Fiasal Hotaky

A05-23327516

Writing Section

There are three API architecture styles RPC, REST and GraphQL. These are all different tools for different situations. In each situation, one API architecture would have a better advantage/ benefit in using over the other two. Design considerations is the main thing that decides which API style is best for the project at hand. Some but not all design considerations are coupling, chattiness, client complexity, cognitive complexity, caching, discoverability, and versioning.

These three API architecture styles all compare and differ in many ways. The RPC pattern (Remote procedure call) the main purpose to use it is to call a function on another server. For example, functions that already exist in our system and put them on a transport like http which is then wrapped with a http endpoint. Situations that RPC are best for command oriented because with a function you are calling something on another system. An advantage of RPC API is it is very easy to add a new function. Companies such as Google, Netflix, Twitter all use RPC API internally with specific high-performance variants such as Apache Thrift and Twirp. Which provide high performance, low overhead messaging. Large internal micro server system that needs to communicate between micro servers need communication overhead to be small and clear. Advantages of RPC are lightweight payloads and low overhead. However, the downside to RPC pattern is that it is highly coupled to the underlying system. Which has the effect of easy to leak details of the underlying system. Designers worry about how underlying system functions are designed. In addition, the RPC pattern has low discoverability, not easy to know which function to call if you have not used it before. Furthermore, the term function explosion refers to when an underlying system big huge list of functions, makes it hard to understand and maintain.

Here is where the Rest pattern differs from the RPC pattern. RPC pattern is tightly coupled between clients and server and the goal of Rest is the decoupling of client and server. Rest solves the tightly coupling problem between the underlying system and the API by force client and server to be less coupled and have a stronger layer of extraction between them. Modeling resources instead of functions. Which is the opposite of what RPC pattern does. Another way in which Rest API differs from RPC API is that it is easier to understand and maintain without prior knowledge of the built API. You can do this by looking and interpret the requests giving by the API can understand how to use your API without any external documentation or prior knowledge. Rest API also reuses http aspects like caching semantics and http verbs. Some disadvantages are that the structure of Rest API is not uniform.

Lastly the API style GraphQL takes both RPC and Rest ideas and adds to them. Instead of modelling functions and resources GraphQL API deals with a query. GraphQL and RPC API both compare because they have a uniform structure unlike Rest API. Can tell the server exactly what you want like certain fields interested in and linked objects. Server would return exactly what you asked for. Which all three API’s compare in but have different ways of obtaining the information from the server. Graphql starts with strongly typed schema, description of All queriers you can make and types they return. Advantages low network overhead, typed schema, fits graph-like data very well

An example where I would use the GraphSql API rather than the others is building a mobile API. Such as a social media app where low network overhead is needed. Furthermore, GraphQL fits graph-like data very well for example a newsfeed. To further back up why I would use GraphQL for a mobile app Facebook also uses GraphQL. Facebook sends massive query to server and pulls all back in single request. Instead of tons of requests only uses one, benefit if you have spotty connection

An example of where I would use Rest API over RPC would be an App that is interacting with foreign software, the client has a new requirement/ use case which a need on the home page on the side of page it would have a section and that section would say here is a list of all of the conversations in the network. In the RPC API the designer would have to build a new method that would return that information. But in Rest API the client can do this because of abstraction available. In Rest API clients can access and interact in a way with the API that you cannot do with RPC. Furthermore, an advantage for API designer is you don’t have to predict every use case, with Rest API because designer decided which resources are expressed in API and built metadata that describes links between resources. Which allows the client to do whatever they want unlike in RPC API where you would have to need a function for every use case.