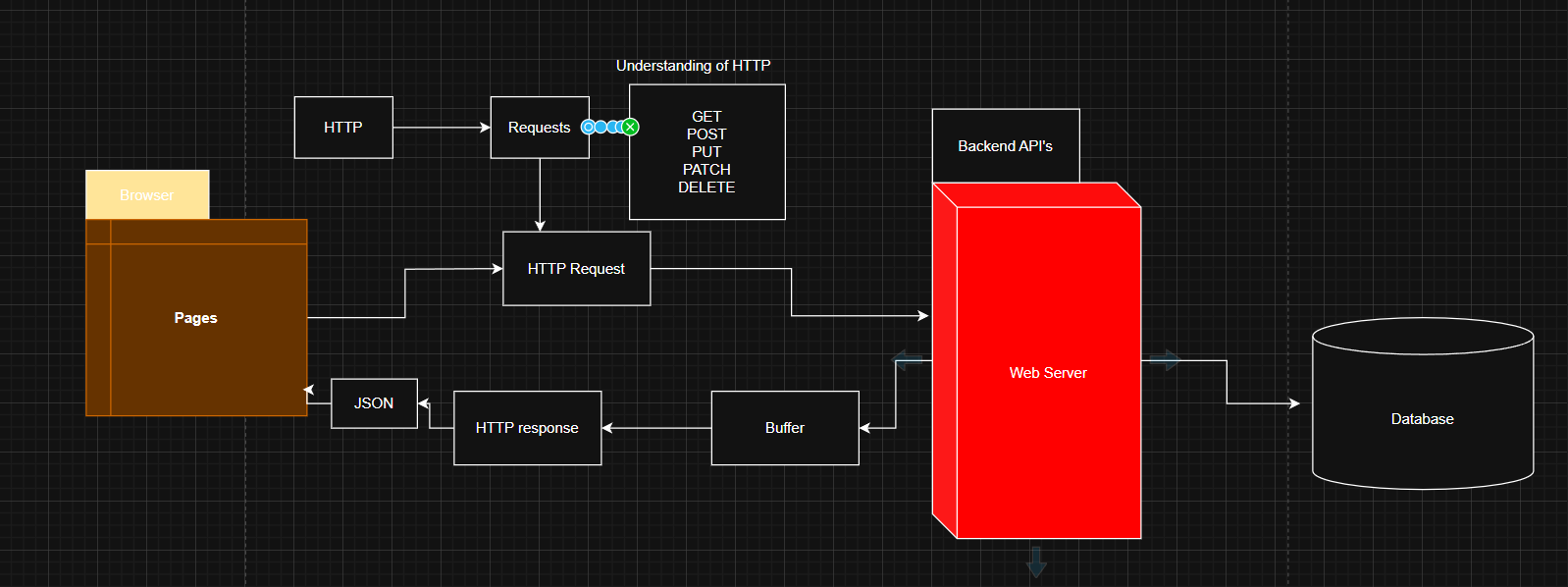
**React**

**Understanding of HTTP**

**What is HTTP**

Hypertext Transfer Protocol (HTTP) is an application layer protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers, but it can also be used for other purposes. HTTP follows a classical client server model with a client opening a connection to make a request, then waiting until it receives a response. HTTP is a stateless protocol. Meaning that the server does not keep any data (state) between two requests.

**HTTP GET Request**

The HTTP GET request method requests a representation of the specified resource. Requests using GET should only be used to request data (They should not include data).

📝 Sending body/payload in a get request may cause some existing implementations to reject the request while not prohibited by the specification, the semantics are undefined. It is better to just avoid sending payloads in GET request.

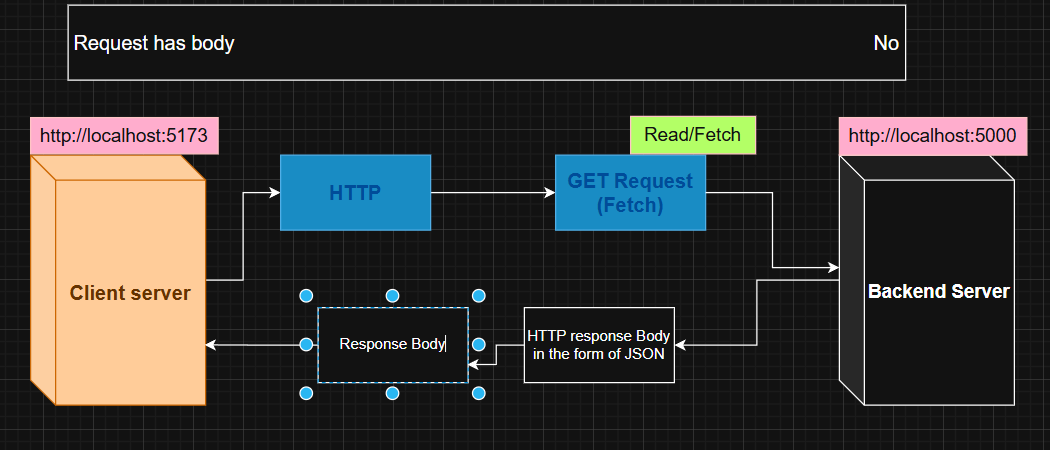
|  |  |
| --- | --- |
| **Request has body** | No |
| **Safe** | Yes |
| **Successful response has body** | Yes |
| **Idempotent** | Yes |
| **Cacheable** | Yes |
| **Allowed in HTML forms** | Yes |

Our get method does not have any body but on the successful response it will have body to get the data.

In terms of **safe** it want to say like it is read only so data won’t get updated by this. It is just reading the data from the server. It does not alter the state of the server.

Idempotent means if you are getting the same data then it is idempotent. Like if we try to fetch `api.github.com/users` we will be getting the same data. An HTTP method is idempotent if the intended effect on the server of making a single request is the same as the effect of making several identical requests.

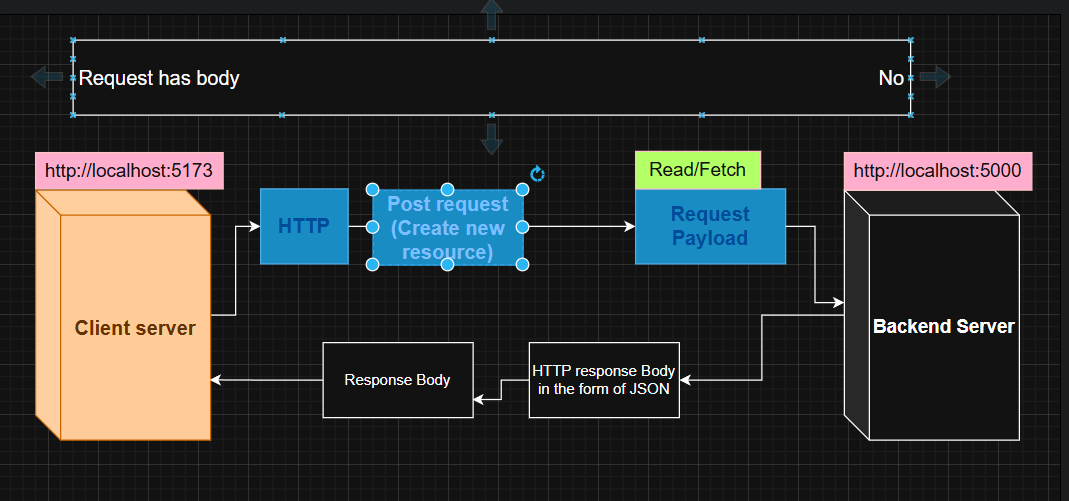
Cacheable response is an HTTP response that can be cached, that is stored to be retrieved and used later, saving a new request to the server. Not all HTTP responses can be cached, these are the following constraints for an HTTP response to be cached.

Allowed in HTML forms means there are two methods in HTML get and post.

**HTTP post request**

The HTTP post method sends data to the server. The type of the body of the request is indicated by the context type header.

The difference between put and post is idempotent: Calling it once or several times successively has the same effect, where successive identical post may have additional effects, like passing an order several times.

****A post request is typically sent via an HTML form and results in a change on the server. In this case, the content type is selected by putting the adequate string in the enctype attribute of the form element or the formenctype attribute of the input or button elements.

|  |  |
| --- | --- |
| **Request has body** | Yes |
| **Successful response has body** | Yes |
| **Safe** | No |
| **Idempotent** | NO |
| **Cacheable** | Only if freshness information is included |
| **Allowed in HTML forms** | Yes |

**Put Request**

It is similar to post request but we have slight difference here. If you want to modify the data then you need to use put request. It is modifying existing resource.

The HTTP put request method creates a new resource or replaces a representation of the target resource with the request payload.

The difference between put and post is that put is idempotent: calling it once or several times successively has the same effect (that is no side effect), whereas successive identical post requests may have additional effects, akin to placing an order several times.

|  |  |
| --- | --- |
| **Request has body** | Yes |
| **Successful response has body** | May |
| **Safe** | No |
| **Idempotent** | Yes |
| **Cacheable** | No |
| **Allowed in HTML forms** | No |

**Patch request**

It is not updating entire resources instead it updates particular field.

The HTTP patch request method applies partial modifications to a resource.

Patch is somewhat analogous to the update concept found in CRUD (in general HTTP is different than CRUD and the two should not be confused.)

A Patch request is considered a set of instructions on how to modify a resource. Contrast this with put which is a complete representation of a resource.

A patch is not necessarily idempotent, although it can be. Contrast this with put which is always idempotent. The word idempotent means that any number of repeated, identical requests will leave the resource in the same state. For example if an auto incrementing counter field is an integral part of the resource then a put will naturally overwrite it (since it overwrites everything), but not necessarily so for patch.

|  |  |
| --- | --- |
| **Request has body** | Yes |
| **Successful response has body** | May |
| **Safe** | No |
| **Idempotent** | No |
| **Cacheable** | Only if freshness information is included |
| **Allowed in HTML forms** | No |

**Delete request**

|  |  |
| --- | --- |
| **Request has body** | May |
| **Successful response has body** | May |
| **Safe** | No |
| **Idempotent** | Yes |
| **Cacheable** | No |
| **Allowed in HTML forms** | No |