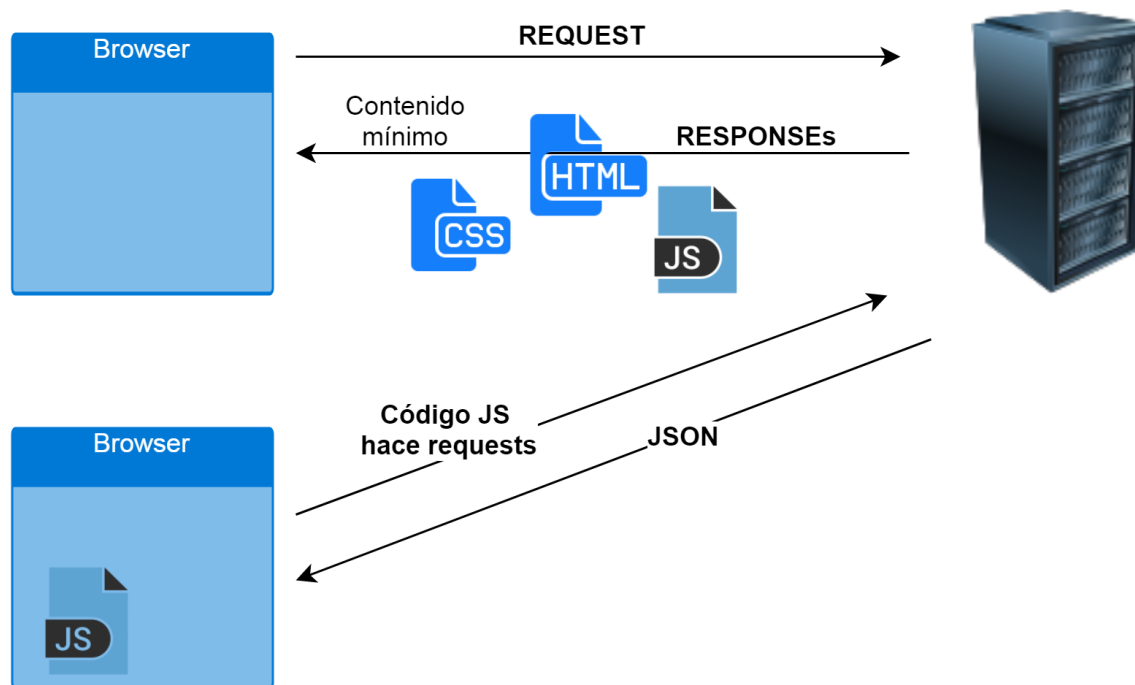
- **Single Page Application:** El código JS que corre en el cliente cambia el contenido de la página sin recargarlo, haciendo requests al server, que consumen JSON/XML del mismo.
- **Progressive Web App:** Es una SPA con mejoras en la interfaz mobile para que sean fluídas y confiables bajo una mala conexión u offline. Funcionan en el navegador y/o desktop y pueden instalarse como una app nativa mobile.



## static/Dynamic web app



## Single Page Application (SPA)





## **Tecnologías: Lado server**

- **Servidores web**

- NodeJS (Javascript)
- Apache web server (PHP)
- Tomcat/Jeti (Java)
- Apache web server (C/C++/Python)
- NGINX (contenido estático)
- Otros



## Tecnologías: Lado server

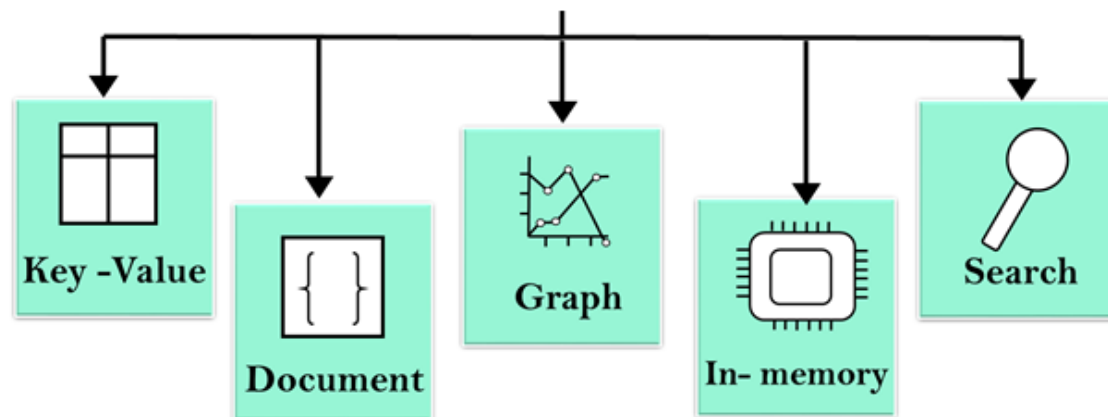
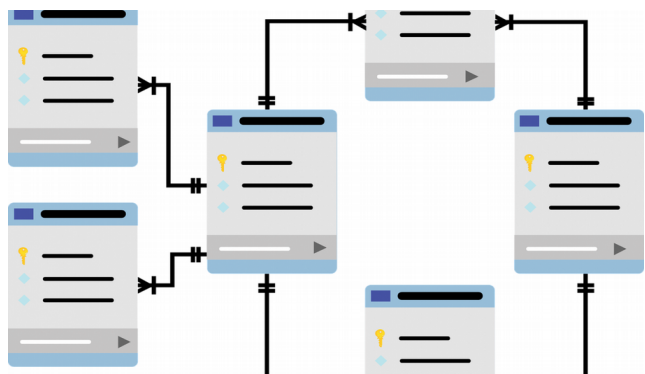
- **Bases de datos**

- Relacionales

- MySQL, MariaDB, Oracle, SQLServer

- No Relacionales

- MongoDB, Cassandra, Redis, Neo4j





## **Tipos de aplicaciones en servidor**

- Servidor de contenido estático (archivos multimedia, archivos js, etc.)
- Servidor de contenido web (HTML,CSS,JS)
- Servicio web RESTful (JSON,XML)
- Servidor websocket
- MQTT broker
- Otros





## Tipos de aplicaciones en servidor

- Servidor de contenido estático (archivos multimedia, archivos js, etc.)
- Servidor de contenido web (HTML,CSS,JS)
- Servicio web RESTful (JSON)
- Servidor websocket
- MQTT broker
- Otros



## Herramientas seleccionadas

- **Cliente:**
  - HTML5
  - CSS3: Materialize
  - JS: Typescript.
- **Server:**
  - NodeJS
  - Servicios web: Express
  - DB: MySQL



## Herramientas desarrollo

- IDE: Visual Studio Code
- OS: Ubuntu 16/18
- Docker



## Bibliografía

- Fielding, Roy T.; Gettys, James; Mogul, Jeffrey C.; Nielsen, Henrik Frystyk; Masinter, Larry; Leach, Paul J.; Berners-Lee, Tim (June 1999). Hypertext Transfer Protocol – HTTP/1.1
- Practical Internet of Things with JavaScript. Arvind Ravulavaru. 2017. Packt>
- Raspberry Pi for Arduino Users – Building IoT and Network Applications and devices. James R. Strickland. 2018. Apress.
- <https://restfulapi.net>