API Rest

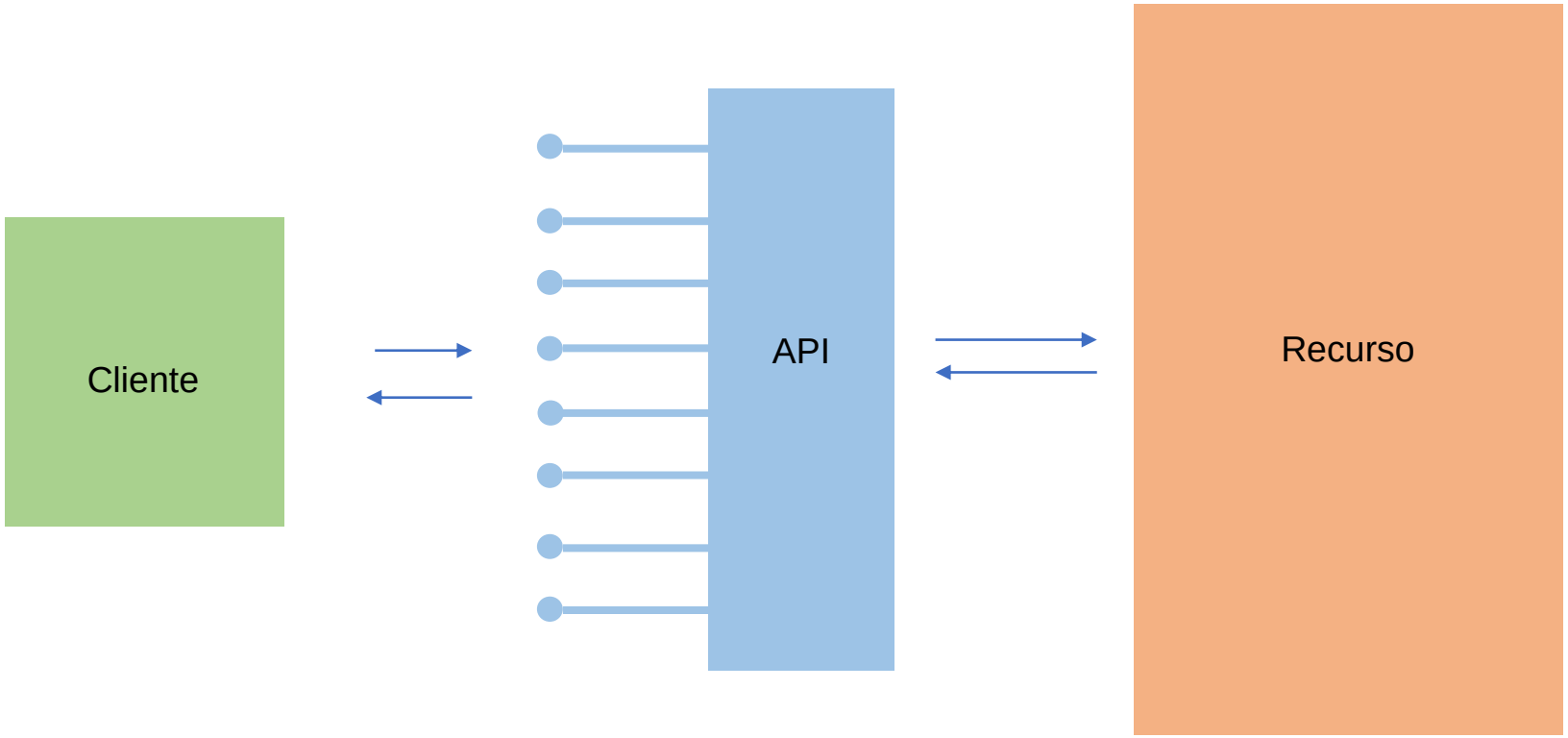
- API
- Procolo Http
- Rest
- API RestFull

API

API

Definición

API es el acrónimo de interfaz de programación de aplicaciones (application programming interface en inglés). Es un conjunto de reglas bien definidas que se utilizan para especificar formalmente la comunicación entre dos componentes de software.



API

tipos de APIs

Existen muchos tipos diferentes de APIs y formas de categorizarlas. Por ejemplo, se pueden categorizar las APIs según quién tiene acceso a ellas. Este marco organizativo incluye:

APIs privadas: Las APIs privadas, también conocidas como APIs internas, se utilizan para conectar diferentes componentes de software dentro de una sola organización, y no están disponibles para uso de terceros. Por ejemplo, una aplicación de redes sociales podría tener una API privada que maneja el flujo de inicio de sesión, otra API privada que maneja el feed y otra API privada que facilita la comunicación entre usuarios. Algunas aplicaciones pueden incluir docenas o incluso cientos de APIs privadas.

APIs públicas: Las APIs públicas proporcionan acceso público a los datos, funciones o servicios de una organización, que los desarrolladores de terceros pueden integrar en sus propias aplicaciones. Algunas APIs públicas están disponibles de forma gratuita, mientras que otras se ofrecen como productos de pago. Por ejemplo, una aplicación de comercio electrónico puede incorporar una API pública de pago, como Stripe, para manejar el procesamiento de pagos sin tener que construir esa funcionalidad desde cero.

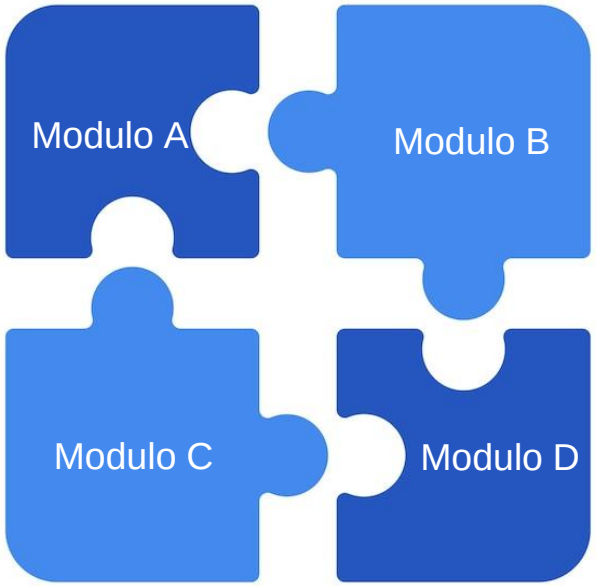
APIs de socios: Las APIs de socios permiten que dos o más empresas compartan datos o funcionalidad para colaborar en un proyecto. No están disponibles para el público en general y, por lo tanto, utilizan mecanismos de autenticación para garantizar que solo sean utilizadas por socios autorizados.

https://www.postman.com/what-is-an-api/#different-types-of-apis

API

Definición

De alguna forma siempre estamos haciendo API en nuestro software para que otras piezas de software las utilicen.



Comunicación entre dispositivos

Formas de comunicación

•Símplex

En este modo solo es posible la transmisión en un sentido, del terminal que origina la información hacia el que la recibe y procesa. Un ejemplo claro de este tipo son las emisoras de radiodifusión.

•Semidúplex (half – dúplex)

Permite la transmisión en ambos sentidos de manera alterna. Un ejemplo de este tipo son las transmisiones efectuadas por radioaficionados.

•Dúplex (full – dúplex)

Consiste en la transmisión en ambos sentidos de manera simultánea. Esta forma de trabajo es la más eficiente. Un ejemplo son las comunicaciones telefónicas.

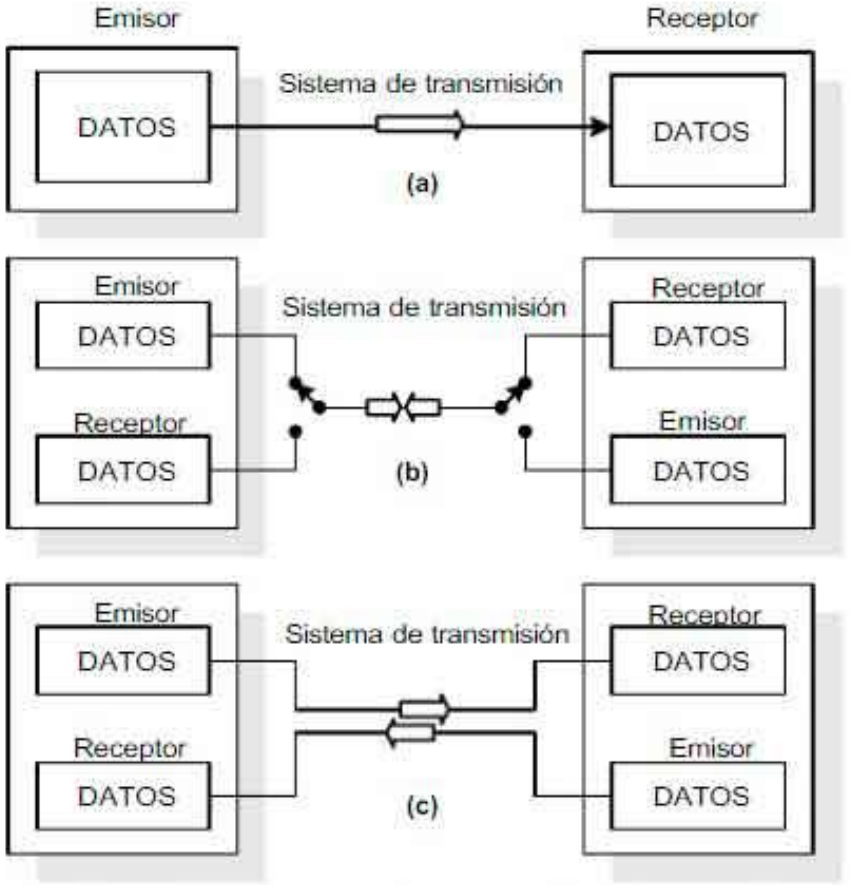
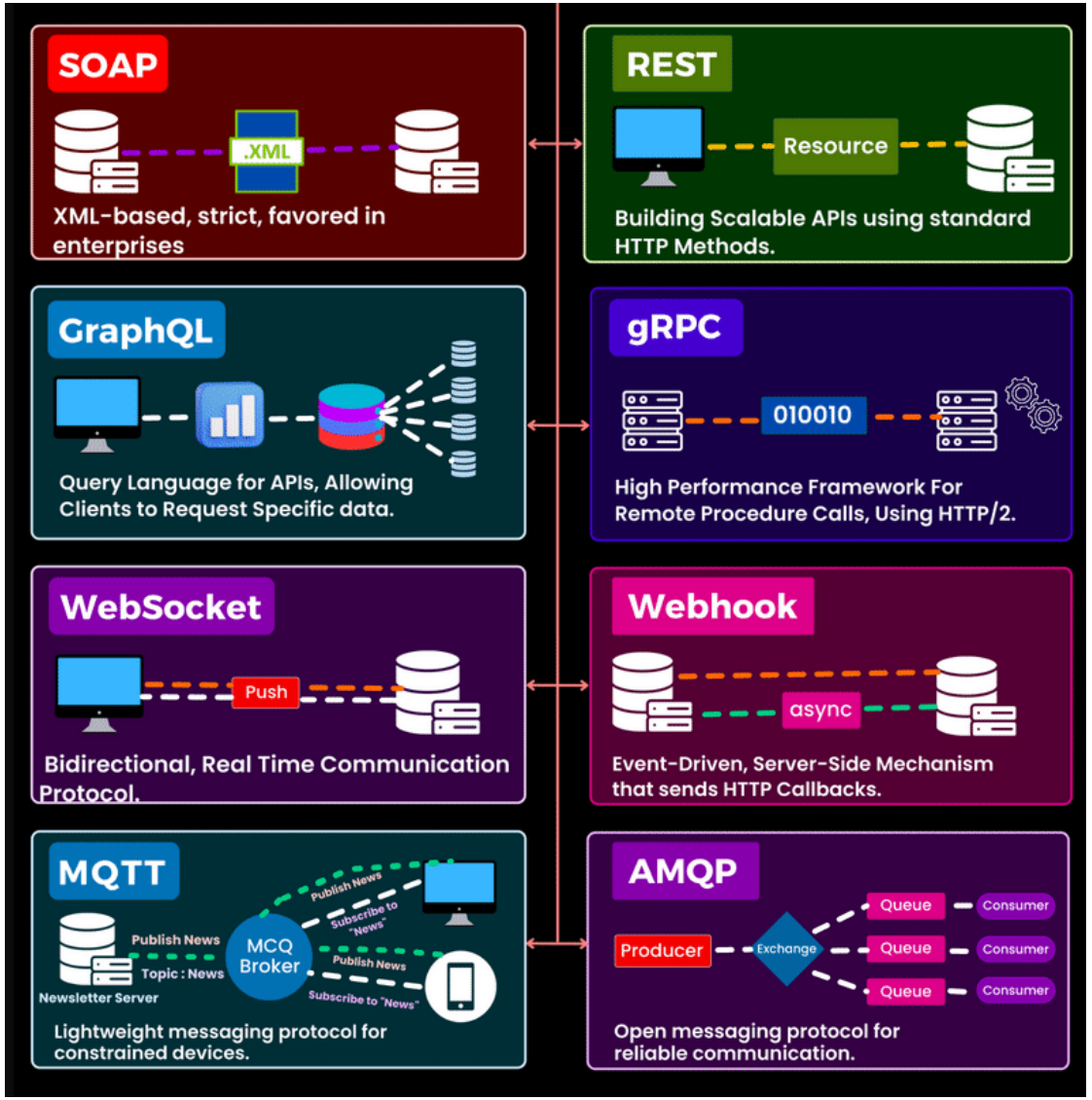


Figura 2.11 Sistemas de transmisión: símplex (a), semidúplex (b), dúplex (c)

API

Tipos de arquitecturas



REST

API REST

Definición

RE es la abreviatura de **RE**presentación, la **S** de Estado y la **T** de Transferencia.

Una API REST es una interfaz de comunicación entre sistemas de información que usa el protocolo de transferencia de hipertexto (*hypertext transfer protocol* o HTTP, por su siglas en inglés) para obtener datos o ejecutar operaciones sobre dichos datos en diversos formatos, como pueden ser XML o JSON

Una API es **RESTful** si cumple con los estándares API REST.

API REST

Restricciones de rest:

Las restricciones determinadas por la arquitectura *Rest* son:

- **Cliente-servidor:** las aplicaciones existentes en el servidor y el cliente deben estar separadas.
- **Sin estado:** las requisiciones se realizan de forma independiente, es decir, cada una ejecuta solo una determinada acción.
- **Caché**: la API debe utilizar la caché para evitar llamadas recurrentes al servidor.
- **Interfaz uniforme**: cada recurso debe tener un único Identificador Uniforme de Recursos (Ruta).

https://www.postman.com/what-is-an-api/#different-types-of-apis

Arquitectura Cliente-Servidor

Petición y respuesta

La **arquitectura cliente-servidor** es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta

https://es.wikipedia.org/wiki/Cliente-servidor

En esta arquitectura el modelo, cliente-servidor ayuda en la separación de responsabilidades entre la interfaz de usuario y el almacenamiento de datos. Es decir, cuando se realiza una solicitud REST, el servidor envía una representación de los estados que se solicitaron.

Arquitectura Cliente-Servidor

Petición y respuesta

Front End

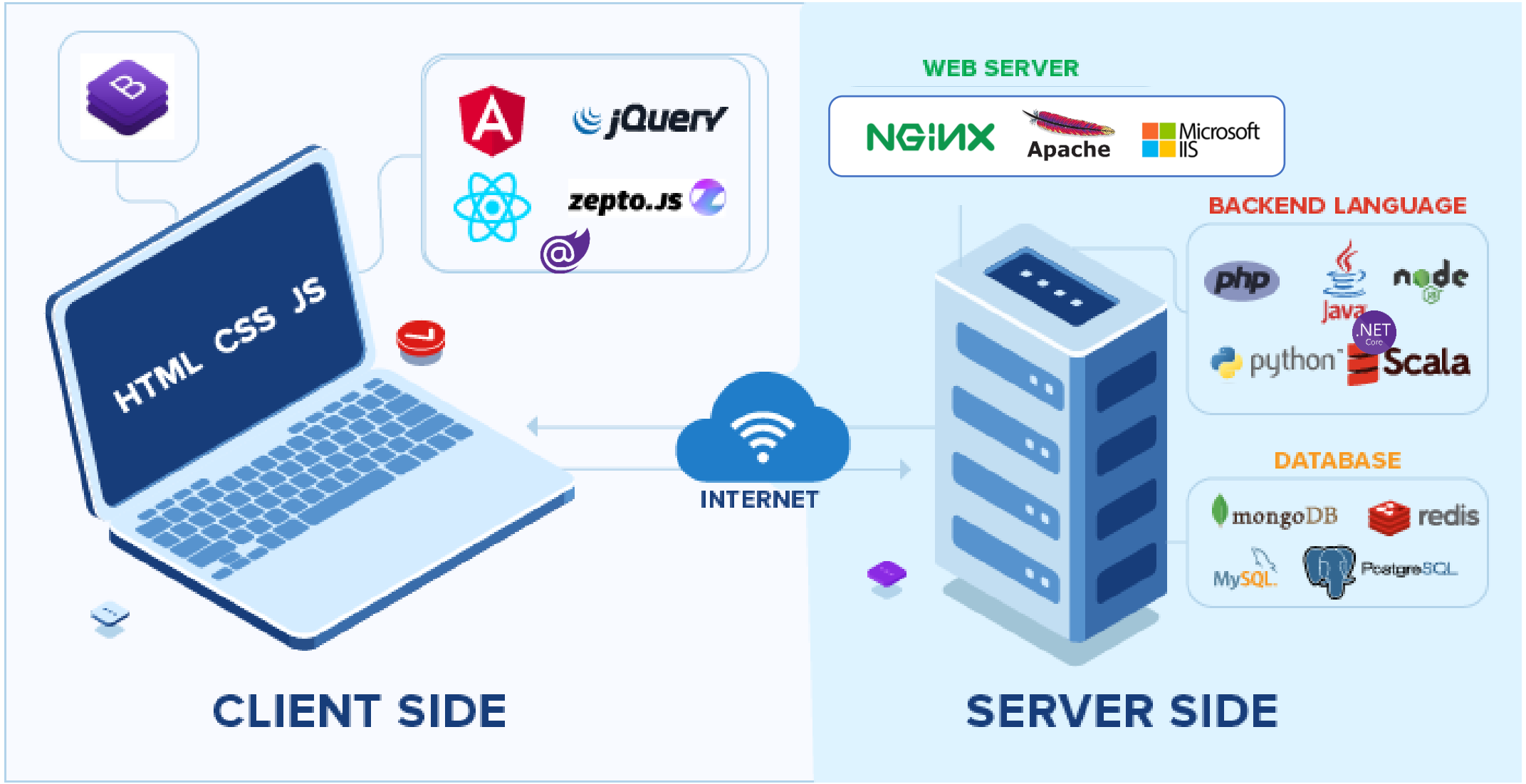
- Markup and web languages such as HTML, CSS and Javascript
- Asynchronous requests and Ajax
- Specialized web editing software
- Image editing
- Accessibility
- Cross-browser issues
- Search engine optimisation

Back End

- Programming and scripting such as Python, Ruby and/or Perl
- Server architecture
- Database administration
- Scalability
- Security
- Data transformation
- Backup

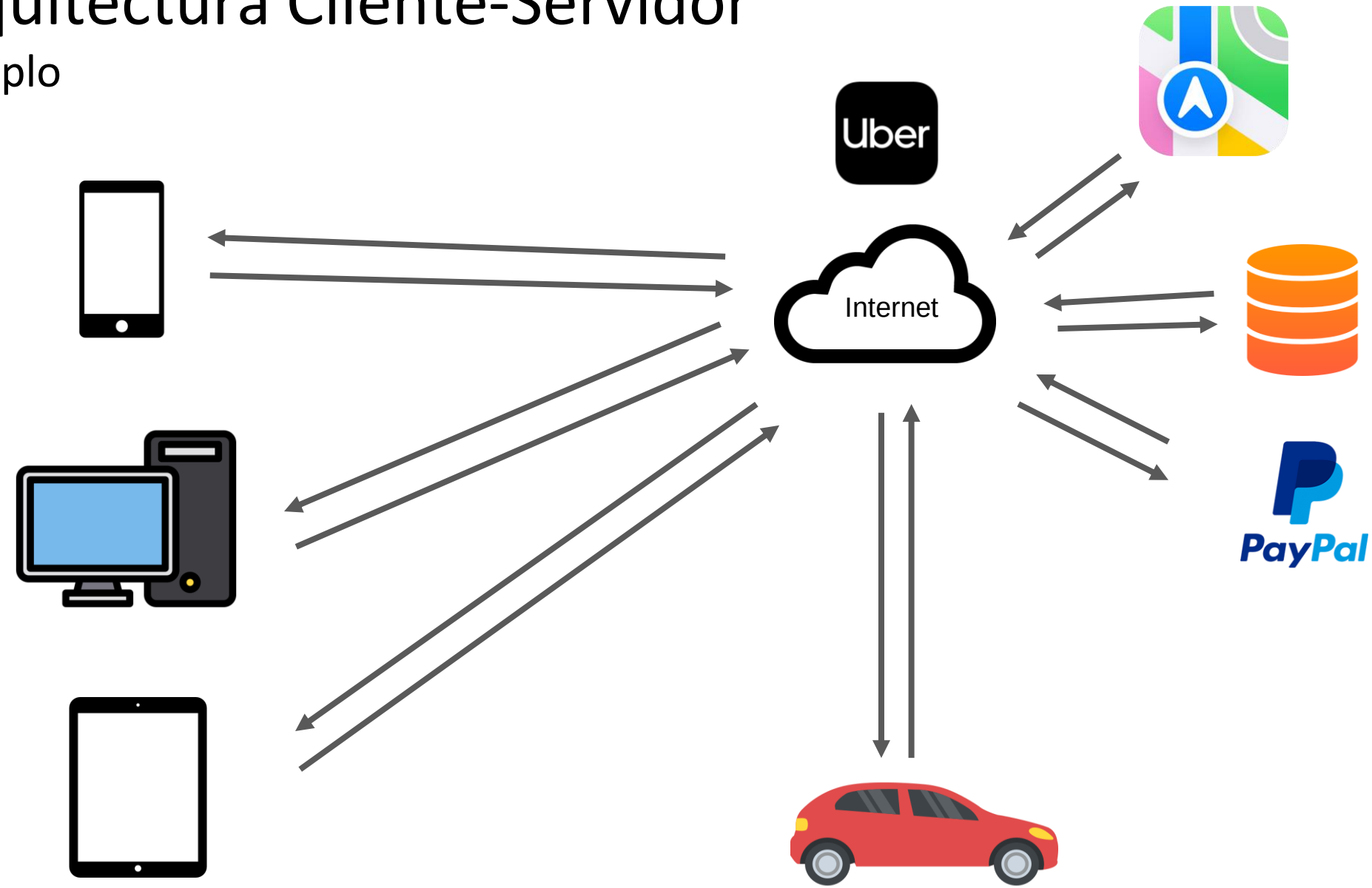
Arquitectura Cliente-Servidor

Petición y respuesta



Arquitectura Cliente-Servidor

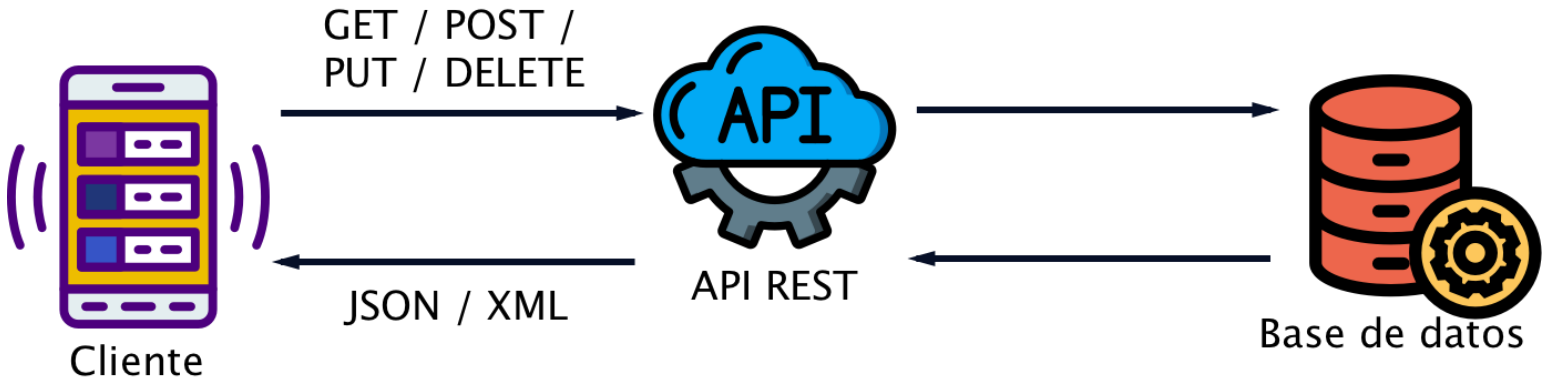
Ejemplo



API REST

Usos y funciones

Una API Rest **permite que la aplicación acceda a bases de datos desde diferentes servidores**, lo que a menudo es importante para el desarrollo en aplicaciones grandes. Por lo tanto, su uso garantiza una mayor visibilidad y credibilidad a la hora de utilizar estos recursos

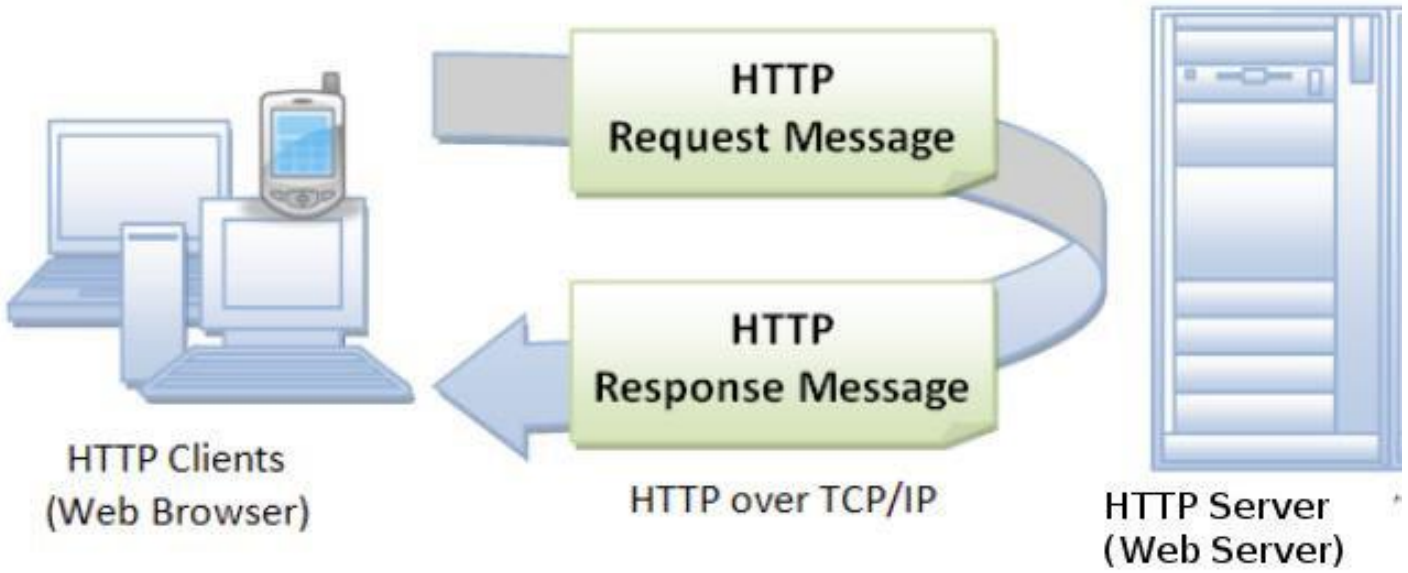


Protocolo HTTP

Protocolo Http

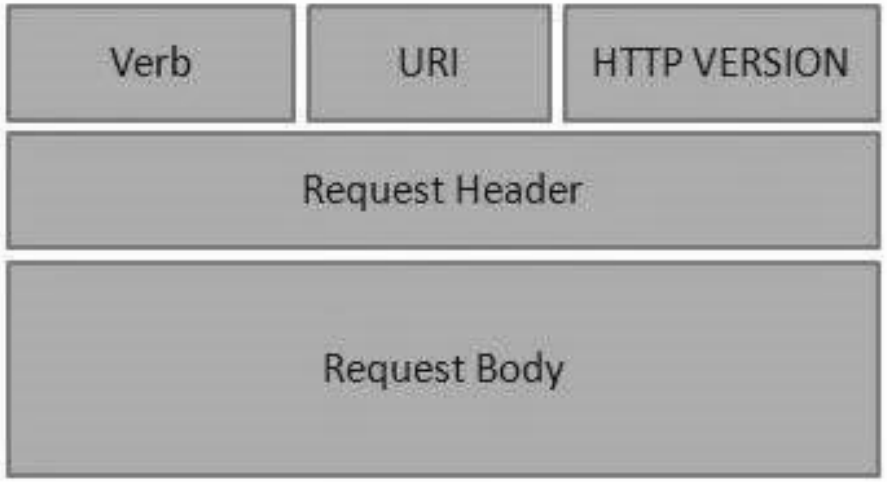
Hyper-Text Transfer Protocol, major conocido como: HTTP

La comunicación en HTTP se centra en un concepto llamado ciclo de solicitud-respuesta. El cliente envía al servidor una solicitud (Request HTTP) para hacer algo. El servidor, a su vez, puede devolver al cliente una respuesta (Response HTTP) diciendo si el servidor puede o no hacer lo que el cliente pidió.

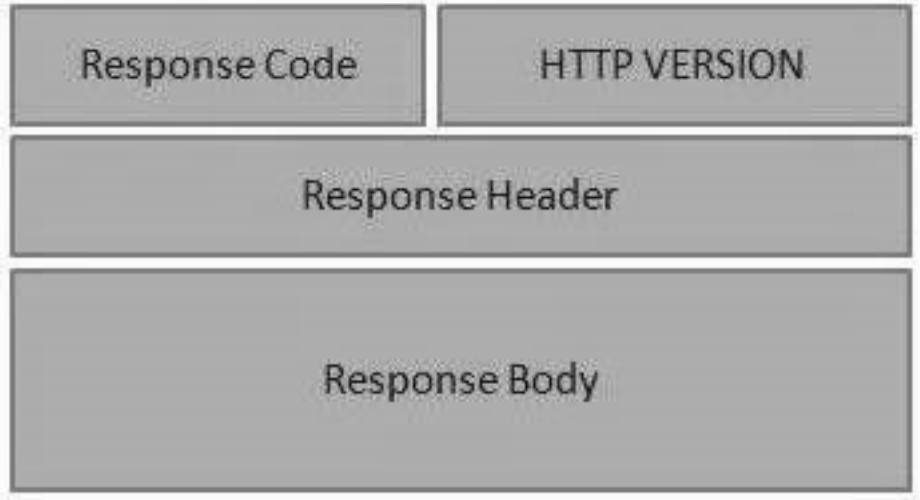


Protocolo Http

Los mensajes HTTP están compuestos de texto, codificado en ASCII, y pueden comprender múltiples líneas. En HTTP/1.1, y versiones previas del protocolo, estos mensajes eran enviados de forma abierta a través de la conexión. En HTTP/2.0 los mensajes, que anteriormente eran legibles directamente, se conforman mediante tramas binarias codificadas para aumentar la optimización y rendimiento de la transmisión



HTTP Request



HTTP Response

Protocolo Http

Estructura de un REQUEST HTTP (Petición HTTP)

•**Verbo** – Indica que método HTTP se utiliza, puede ser GET, POST, DELETE, PUT, etc.

•**URI** – Uniform Resource Identifier (URI) identifica la ruta al recurso que se hace referencia

•**HTTP Version** – Indica la versión HTTP. Por ejemplo, HTTP v1.1.

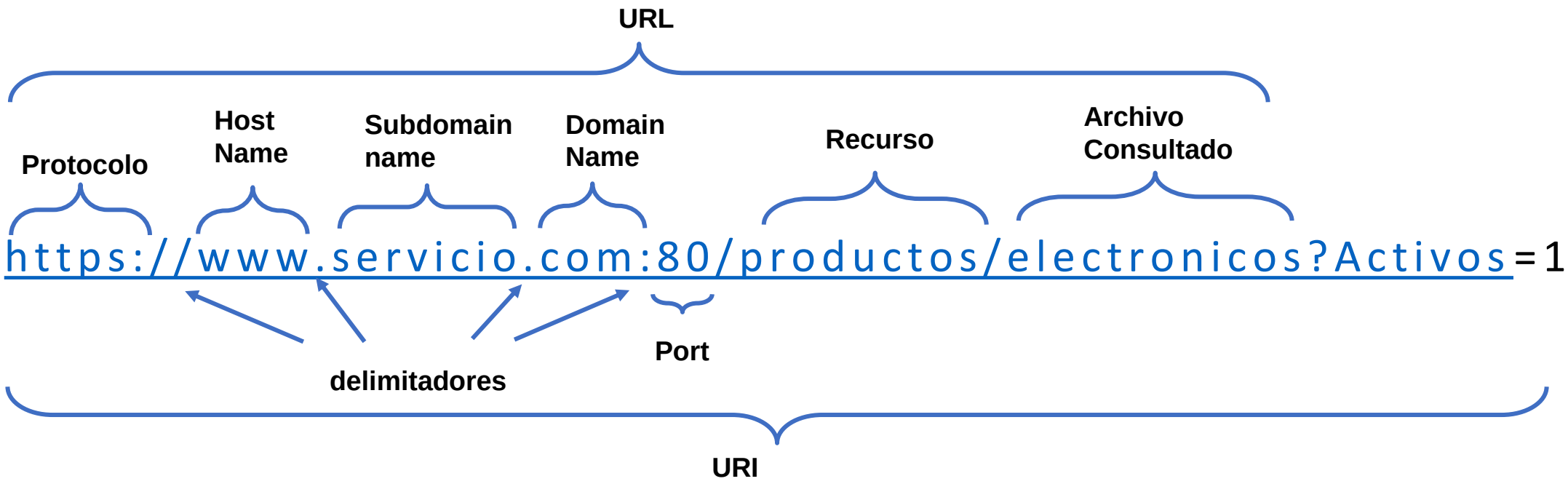
•**Request Header** – Contiene Metada para el Request HTTP estructurado como pares de (Llave-Valor). Como son: cliente (o Navegador) tipo, formato soportado por el cliente, formato del mensaje (body), cache settings, etc.

•**Request Body** – (opcional) Contenido del mensaje.

Request Http

URI - URL

Una URL o localizador de recursos uniforme es un identificador que sólo indica la ubicación de una página web. Se refiere a una dirección web y a sus posibilidades de acceso, como son HTTP, FTP, Etc

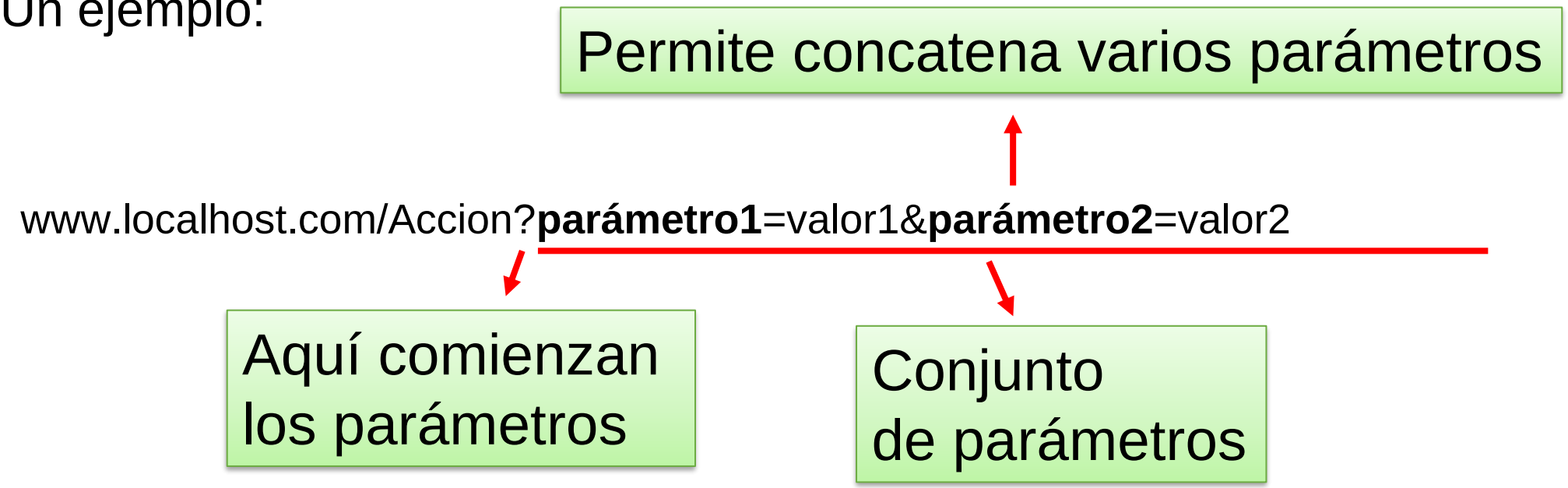


https://en.wikipedia.org/wiki/Query_string

Request Http

Ejemplo Query String

Podemos pasar parámetros al servidor incluyéndolos en la URL. Normalmente se componen por un nombre y un valor separados por el signo igual, pudiéndose concatenar un número arbitrario de ellos mediante el signo &.
Un ejemplo:



https://en.wikipedia.org/wiki/Query_string

Protocolo Http

Estructura de un REQUEST HTTP (Petición HTTP)

Verbos HTTP

GET	El método GET es el método más común, generalmente se usa para solicitar a un servidor que envíe un recurso. Los parámetros los puede ver cualquiera persona simplemente mirando la URL de la web y los parámetros que esta lleva.
POST	POST consiste en datos "ocultos" (porque el cliente no los ve) enviados por un formulario cuyo método de envío es post. Es adecuado para formularios. Los datos no son visibles en la url.
PUT	Utilizado normalmente para actualizar contenidos, pero también pueden crearlos. Tampoco muestra ninguna información en la URL
DELETE	Elimina un recurso identificado en la URI. Si se elimina correctamente devuelve 200 (OK) junto con un body response, o 204 sin body.
HEAD	Es idéntico a GET, pero el servidor no devuelve el contenido en el HTTP response. Cuando se envía un HEAD request, significa que sólo se está interesado en el código de respuesta y los headers HTTP, no en el propio documento.

Protocolo Http

Estructura de un RESPONSE HTTP (Petición HTTP)

- **Status/Response Code** – Indica el estado del servidor para lo que se solicitó. Por ejemplo, 404 significa que un recurso no fue encontrado y 200 significa que una respuesta es OK.

- **HTTP Version** – indica la versión de HTTP utilizada. Por ejemplo HTTP v1.1.

- **Response Header** – Contiene metada de la respuesta HTTP como pares Llave-valor. Por ejemplo, content length, content type, response date, server type, etc.

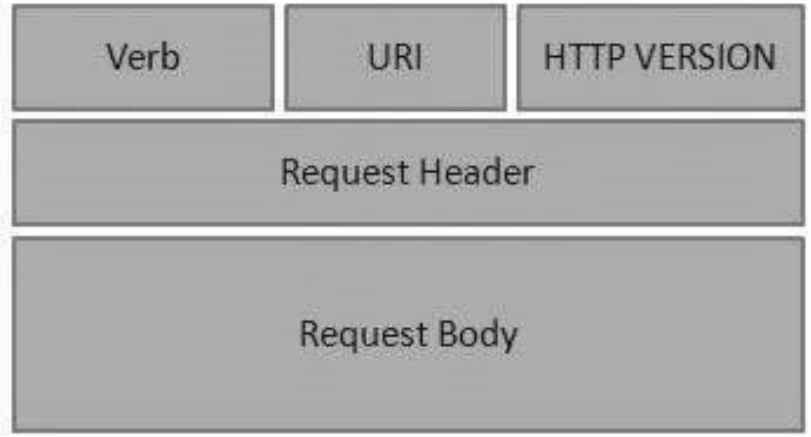
- **Response Body (opcional)** – Contenido del mensaje de respuesta o representación del rescurso solicitado.

Request Http

Estructura de un REQUEST HTTP (Petición HTTP)

Ejemplo de un Post

```
POST http://MyService/Person/ HTTP/V2.0
Host: MyService
Content-Type: text/Json; charset=utf-8
Content-Length: 123
{
 "Person":
            {
            "ID": 1,
            "Name": “Pablo Paramo",
            "Email": “pparamo@gmail.com",
            "Country": “Argentina"
            }
}
```

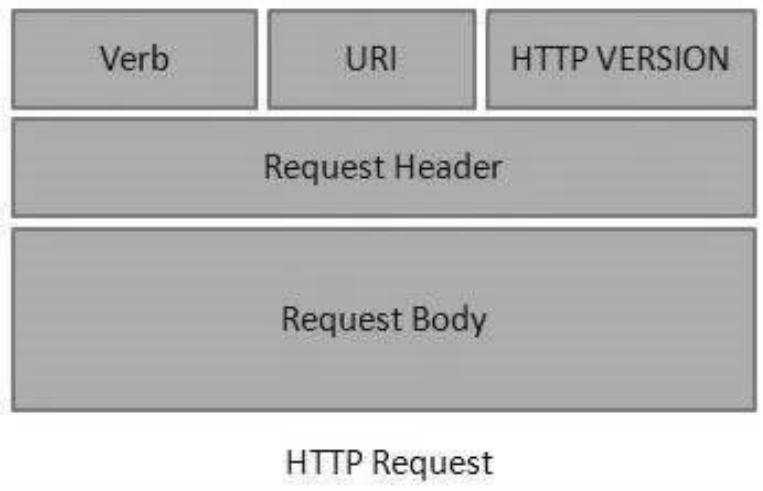


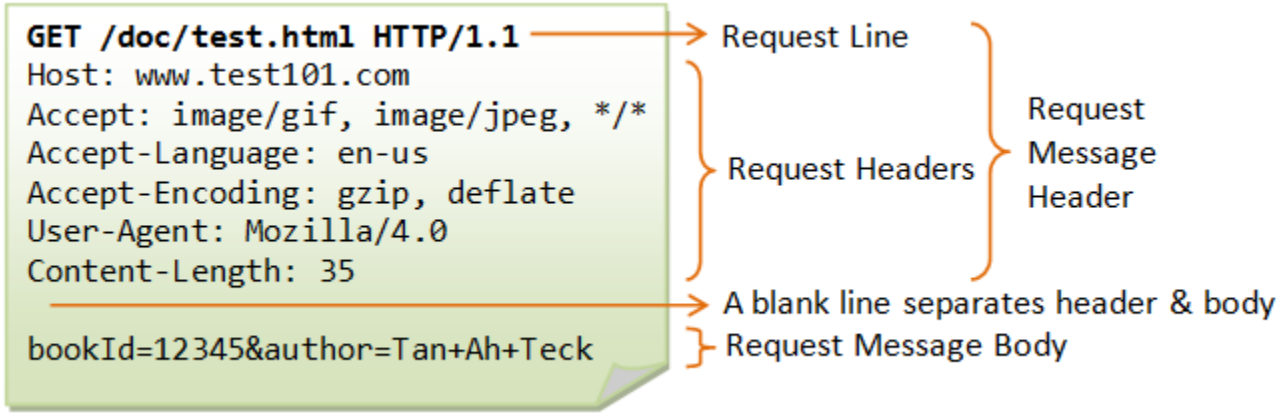
HTTP Request

Protocolo Http

Estructura de un REQUEST HTTP (Petición HTTP)

Ejemplo de un GET

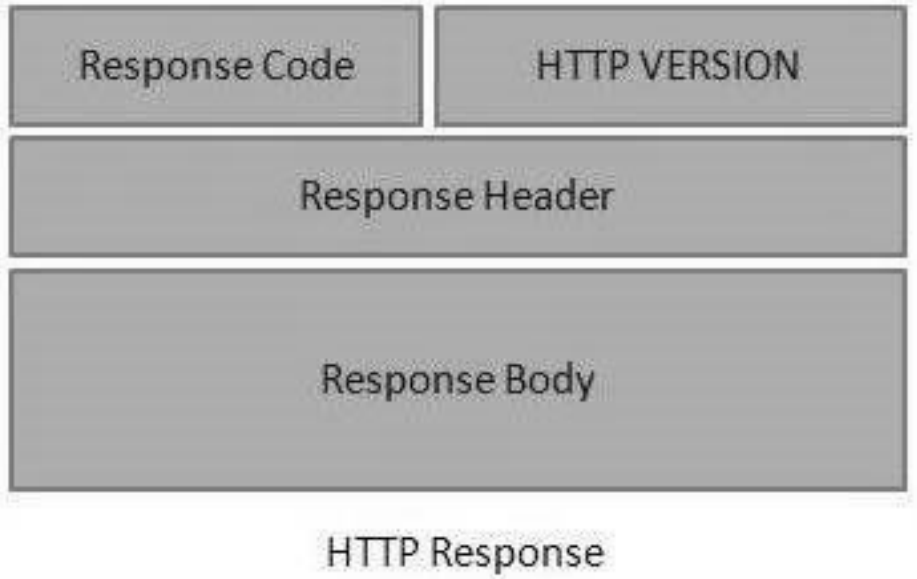




Protocolo Http

Estructura de un RESPONSE HTTP (Petición HTTP)

```
200 OK     HTTP/1.1
Date: Wed, 23 Oct 2019 19:51:41 GMT
Server: Apache/2.4.29
X-Robots-Tag: noindex
Link: <http://yourdomain.com/wp-json/>; rel="https://api.w.org/"
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages
Access-Control-Allow-Headers: Authorization, Content-Type
Allow: GET
Transfer-Encoding: chunked
Content-Type: application/json; charset=UTF-8
```



Protocolo Http

Estructura de un RESPONSE HTTP (Petición HTTP)

Los códigos de estado (**Status/Response Code**) de respuesta HTTP indican si se ha completado satisfactoriamente una solicitud HTTP específica. Las respuestas se agrupan en cinco clases:

- Respuestas informativas (100–199),
- Respuestas satisfactorias (200–299),
- Redirecciones (300–399),
- Errores de los clientes (400–499),
- y errores de los servidores (500–599).

Protocolo Http

Estructura de un RESPONSE HTTP (Petición HTTP)

Los códigos de estado (**Status/Response Code**) más utilizados son:

- 200 (OK) solicitud cumplida con éxito.
- 201 (Created) objeto o recurso creado con éxito.
- 204 (Non Content) objeto o recurso eliminado con éxito.
- 400 (Bad Request) ocurrió un error en la solicitud.
- 404 (Not Found) ruta o colección no encontrada.
- 500 (Internal Server Error), se ha producido algún error del servidor.

Bibliografía

https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/http/httpclient

https://en.wikipedia.org/wiki/Query_string

API REST
https://www.aluracursos.com/blog/rest-concepto-y-fundamentos?gclid=Cj0KCQjwoeemBhCfARIsADR2QCvlNJNB5QGQINBgSrBVd7quxnglnOau1IGIS7oVIcavAuocCV3FINsaAjiSEALw_wcB

https://openwebinars.net/blog/que-es-rest-conoce-su-potencia/#:~:text=REST%20es%20una%20interfaz%20para,específicos%2C%20como%20XML%20y%20JSON.

https://rockcontent.com/es/blog/api-rest/

https://sentai.eu/info/restful-web-services/

Request / Response HTTP

https://solace.com/blog/inside-a-solace-message-using-header-properties/
https://softwarelab.org/es/blog/que-es-una-url/

Verbos HTTP

https://developer.mozilla.org/es/docs/Web/HTTP/Methods

URI / URL

https://www.hostinger.com.ar/tutoriales/uri-vs-url#:~:text=La%20URL%20identifica%20la%20dirección,-476-35557-4.