
GraphQL Report & Documentation

Tudor Haba
2420645

Video - <https://youtu.be/SwHS6urlkKI>

Documentation

To start the application first launch the RESTWebServiceApplication by clicking the run button in IntelliJ. Then cd into the Server folder within the ReactApp and start the server using 'npm start'. This will start the application on port 8081. To access the application, go to <http://localhost:8081> or follow the link from the terminal.

The web service makes use of the generated REST CRUD operations as well as implementing more complex queries such as the partial search query that looks for partial matches of license plates. DTOs were also added for every entity to display the relationships they have with the other entities on the frontend.

A CORS Config file was added that allows the communication between the web service hosted on port 8080 and the react front end that's hosted on 8081. Within the file all basic HTTP requests are allowed.

GraphQL

GraphQL is a query language for your APIs designed with the intention of simplifying the interactions between a web service and the client and other common problems that come with the industry standard of REST. It was originally made by Facebook in 2012 and later on made open source in 2015 [1]. GraphQL defines an entity within the database and unlike REST it can follow the relationships it has with other entities and bring the data forward in the response. In the application, to get the same functionality that GraphQL provides, DTOs had to be created for every entity. The DTO is defined, then the data is flattened and transformed into a DTO in order to produce the same output that GraphQL would. In REST it is required to make multiple queries to the database to retrieve the relationships between the entity and other entities from the database. In other words, it helps developers by saving them the time to write their own implementation of the entities [2].

Benefits of GraphQL

Compared to other, traditional query languages, like REST, GraphQL performance and response time is faster. Findings from [3] have found that for simple queries the performance is not much faster only by 0.02 times. However, as the complexity of the queries increased this is where the performance increase can be seen. For queries that interact with four endpoint the performance is increased by 16 times and for over 5 endpoints the performance is increased by up to 187 times. There is a complex query in the application which is used to purchase a plate that has to interact with all the endpoints. If taken into consideration, although only involving three endpoints in total,

then there could have been a performance boost from using GraphQL that could be noticeable in a real application where there are thousands of records in the database.

Another benefit of using GraphQL is its ability to simplify versioning APIs. While REST requires to version every change in an API, with GraphQL, it eliminates the versioning aspect from APIs improving the maintainability of the application [3]. What this means is that when developers decide they want to add more data to be returned in a query, they can easily do so by changing the schema instead of making separate versions and fixing the queries to return the correct entities. This also makes the APIs more reusable, because there is a single endpoint that can adapt to different needs without requiring any changes. This means that the frontend and backend services need less coordination between themselves, as frontend developers can request the data for additional features without having to wait for it to be implemented [3].

The biggest benefit remains the fact that in GraphQL the developers specify the data they want, and the response will only contain the specified data. This solves some of the over-fetching problems encountered before by REST where more data would be brought in the response than needed. This would mean that the unused data would get discarded after it was already pulled from the database adding unnecessary performance overheads to the application. This improves the overall efficiency of the application by minimizing the amount of data that's being fetched. This is especially important for scalability benefits, and it is the reason sites such as Facebook have been using it, as the amount of data in a single interaction could be massive. Reducing the amount of unnecessary data fetched contributes to improving the performance and benefits the users [4].

Scenario

A good scenario for using GraphQL would be the purchase log implemented in the web service. The purchase log has to communicate with both the account database to retrieve details about the user that just made the purchase and to communicate with the license plate database to get the details of the plate that has been purchased. To view a purchase log in REST you would have to interact with 3 separate databases.

This is how a query for the scenario described above would look like in GraphQL:

```
query purchaseLogDetails {  
  getPurchaseLog(id: 1) {  
    id  
    purchaseDate  
    user {  
      username  
      name  
    }  
    licensePlate {  
      licensePlateNumber  
      price  
    }  
  }  
}
```

References

- [1] S. L. Vadlamani, B. Emdon, J. Arts, and O. Baysal, 'Can GraphQL Replace REST? A Study of Their Efficiency and Viability', in *2021 IEEE/ACM 8th International Workshop on Software Engineering Research and Industrial Practice (SER&IP)*, Jun. 2021, pp. 10–17. doi: 10.1109/SER-IP52554.2021.00009.
- [2] W. Guedes, 'How to succeed with GraphQL APIs', *InfoWorld.com*, Aug. 2023, Accessed: May 11, 2025. [Online]. Available: <https://www.proquest.com/docview/2858387507/abstract/E6B1C0F615594B91PQ/1>
- [3] A. Quiña-Mera, P. Fernandez, J. M. García, and A. Ruiz-Cortés, 'GraphQL: A Systematic Mapping Study', *ACM Comput. Surv.*, vol. 55, no. 10, pp. 1–35, Oct. 2023, doi: 10.1145/3561818.
- [4] A. Lawi, B. L. E. Panggabean, and T. Yoshida, 'Evaluating GraphQL and REST API Services Performance in a Massive and Intensive Accessible Information System', *Computers*, vol. 10, no. 11, Art. no. 11, Nov. 2021, doi: 10.3390/computers10110138.