Megan Walker

Bellevue University

WEB 335 Introduction to NoSQL

Professor Krasso

April 4, 2023

Discussion 4.1 – Stateless vs. Stateful

Stateless APIs are designed to operate without any form of server-side state management. This means that each request is treated as an independent and separate transaction, with no memory or context of previous requests. Basically, the server does not maintain any session data between requests. Instead, all data is included with each individual request, allowing the server to process it independently. This makes stateless APIs better at handling large numbers of clients, because they don't need to keep track of a lot of information on the server side, which can slow things down (Fielding, 2000). This is why Stateless APIs are often used in scenarios where scalability is a concern because they can handle large numbers of clients without slowing down. A few examples of stateless API's are RESTful API's, OpenAPI, and GraphQL.

On the other hand, stateful APIs require the server to keep track of client data. This means that stateful APIs can offer a more personalized experience for the user, as they can remember previous interactions with the client. SOAP, TCP, and WebSockets are examples of stateful APIs. SOAP, in particular, is a protocol that was designed to enable communication between applications built with different languages and platforms. However, because SOAP is a protocol, it imposes built-in rules that make it more complex and slower than other APIs like REST. The reason it is more complex and a bit slower is because it has built-in compliance standards, such as security, atomicity, consistency, isolation, and durability (ACID), that ensure reliable database transactions, making it useful for enterprise scenarios.

References

REST vs. SOAP. (2019). Redhat.com. https://www.redhat.com/en/topics/integration/whats-the-difference-between-soap-rest

Verborgh, R., van Hooland, S., Cope, A. S., Chan, S., Mannens, E., & Van de Walle, R. (2015). The fallacy of the multi-API culture. *Journal of Documentation*, *71*(2), 233–252. https://doi.org/10.1108/jd-07-2013-0098