Do you really know why you prefer REST over RPC?

When speaking about web-services, there is a little confusion about them. Remote-Procedure-Call (RPC), for example, is one style of creating web APIs, while Representational State Transfer (REST), is another approach. Each style has a separate implementation.

The debate of using JSON-RPC over REST is an interesting one, and the usual response is: “it depends”. Both of them are architectural styles for serving content remotely, using a client-server model.

A REST endpoint treats the request like making a call to a resource. The resource becomes the domain that has the data. The resource does not concern itself with functionality at all. It is only a place where you contain data and do with it as you see fit. What is nice about REST is that you have a resource API you can then call from many clients. REST API’s only concern is all the data that belongs to that specific domain.

RPC style endpoints are great when you want only one job done well. This makes it useful for one or two app clients because it is a niche service. RPC endpoints can implement business logic inside the service, given that it only does one thing. This adds simplicity and clarity to the service.

Here are a few benefits of RPC over REST:

* Only one method to send a request – usually POST for HTTP
* The content of the request is completely separated to the transmission mechanism. All errors, warnings, and data are in the payload of the request.
* The separation of message and transmission channel in a JSON-pure API results in an API that is usually faster, and almost always more reliable, easier to use, easier to port

However, you should use REST instead of RPC because:

* REST API is always independent of the type of platform or languages
* The separation between client and server has one evident advantage, and that is that each development team can scale the product without too much problem
* RPC does not provide caching when using mostly POST REQUESTS
* REST is more predictable than RPC as it relies on the shared semantic of HTTP verbs

References:

<https://sites.google.com/site/wagingguerillasoftware/rest-series/what-is-restful-rest-vs-rpc>

<https://www.linkedin.com/pulse/rest-vs-rpc-soa-showdown-joshua-hartman/>

<https://restfulapi.net/rest-architectural-constraints/>

<https://filipaivars.pt/2019/03/18/api-paradigms-and-when-to-use-them/>