Cloud-Based Web Services

Assignment 1

Siobhan Stephens - 1800709
University of Wales Trinity Saint David

Abstract

This report compares and contrasts the key principles of REST, SOAP and GraphQL web services, along with discussing the benefits and drawbacks of each. In recent years, the interest and importance of web services has increased rapidly, and therefore, understanding the uses, benefits and drawbacks of each of these services is critical in deciding which is most suitable for a given use and system architecture.

Table of Contents

1. Introduction 3

2. Service Types 3

2.1. REST 3
2.2. GraphQL 3
2.3. SOAP 4

3. Comparison 4

3.1. Transport Layer 5
3.2. Performance 5
3.3. Data Types 5
3.4. Caching 6
3.5. Querying 6
3.6. Implementation 6
3.7. Security 6

4. Conclusion 7

5. References 8

1. Introduction

One of the difficulties with web applications is the application-integration, therefore the use of a web service is the most common way for application-to-application communication across an internet protocol backbone [1]. A web service is an intermediate software that allows interaction between the applications, that may be written in different languages and using varying technology and application architecture, enabling heterogeneous environments [2].

The purpose of this report is to compare the key principles of REST, SOAP and GraphQL web services, and discuss their key benefits and drawbacks based on aspects such as performance and querying.

2. Service Types

2.1. REST

REST is an acronym for Representational State Transfer, first introduced by Roy T. Fielding in his Ph.D. dissertation Architectural Styles and the Design of Network-Based Software Architectures [3]. It is a software architectural style that defines a set of constraints to be used for creating web services, with services that conform to this style called RESTful web services [4]. The REST architecture is modelled around the client/server architecture style, with requests and responses based on the transferring process of the resources and deals only with data structure and transport.

Designing RESTful web services requires conforming to multiple principles, defined by the Richardson Maturity Model, which describes 4 levels (0-3), in which level 3 designates a truly RESTful API [5]. As stated by França et al, the adoption of REST as the predominant method for building public APIs has obfuscated any other technology or approach in the recent years [5].

2.2. GraphQL

In 2016, Facebook released a specification for their GraphQL framework, which is an alternative to REST-based data access interfaces [6]. A fundamental component of the GraphQL framework is a query language to submit data retrieval requests to GraphQL-aware web servers. The syntax for GraphQL queries resemble the Javascript Object Notation (JSON) [7], [8]. GraphQL queries are written in terms of a professed schema which the queried Web server supports [6]. As described by Hartig et al, “informally, such a schema defines types of objects by specifying a set of so-called fields for which the objects may have values; the possible values can be restricted to a specific type of scalars or objects” [6]. Since its release, GraphQL has gained significant momentum and has been adopted by an increasing number of users [6].

2.3. SOAP

Simple Object Access Protocol (SOAP) is a protocol maintained by the W3C, used by web services to construct and comprehend the messages they exchange, by using a standard, well-understood message format [9]. SOAP represents a one-way stateless communication paradigm between nodes or devices [10]. It is an independent protocol, due to the fact that it is platform agnostic and responds equally on any platform or OS, owing to both XML and HTTP protocols [11]. The SOAP specification defines an XML-based standard message format, which contains a SOAP header, message metadata, body section [12].

3. Comparison

In order to best compare the three web services, a comparison table has been provided below.

Table 1. Table comparing REST, GraphQL and SOAP web services

KEY PRINCIPLE	REST	GRAPHQL	SOAP
PAYLOAD	Lightweight Payload [13]	Succinct Payload [10]	Verbose on both request and response [13]
DATA TYPE	XML, JSON [1]	JSON [10]	XML [13]
NESTED QUERIES SUPPORTED?	No, must make multiple requests [5]	Yes, designed for this purpose. [5]	No, must make multiple requests [11]
SPECIFY EXACT RESPONSE	No, must return all values for a given query or make multiple requests [5]	Yes, the client specifies the response values, therefore only required data is returned [5]	No, must return all values for a given query or make multiple requests [11]
DISTRIBUTED COMPUTING SUPPORTED?	No [14]	No [6]	Yes, designed with this in mind [14]
HTTP CACHING INBUILT	Yes [10]	No, must use a client [10]	No [1]
PERFORMANCE	Focus on reliability, however regarded as good performance [15]	Main priority of GraphQL, succinct payload, fast performance [16]	XML slower than JSON, verbose payload, 10x as many bits as REST resulting in suboptimal performance [13]
PLATFORM	Agnostic [13]	Agnostic [6]	Agnostic [13]
TRANSPORT-LAYER	HTTP only [13]	Agnostic, can work with HTTP, WebSocket, TCP. HTTP most common [10]	Agnostic. Supports SMTP, FTP, IBM's MQSeries, or Microsoft Message Queuing (MSMQ) [13]
INBUILT RETRY LOGIC	No, it is the client's responsibility [15]	No, it is the client's responsibility [15]	Built-in retry logic for failed communications [15]

SECURITY	HTTPS with SSL/TLS [17]	No inbuilt mechanism [10]	No inbuilt mechanism, WS-Security available [18]

3.1. Transport Layer

REST, GraphQL and SOAP are all platform and language agnostic, SOAP and GraphQL are also transport agnostic, allowing them to use different transport-layer protocols, while REST can only use HTTP [13]. A key benefit of the GraphQL service would be if the use of WebSocket or TCP is a requirement, while if SMTP, TFP or MSMQ is required, SOAP would be more beneficial. However, with the majority of uses being HTTP, the features below may have greater impact.

3.2. Performance

The performance of REST is regarded as very good, when compared with SOAP, which is described as being far too verbose both on the request and response, resulting in suboptimal performance due the large amount of data being transferred within the SOAP structure [13]. As a result of SOAP's heavy payload, it could also require more than ten times as many bytes compared to REST, resulting in high bandwidth consumption in comparison [13].

REST does not require formats such as headers to be included in the message. It parses JavaScript Object Notation (JSON), a lightweight format for storing and transporting data which is human readable and easy for computers to parse and use. JSON provides significant performance gains over XML and is estimated to parse up to one hundred times faster than XML [19].

Although, REST performance may vary based on use-case and implementation due to the requirement of REST where multiple requests must be made to get complex objects with nested attributes. A drawback is that REST implementations often encourage over-fetching, where information from the server and the response contains more information than is required, making it less efficient than GraphQL [5]. GraphQL allows specific entities to be requested by the client, along with allowing complex nested queries from a single request, and as such the response will always the be the smallest possible [10]. This data fetching approach greatly reduces network payload and increases application performance [5]. Additionally, queries using GraphQL can be used to access the properties of multiple resources and follow the references between them [6].

3.3. Data Types

One of the most significant differences is the response data types – SOAP allows only XML data, while REST provides flexibility to the type of data returned, including as XML or JSON [11], and GraphQL allows only JSON. Depending on the application requirement, being limited to JSON only may not be an issue, as it has performance benefits over XML and is the most widely used data format [12]. However, SOAP's limitation to XML is likely to be a drawback in many use cases as it is slow to parse and is less widely used. If flexibility is important for integration into different systems, REST has a distinct advantage by allowing both JSON and XML.

3.4. Caching

Caching eliminates the need to send full requests and responses in the cases where data has been cached from previous requests. REST has a key advantage over both GraphQL and SOAP in this respect, as it uses the HTTP caching mechanism. If caching is required with GraphQL, additional mechanisms such as Apollo Client can be implemented on the client side, with SOAP also requiring a similar mechanism [10].

3.5. Querying

The key benefit of using the GraphQL is its allowance for complex and nested queries with specific entities to be requested returning succinct, predictable responses. GraphQL queries not only access the properties of a single resource, but also follow the references between them [20]. In comparison, typical REST and SOAP APIs would require making multiple requests to gather similar results, also resulting in over-fetching [10].

3.6. Implementation

When making changes to services in SOAP web services, a complicated code change on the client side is often required, whereas REST and GraphQL service changes would not require any client-side code change [3], [21]. Additionally, GraphQL requires a learning curve due to its proprietary querying, however as it bears a resemblance to JSON, the curve should not be steep.

3.7. Security

Security is a major aspect of all web services, however it has been noted that GraphQL may facilitate denial-of-service attacks [20]. Mikula et al states that "a GraphQL server can be attacked with an excessive number of complex queries that may to consume all server resources" [10]. Therefore, additional precaution must be taken to avoid denial-of-service attacks, such as performing a cost analysis of the query and imposing a threshold on the amount of data a request can consume, or by implementing a timeout in which the requests taking too long to resolve are removed from the execution queue [10]. Further to this, there is no inbuilt mechanism for authentication and authorization, and the resolvers typical to GraphQL make it complicated for developers to implement thoroughly. As a result, it could be said that security is a drawback of GraphQL.

REST web services may be protected through the use of SSL/TLS [17], with secure REST services using HTTPS endpoints, protecting data in transit. It allows clients to authenticate and provides integrity of the data [18]. Alternatively, SOAP comes with a well-elaborated security framework, WS-Security, which enforces integrity and confidentiality on messages and allows the communication of various security token formats [18].

4. Conclusion

In conclusion, the three web services discussed share many similarities but also many differences. REST continues to be the most widely used service, however, due to the increasing complexity of modern APIs, communication via the REST protocol is often not viable, due to the execution of multiple requests to get complex objects with nested attributes. If performance or complex querying is the critical aspect, GraphQL has a clear benefit, whereas if caching, data types or implementation are important, REST may be best suited. SOAP's key benefit is its transport layer options along with some security advantages.

From the given comparison and overall features, benefits and drawbacks, RESTful web services are the most balanced thanks to their ease of use, support for data types and lightweight implementation, along with widespread use and therefore support for developers.

5. References

[1] F. Halili and E. Ramadani, "Web Services: A Comparison of Soap and Rest Services," *Modern Applied Science*, vol. 12, no. 3, p. 175, Feb. 2018, doi: 10.5539/mas.v12n3p175.

[2] "View of Review of Web Service Technologies: REST over SOAP." http://qu.edu.iq/journalcm/index.php/journalcm/article/view/715/533 (accessed Dec. 30, 2020).

[3] H. Paik *et al.*, "Web Services – REST or Restful Services," in *Web Service Implementation and Composition Techniques*, Springer International Publishing, 2017, pp. 67–91.

[4] "Web Services Architecture." https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest (accessed Dec. 15, 2020).

[5] M. Diego, C. França, and E. da Silva, "Performance Evaluation of REST and GraphQL APIs Searching Nested Objects." Accessed: Dec. 24, 2020. [Online].

[6] O. Hartig and J. Pérez, "An Initial Analysis of Facebook's GraphQL Language Linked Data Fragments View project An Initial Analysis of Facebook's GraphQL Language," 2017. Accessed: Dec. 21, 2020. [Online]. Available: http://graphql.org/code/.

[7] International ECMA, "The JSON Data Interchange Syntax," 2017. Accessed: Dec. 18, 2020. [Online].

[8] "GraphQL." http://spec.graphql.org/June2018/ (accessed Dec. 18, 2020).

[9] F. Halili and E. Ramadani, "Web Services: A Comparison of Soap and Rest Services," *Modern Applied Science*, vol. 12, no. 3, p. 175, Feb. 2018, doi: 10.5539/mas.v12n3p175.

[10] M. Mikuła and M. Dzieńkowski, "Comparison of REST and GraphQL web technology performance," *Journal of Computer Sciences Institute*, vol. 16, pp. 309–316, Sep. 2020, doi: 10.35784/jcsi.2077.

[11] A. Šimec and G. T. Lidija, "Analysis of developers choices for API Rest or Soap protocols," *International Journal of Circuits, Systems and Signal Processing*, vol. 11, no. 2017, pp. 178–183, 2017, Accessed: Dec. 24, 2020. [Online]. Available: https://www.bib.irb.hr/902324?rad=902324.

[12] M. Seabra, M. F. Nazário, and G. Pinto, "REST or GraphQL? A performance comparative study," in *ACM International Conference Proceeding Series*, Sep. 2019, pp. 123–132, doi: 10.1145/3357141.3357149.

[13] S. H. Toman, "Review of Web Service Technologies: REST over SOAP," *Journal of Al-Qadisiyah for computer science and mathematics*, vol. 12, no. 4, p. Page 18 – 30-Page 18 – 30, Nov. 2020, doi: 10.29304/JQCM.2020.12.4.715.

[14] J. Sayago Heredia, E. Flores-García, and A. R. Solano, "Comparative Analysis Between Standards Oriented to Web Services: SOAP, REST and GRAPHQL," in *Communications in Computer and Information Science*, Dec. 2020, vol. 1193 CCIS, pp. 286–300, doi: 10.1007/978-3-030-42517-3_22.

[15] J. Sayago Heredia, E. Flores-García, and A. R. Solano, "Comparative Analysis Between Standards Oriented to Web Services: SOAP, REST and GRAPHQL," in *Communications in Computer and Information Science*, Dec. 2020, vol. 1193 CCIS, pp. 286–300, doi: 10.1007/978-3-030-42517-3_22.

[16] M. Cederlund, "Performance of frameworks for declarative data fetching An evaluation of Falcor and Relay+GraphQL," 2016. Accessed: Dec. 20, 2020. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-196058.

[17] V. Atlidakis, P. Godefroid, and M. Polishchuk, "Checking Security Properties of Cloud Service REST APIs," in *Proceedings - 2020 IEEE 13th International Conference on Software Testing, Verification and Validation, ICST 2020*, Oct. 2020, pp. 387–397, doi: 10.1109/ICST46399.2020.00046.

[18] L. lo Iacono, H. Nguyen, and P. Gorski, “On the Need for a General REST-Security Framework,” *Future Internet*, vol. 11, no. 3, p. 56, Feb. 2019, doi: 10.3390/fi11030056.

[19] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta, “Comparison of JSON and XML Data Interchange Formats: A Case Study.” Accessed: Dec. 19, 2020. [Online].

[20] M. Seabra, M. F. Nazário, and G. Pinto, “REST or GraphQL? A performance comparative study,” in *ACM International Conference Proceeding Series*, Sep. 2019, pp. 123–132, doi: 10.1145/3357141.3357149.

[21] H. Y. Paik, A. L. Lemos, M. C. Barukh, B. Benatallah, and A. Natarajan, *Web service implementation and composition techniques*. Springer International Publishing, 2017.