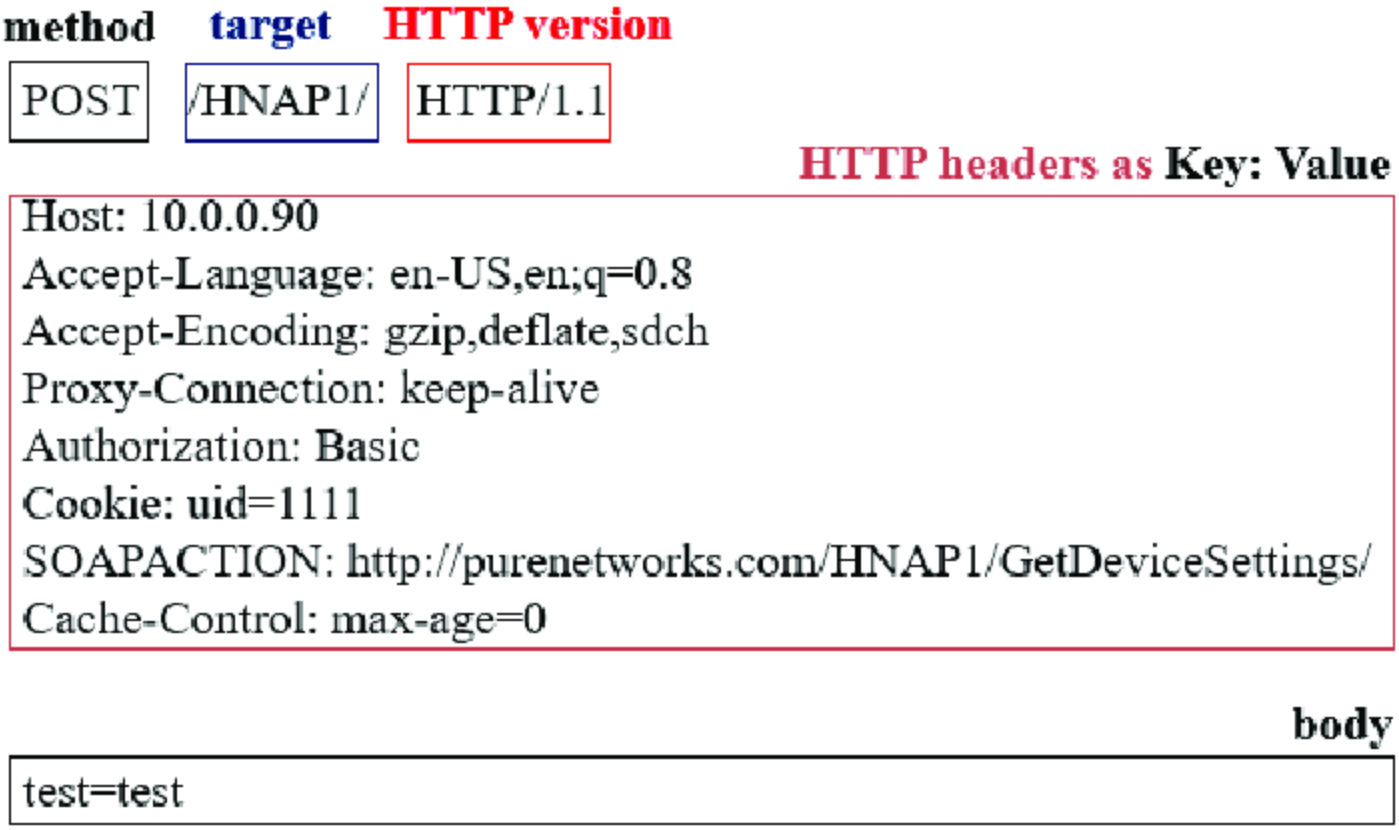
Lectura y Videos

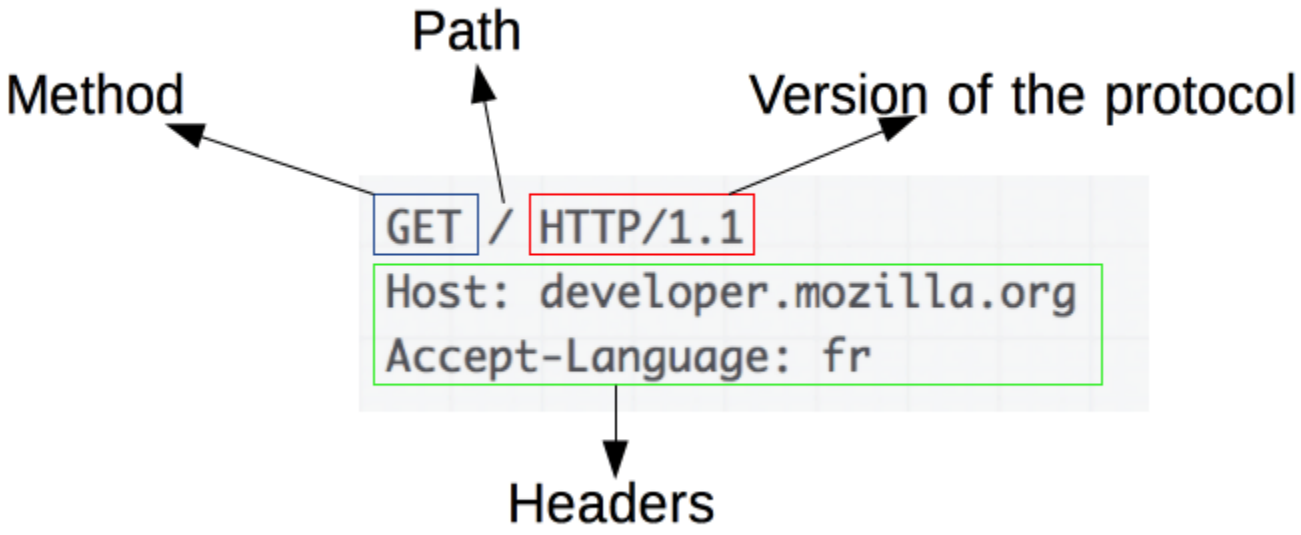
Un sistema Distribuido implica un conjunto de sistemas de computo coordinando acciones a través de mensajes.

Sockets TCP por su confiabilidad, Orientado a Conexión, solo funciona con Unicast.

Al ser Orientado a Conexión, cuando uno o varios computadores quieren enviar mensajes a la misma computadora, cada computadora requiere su propio socket para enviar mensajes.

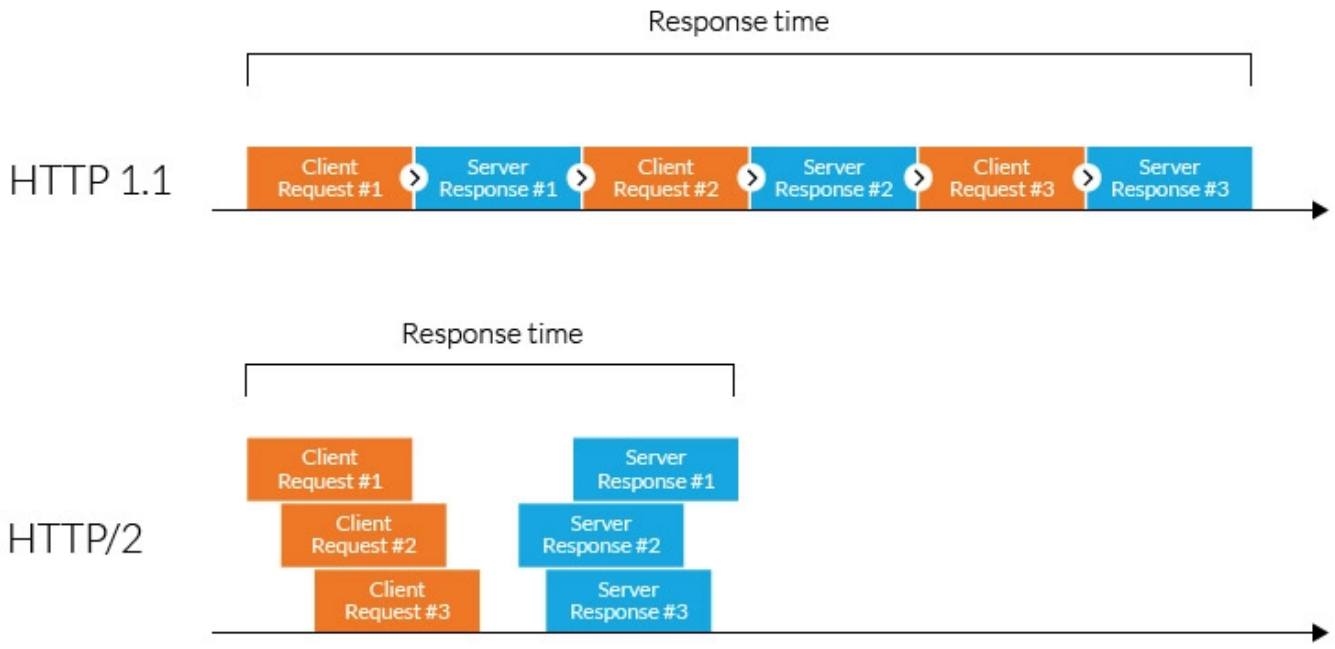
El protocolo HTTP usa un esquema Solicitud / Respuesta, es decir una Transacción HTTP esta conformado por un mensaje de Solicitud y un Mensaje de Respuesta que puede estar vacío.





- Método: Indica como se va a comportar el mensaje
  - Get → Recuperar Datos (no requiere Cuerpo del mensaje)
    - Se utilizará para pedir información entre los miembros del sistema distribuido o microservicios.
  - Post → Enviar Información (se especifican en el Cuerpo del mensaje)
- Ruta Relativa: Permite dirigir el mensaje hacia la ruta relativa dentro del servidor
  - La ruta relativa es la que viene en un URL después del Host y el puerto.
  - Nos ayuda a definir que parte de nuestro código en el servidor atenderá dicha petición.
  - Podemos pasar parámetros a los métodos dentro de la ruta relativa.
- Versión: Indica la version del Protocolo HTTP que se está utilizando
- Headers: permiten enviar metadatos
  - content-length → especifica la cantidad de bytes que se envían
  - content-type → indica el tipo de dato que se envía.

Se usará HTTP 1.1 y HTTP/2



HTTP 1.1 trabaja de manera lineal al hacer solicitudes al Cliente o al servidor.

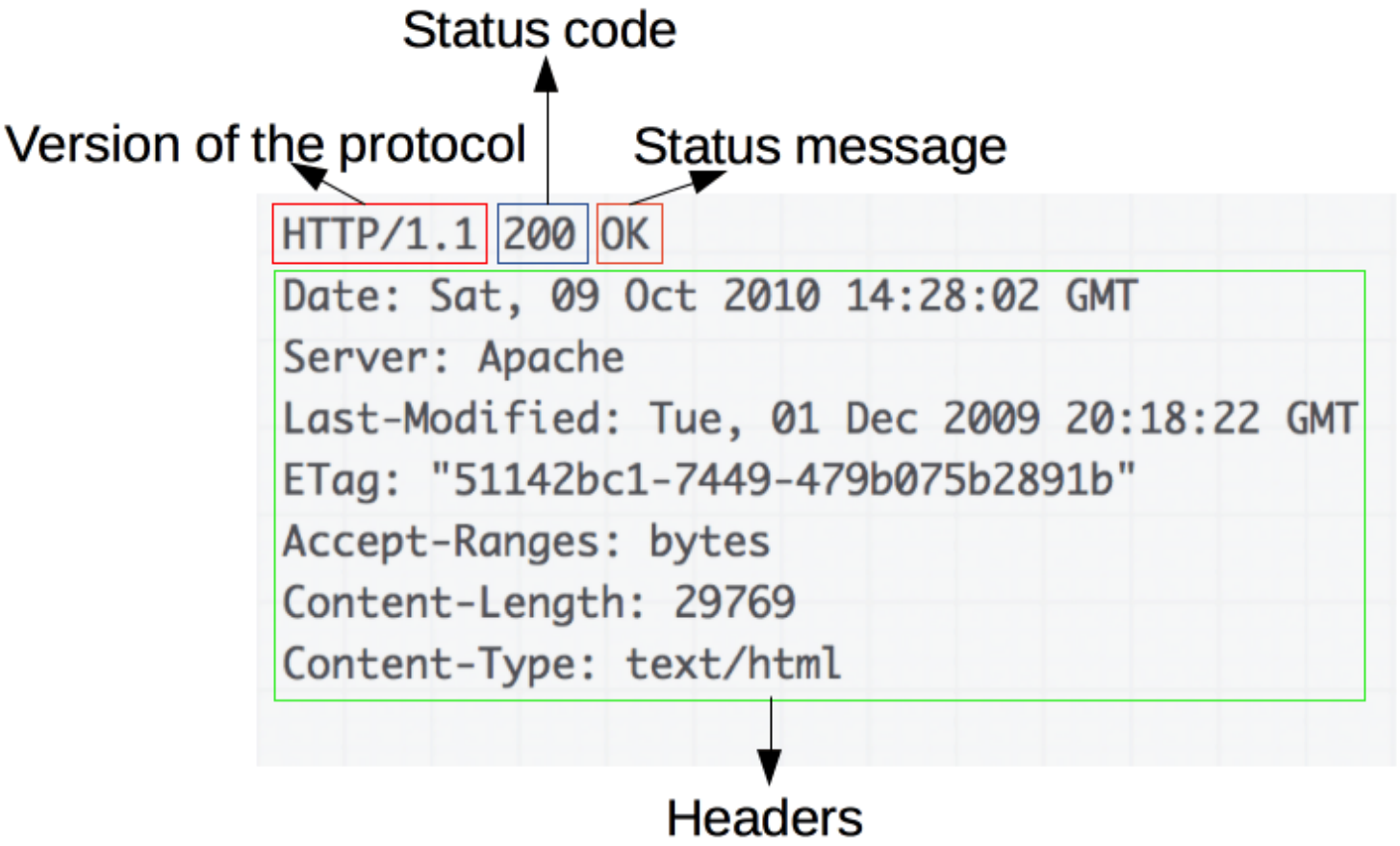
HTTP/2 puede trabajar de manera simultanea e intercalar multiples solicitudes en la misma conexión TCP de manera transparente. Lo hace al dividir la conexión TCP en varios Streams independientes.

Lista de Headers HTTP: List of HTTP header fields

Se pueden crear nuestros propios Headers para depurar la aplicación.

Para la respuesta HTTP es muy parecida a la solicitud.

se cambia el Método por Status Message, el cual incluye un Status Code, los cuales ya están definidos.



Status Codes

1. Informational responses (`100` – `199`)
2. Successful responses (`200` – `299`)
3. Redirection messages (`300` – `399`)
4. Client error responses (`400` – `499`)
5. Server error responses (`500` – `599`)

Mozilla Developer References

Ejemplo HTTP Server

```
giokim in ~ λ curl -v localhost:8081/status
*   Trying 127.0.0.1:8081...
* Connected to localhost (127.0.0.1) port 8081 (#0)
> GET /status HTTP/1.1
> Host: localhost:8081
> User-Agent: curl/7.79.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Tue, 25 Apr 2023 15:37:37 GMT
< Content-length: 23
<
El servidor está vivo
* Connection #0 to host localhost left intact
giokim in ~ λ
```

Peticion del Cliente tipo GET y protocolo HTTP ver. 1.1

Servidor al que se le hizo la peticion

Respuesta del servidor tipo 200 (es decir todo OK)

Header de tipo Date para saber que fecha y hora se hizo la peticion

Tamaño en Bytes del mensaje

```
giokim in ~ λ curl -v localhost:8081/start
*   Trying 127.0.0.1:8081...
* Connected to localhost (127.0.0.1) port 8081 (#0)
> GET /start HTTP/1.1
> Host: localhost:8081
> User-Agent: curl/7.79.1
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 404 Not Found
< Content-Length: 50
< Content-Type: text/html
<
* Connection #0 to host localhost left intact
<h1>404 Not Found</h1>No context found for requestgiokim in ~ λ
```

La ruta relativa indica la URI o Endpoint al que va dirigido el mensaje

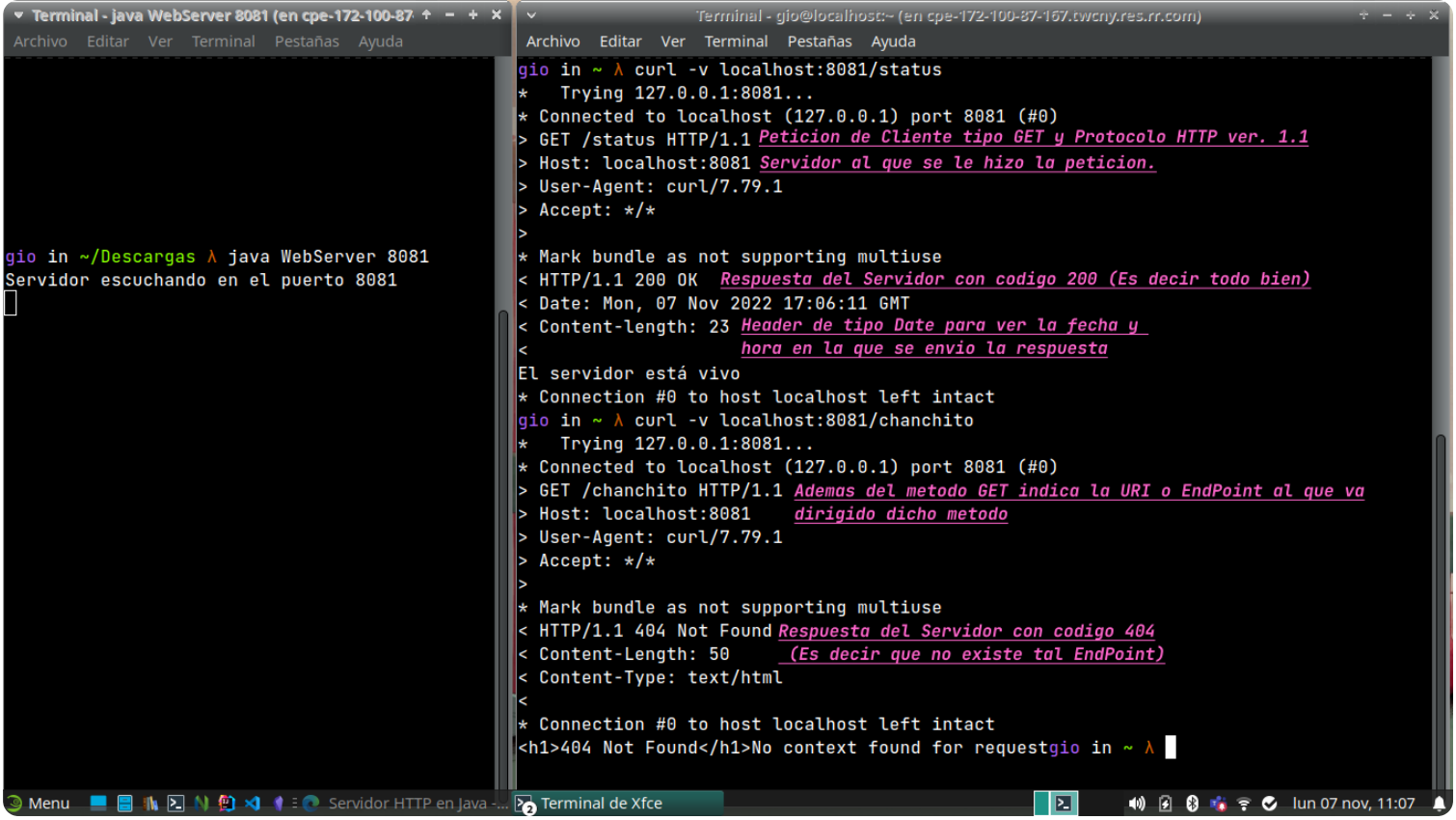
Respuesta del servidor con codigo 404 es decir no se encontro dicho endpoint

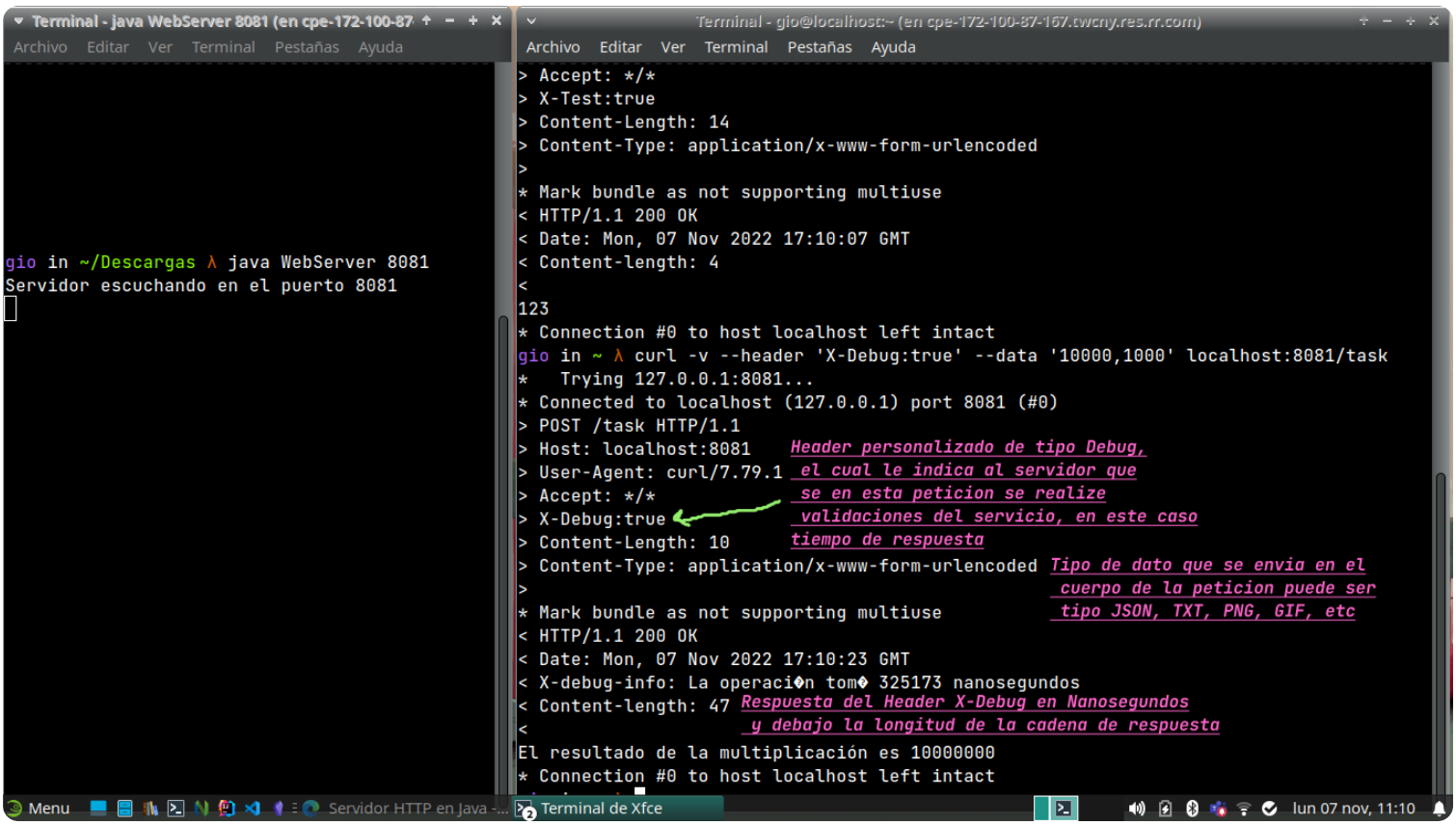
```
giokim in ~ λ curl -v --data '50,100' localhost:8081/task
*   Trying 127.0.0.1:8081...
* Connected to localhost (127.0.0.1) port 8081 (#0)
> POST /task HTTP/1.1
> Host: localhost:8081
> User-Agent: curl/7.79.1
> Accept: */*
> Content-Length: 6
> Content-Type: application/x-www-form-urlencoded
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Tue, 25 Apr 2023 15:40:24 GMT
< Content-length: 43
<
El resultado de la multiplicación es 5000
* Connection #0 to host localhost left intact
giokim in ~ λ
```

```
giokim in ~ λ curl -v --data '5000543232236,176540324562345' localhost:8081/task
*   Trying 127.0.0.1:8081...
* Connected to localhost (127.0.0.1) port 8081 (#0)
> POST /task HTTP/1.1
> Host: localhost:8081
> User-Agent: curl/7.79.1
> Accept: */*
> Content-Length: 29
> Content-Type: application/x-www-form-urlencoded
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Tue, 25 Apr 2023 15:41:15 GMT
< Content-length: 66
<
El resultado de la multiplicación es 882797525206981168395753420
* Connection #0 to host localhost left intact
giokim in ~ λ
```

```
giokim in ~ λ curl -v --header 'X-Test:true' --data '50,100' localhost:8081/task
*   Trying 127.0.0.1:8081...
* Connected to localhost (127.0.0.1) port 8081 (#0)
> POST /task HTTP/1.1
> Host: localhost:8081
> User-Agent: curl/7.79.1
> Accept: */*
> X-Test:true
> Content-Length: 6
> Content-Type: application/x-www-form-urlencoded
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Tue, 25 Apr 2023 15:43:50 GMT
< Content-length: 4
<
123
* Connection #0 to host localhost left intact
giokim in ~ λ
```

```
giokim in ~ λ curl -v --header 'X-Debug:true' --data '50,100' localhost:8081/task
*   Trying 127.0.0.1:8081...
* Connected to localhost (127.0.0.1) port 8081 (#0)
> POST /task HTTP/1.1
> Host: localhost:8081
> User-Agent: curl/7.79.1
> Accept: */*
> X-Debug:true
> Content-Length: 6
> Content-Type: application/x-www-form-urlencoded
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Tue, 25 Apr 2023 15:45:07 GMT
< X-debug-info: La operaci◆n tom◆ 829856 nanosegundos
< Content-length: 43
<
El resultado de la multiplicación es 5000
* Connection #0 to host localhost left intact
```





Teoría de DSD

Práctica del protocolo HTTP