LONTAR KOMPUTER VOL. 11, NO. 2 AUGUST 2020 p-ISSN 2088-1541
DOI : 10.24843/LKJITI.2020.v11.i02.p05 e-ISSN 2541-5832
Accredited B by RISTEKDIKTI Decree No. 51/E/KPT/2017

Graph-QL Responsibility Analysis at Integrated Competency Certification Test System Base on Web Service

I Gede Susrama Mas Diyasa[a1], Gideon Setya Budiwitjaksono[b2], Hafidz Amarul M[a3], Ilham Ade Widya Sampurno[a4], Ni Made Ika Marini Mandenni[c5]

[a]Department of Informatic, University of Pembangunan Nasional "Veteran Jawa Timur
Jl. Rungkut Madya Surabaya, Indonesia
[1]igsusrama.if@upnjatim.ac.id, [3]hafidzamarul@gmail.com
[4]ilhamade@gmail.com

[b]Department of Accounting, University of Pembangunan Nasional "Veteran Jawa Timur
Jl. Rungkut Madya Surabaya, Indonesia
[2]gideon.ak@upnjatim.ac.id

[c]Department of Information Technology, Udayana University
Jl. Raya Kampus Unud Bukit Jimbaran, Bali, Indonesia
[5]made_ikamarini@unud.ac.id

Abstract

Graph-QL (Query Language) is a new concept in the Application Programming Interface (API). Graph-QL was developed by Facebook which is implemented on the server-side. Although it is a query language, Graph-QL is not directly related to the database, in other words, Graph-QL is not limited to certain databases, either SQL or NoSQL. The position of Graph-QL is on the client and server-side that access an API. One of the objectives of developing this query language is to facilitate data communication between the backend and frontend / mobile applications. For this reason, this paper will examine the responsibility of Graph-QL in terms of response time and response size in the development of an integrated competency certification test system based on web service and compared with efficiency and flexibility using the REST API. From the test results, it was found that Graph-QL provided some advantages compare to REST API. It give more flexibility for the clients to access the data and solve the most typical problem that was over or under fetching cause by fixed data given by REST API endpoints.

Keywords: *Graph-QL, Rest, Responsibility, Analysis, Web Service*

1. Introduction

Graph-QL is a server-side query and runtime language for Application Programming Interfaces (API) that prioritizes giving clients data exactly what they request [1]. In essence, Graph-QL is a language for querying databases from client-side applications [2]. On the backend, Graph-QL can specify to the API how the data is presented to the client. It is also designed to make APIs faster, more flexible and developer friendly [3]. As a REST alternative, Graph-QL allows developers to make requests that pull data from multiple data sources in a single APIs endpoint [4]. In addition, the API manager will also have the flexibility to be able add and remove fields without having affect to existing queries [5]. Developers can also build APIs by any method they want. To prove that Graph-QL has a fairly good responsibility, this paper is implemented in the manufacture and testing of an integrated competency certification test system, and compared using the REST API [6].

Some previous studies related to Graph-QL, among others in reference [7] is analyzing the calculation of performance of Graph-QL and RESTful technology in the web information service system of the Institute for Research and Community Service Hasanuddin University [7]. The performance parameters used are Response Time and Throughput, with RESTful speeds still superior to Graph-QL because RESTful speeds are consistently stable in terms of access time and data size. Whereas Graph-QL is dynamic because it can change depending on demand fluctuations.

Another researcher [8] showed in his research that Graph-QL can reduce the size of the JSON document returned by REST APIs in 94% (in the number of fields) and 99% (in the number of bytes), both of which are median results, but the dataset is used in this paper includes gray literature articles, migrated system source code, and queries used in runtime evaluation publicly available at https://github.com/gleisonbt/migrating-to-graphql [8].

Another case with another researcher, in reference [9], where the purpose stated in his papers to understand the properties of language in the Facebook initial, by providing semantic formal queries [9]. After that, language analysis is performed and shows that the language has very low complexity for evaluation. This paper only compares the Graph-QL request language with the classic request language, which is the acyclic conjunctive query language (ACQ). Research on Advanced Data Retrieval with Graph-QL: a case study in Case Bakery Services, where this paper also studies and compares two data collection approaches. REST and Graph-QL in the context of case studies for web applications (Bakery Service applications), however, this paper does not consider aspects such as caching, mutation, and security [10].

From some of the research above, it focuses on comparing performance of the Graph-QL with the REST API based on aspects of Mutation, Query, and Type before using Data Manipulation Language (DML) [11], whereas in the paper presented here, it has novelty, which is about responsibility analysis of Graph-QL on the response time and response size, and also comparing with REST in making an integrated system of "competency certification test" based on web services which based on time and size of the response, and compare it with REST, and also focuses on several aspects including Mutation which is an operation that involves changes in the database, Query is an operation to take data in the database, and Types are almost the same like classes in programming languages, and include aspects such as caching, mutation, and security [12].

The steps for testing the responsibility of Graph-QL, first is the initial step to create an integrated system of "competency certification testing" based on web services and build API REST and Graph-QL APIs [13], then test each of the APIs above. The goal is the Graph-QL approach can be set any conditions or data needed by a query in the manufacture of a system so that all data as needed without additional information that is not needed. With hope that using Graph-QL will be more efficient and flexible to get data [14].

2. Reseach Methods

The research began by building an integrated system of competency tests then input the data into the database. This system will be built with 2 concepts in the API, Graph-QL and REST [15], the data will be used as output requests from clients, then test and comparing the performance of each API concept that has been built using the characteristics of QoS (Quality of Service). As shown in Figure 1. Graph-QL test system design and REST APIs [16] [17].

The experiment was carried out, namely conducting an experiment to access the API endpoint of Graph-QL and REST which has been applied to the integrated competency test system. The factors that will be used as a comparison in this study are the speed, size, and effectiveness of the response from Graph-QL and REST.

2.1. Building an information system Competency test

A competency test is a process of assessment both technical and non-technical through the collection of a relevant test to determine whether a person is competent or not yet competent in a Competency Unit or certain qualifications.

The competency test system is built based on problems in data processing and distribution in the competency certification process at a professional certification Institution [18]. This system has 3 role users namely admin, assessor, and assessee (competency test participants). In this system, the assessment has the role to register with the system, fill in the APL 1 form, register professional certification, and fill in the APL-2 Self Assessment form. The assessor has the role to verify the registration of assesse, check the assessee's Self Assessments APL-2 form, conduct the assessment, fill in the observation form, portfolio, and decide the results of the assessee graduation with the assessment record form. Admin in this system has the role to manage the competency scheme data, manage the competency test data site, select assessors for each assessee who register, manage the assessor user, verify the registration of the assessment and create a competency test schedule.

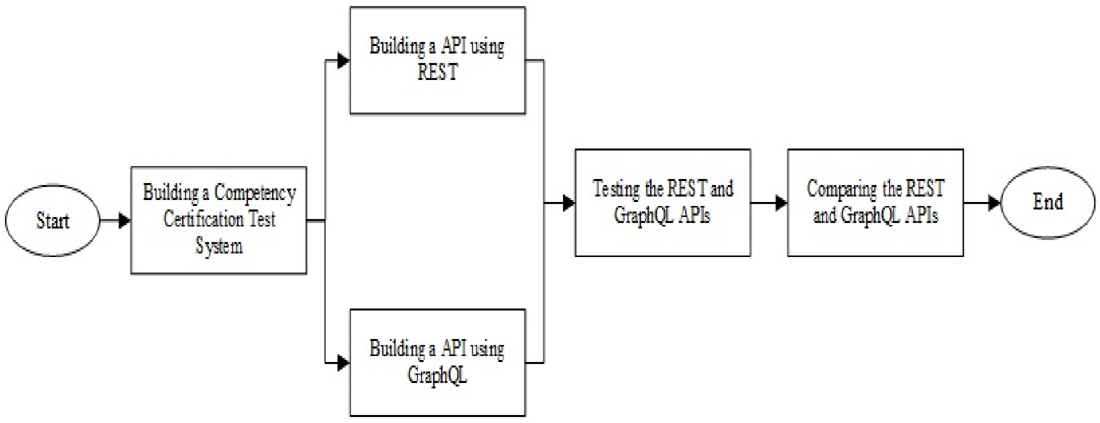

Start
Building a Competency Certification Test System
Building a API using REST
Building a API using GraphQL
Testing the REST and GraphQL APIs
Comparing the REST and GraphQL APIs
End

Figure 1. Design of Graph-QL test systems and REST API

This system is built using an API [19] that allows for multiplatform system development. The fire concept used is Graph-QL which consists of 24 types of objects, 52 mutations, and 44 queries. The tools used to build this system are using the PHP 7 programming language and the Lumen framework [20]. The database management system (DBMS) used is MariaDB 10.4.6. The web server used to run the competency test information system is Apache 2.4.41 on the local server [21].

2.2. Building an API with Graph-QL

In the Graph-QL implementation using the Lumen framework and Lighthouse library the architectural pattern used is in Figure 2, which explains the system architecture used in the Graph-QL API. Requests will be accepted by the server and will be checked on Graph-QL Schema [22]. Then the request is continued to the resolver which uses a model to access the database. The results of the requested data will be issued with a JSON data type [22].

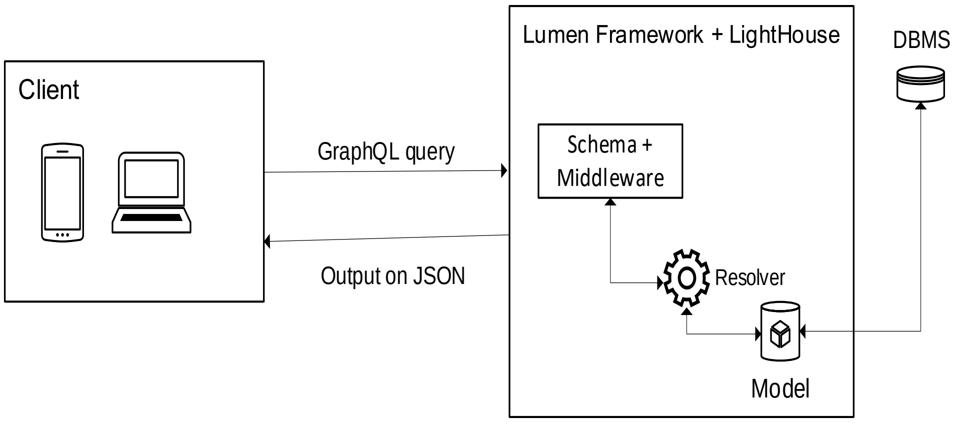


Figure 2. Graph-QL Architecture

p-ISSN 2088-1541
e-ISSN 2541-5832

2.3 Building an API with REST

In the implementation of REST using the Lumen framework the pattern used is in Figure 3. Figure 3 is an architecture in implementing REST on the Lumen framework. Requests will be accepted by the server and forwarded to Route [21]. The route will be then connected to a Controller that uses a model to access the database. The results of the requested data will be issued with a JSON data type [21].

2.4 Test Design of the REST API and Graph-QL

A trial was conducted to find out the functionality of the API with the concept of REST and Graph-QL can run as desired or not. The trial was carried out using the Postman application, which is an application commonly used to make HTTP requests on the server. Tests are carried out on functions that have the same data output between REST and Graph-QL [22].

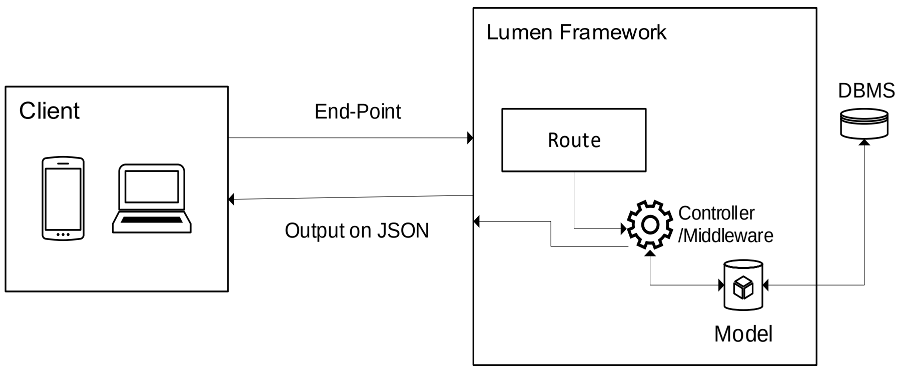


Figure 3. REST Architecture

3. Result and Discussion

The use of an API in an information system is a bridge between systems built on different platforms so that the information system has one central server and database storage. Graph-QL was present in 2015, according to the developer Graph-QL is easier to implement in an information system and can reduce the number of requests on the server and have an impact on reducing network traffic on the server.

3.1. Data retrieval using Graph-QL

Before performing data retrieval, the Schema of Graph-QL must first define all the attributes of the database tables that are needed as output from requests received. An example of defining a Graph-QL Schema on an object and query is in Figure 4 (a).

Figure 4. (a) shows the code used to define an object named Schema and the Query Graph-QL code used to retrieve 1 schema data in the Lumen framework using the Lighthouse library. To access the Query that was created in Figure 4. (a) using the code as in Figure 4. (b). Figure 4. (b) shows the code used in querying Graph-QL with the writing format used is JSON. The results of the query are in Figure 5.

```
type Skema {
    id : ID!
    kode : String!
    skema : String!
    kategori : String
    bidang : String
    mea : String
    panduan : String
    flag : Int!
    created_at : String
    updated_at : String

    unitKompetensi : [UnitKompetensi] @hasMany
}

Type Query {
    skema(id: ID! @eq) : Skema @find
}
```

(a)

```
query{
    skema(id:11){
      id
      kode
      skema
      kategori
      bidang
      mea
      panduan
      flag
      created_at
      updated_at
      unitKompetensi{
        id
        unit
      }
    }
}
```

(b)

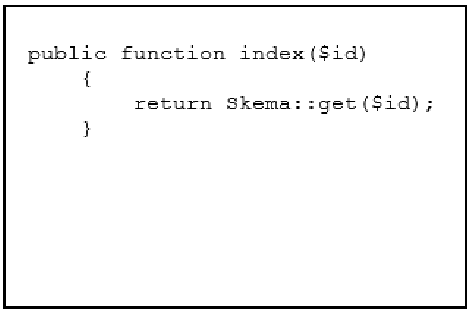
Figure 4. (a) Graph-QL Schema, (b) Graph-QL Query

```
{
  "data": {
    "skema": {
      "id": "11",
      "kode": "SS.LSPUPNVJT.FIK.01.1",
      "skema": "Junior Web Programmer",
      "kategori": "Aktivitas Jasa Lainnya",
      "bidang": "Aktivitas Jasa Perorangan Lainnya",
      "mea": "None Mea",
      "panduan": null,
      "flag": 1,
      "created_at": "2020-02-19 08:01:44",
      "updated_at": "2020-02-19 08:01:44",
      "unitKompetensi": [
        {
          "id": "1",
          "unit": "Menganalisis Tools"
        },
        {
          "id": "2",
          "unit": "Menganalisis Skalabilitas Perangkat Lunak"
        },
        {
          "id": "3",
          "unit": "Melakukan Identifikasi Library, Komponen atau
Framework yang Diperlukan "
        },
```

Figure 5. Graph-QL Query Results

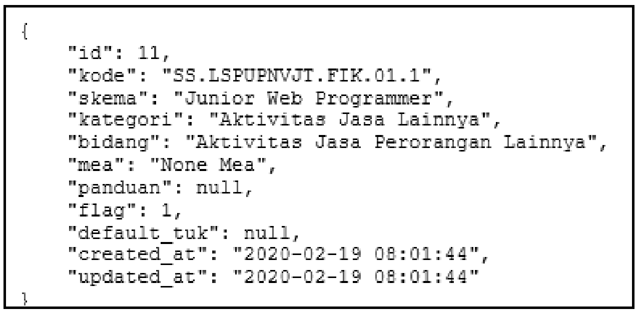
3.2. Retrieving data using REST

Using REST requires a function in the controller class that handles requests from clients. The function code for retrieving the schema data is shown in Figure 6 (a). Namely a function that is used to retrieve 1 schema data according to id. The result of this function is in Figure 6. (b), is the result of REST, the writing format used is the same, JSON.



```
public function index($id)
    {
        return Skema::get($id);
    }
```

(a)



```
{
    "id": 11,
    "kode": "SS.LSPUPNVJT.FIK.01.1",
    "skema": "Junior Web Programmer",
    "kategori": "Aktivitas Jasa Lainnya",
    "bidang": "Aktivitas Jasa Perorangan Lainnya",
    "mea": "None Mea",
    "panduan": null,
    "flag": 1,
    "default_tuk": null,
    "created_at": "2020-02-19 08:01:44",
    "updated_at": "2020-02-19 08:01:44"
}
```

(b)

Figure 6. (a) Schema Function, (b). REST Result

3.3 Response time comparison results

In comparing the response time of the API implementation with the concept of Graph-QL and REST, 20 experiments were carried out on each concept. The results of the response time comparison between Graph-QL and REST are shown in Figure 7. The time displayed has units of millisecond (ms)

Figure 7. Shows the results of the response time comparison between REST and Graph-QL. The result shows that the REST response time is faster than Graph-QL. The average response time of REST is 125.35ms while the average response time of Graph-QL is 262.15. In response time testing is carried out on the process of fetching the schema data that is on the system. Only 25 data are available. From the results depicted in Figure 7, it will be different for each process in the system. Figure 2 shows that in terms of speed, REST is faster than Graph-QL.

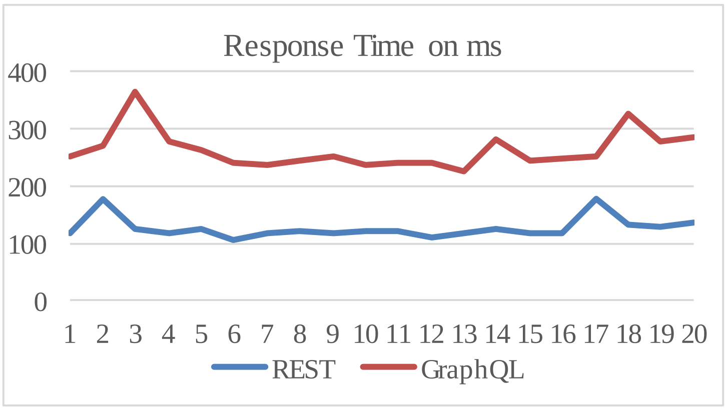
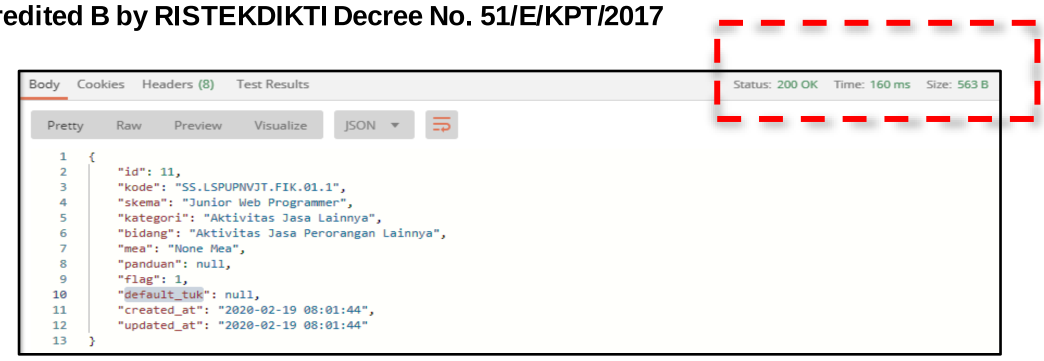


Figure 7. Results of Comparison of REST and Graph-QL response times

3.4 Response size comparison results

In comparing the response sizes of the API Implementation with the Graph-QL and REST concepts an experiment was conducted with the same required output goals. The results of testing the REST response size are in Figure 8.

In Figure 10 it is explained that the response size of requests to the API with REST of 563 Bit is taken in 160ms. While the results of the Graph-QL response size test in Figure 9.



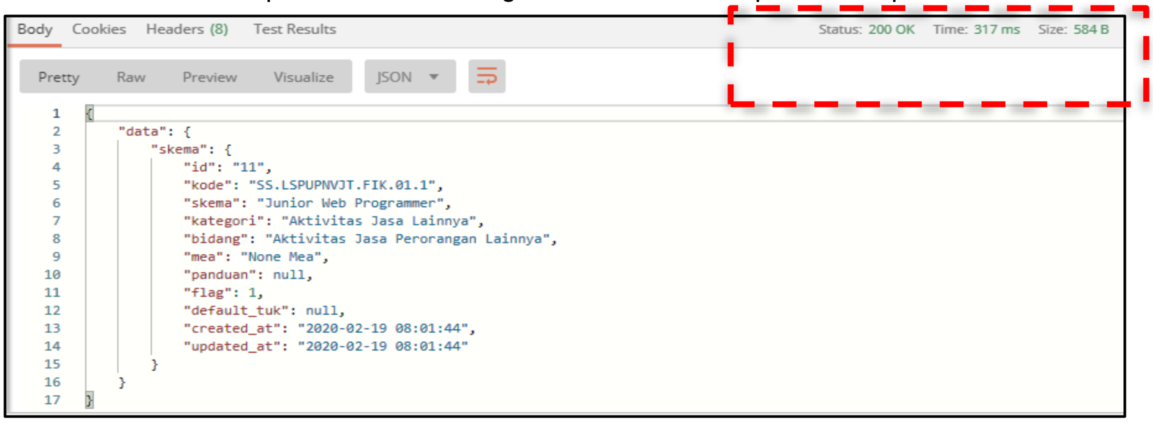
```
{
    "id": 11,
    "kode": "SS.LSPUPNVJT.FIK.01.1",
    "skema": "Junior Web Programmer",
    "kategori": "Aktivitas Jasa Lainnya",
    "bidang": "Aktivitas Jasa Perorangan Lainnya",
    "mea": "None Mea",
    "panduan": null,
    "flag": 1,
    "default_tuk": null,
    "created_at": "2020-02-19 08:01:44",
    "updated_at": "2020-02-19 08:01:44"
}
```

Figure 8. REST response size results

In Figure 9 explains that the response size of requests to the API with Graph-QL of 584 Bit reached with 317 ms. these results are greater than the REST response size. But Graph-QL has dynamic properties that can be adjusted to the needs so that if the data requirements are less than the data in Figure 9, the response size of Graph-QL is smaller.

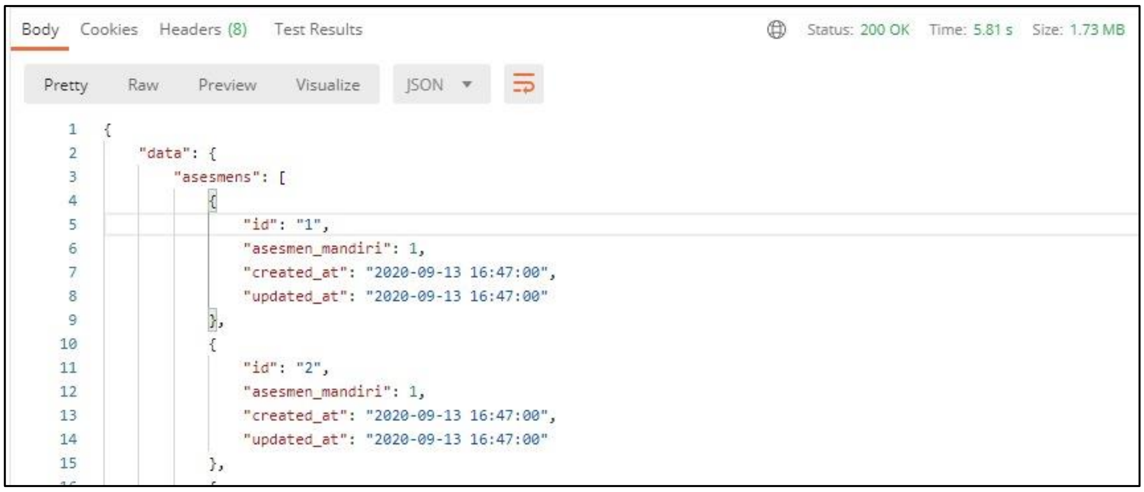
3.4 Comparisons With Large Data

Another experiment carried out to test the size or speed was to collect 17,329 lines of data with the same code as the process for retrieving schema data in the previous experiment.



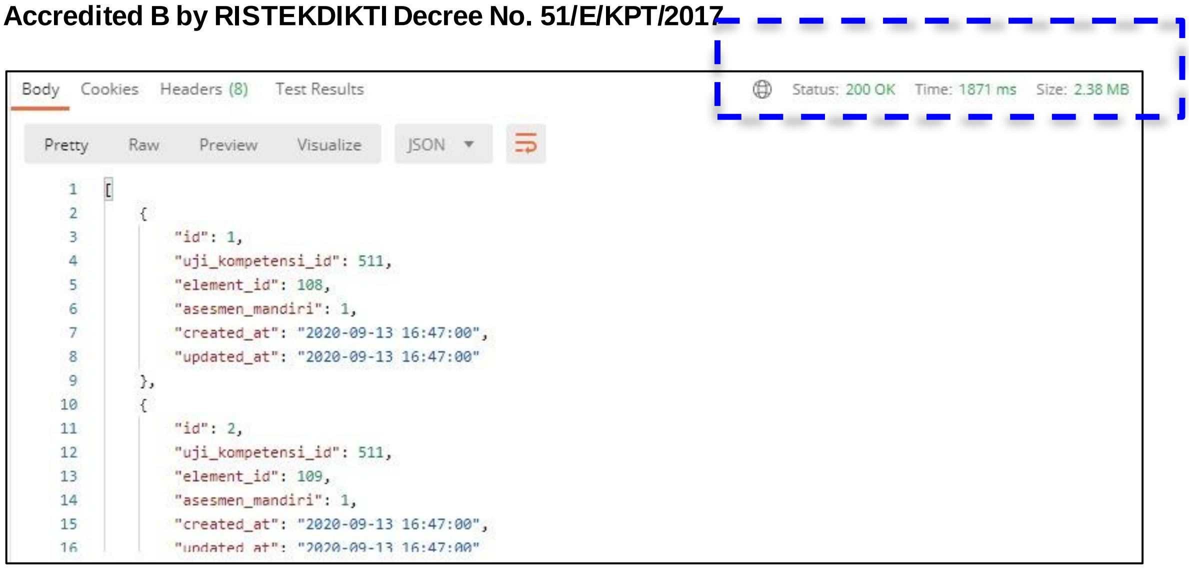
```
{
    "data": {
        "skema": {
            "id": "11",
            "kode": "SS.LSPUPNVJT.FIK.01.1",
            "skema": "Junior Web Programmer",
            "kategori": "Aktivitas Jasa Lainnya",
            "bidang": "Aktivitas Jasa Perorangan Lainnya",
            "mea": "None Mea",
            "panduan": null,
            "flag": 1,
            "default_tuk": null,
            "created_at": "2020-02-19 08:01:44",
            "updated_at": "2020-02-19 08:01:44"
        }
    }
}
```

Figure 9. REST response size results



```
{
    "data": {
        "asesmens": [
            {
                "id": "1",
                "asesmen_mandiri": 1,
                "created_at": "2020-09-13 16:47:00",
                "updated_at": "2020-09-13 16:47:00"
            },
            {
                "id": "2",
                "asesmen_mandiri": 1,
                "created_at": "2020-09-13 16:47:00",
                "updated_at": "2020-09-13 16:47:00"
            },
```

Figure 10. Graph-QL response results in asesmens



Status: 200 OK Time: 1871 ms Size: 2.38 MB

```
[
    {
        "id": 1,
        "uji_kompetensi_id": 511,
        "element_id": 108,
        "asesmen_mandiri": 1,
        "created_at": "2020-09-13 16:47:00",
        "updated_at": "2020-09-13 16:47:00"
    },
    {
        "id": 2,
        "uji_kompetensi_id": 511,
        "element_id": 109,
        "asesmen_mandiri": 1,
        "created_at": "2020-09-13 16:47:00",
        "updated_at": "2020-09-13 16:47:00"
```

Figure 11. REST response results in assessments

Figure 10 shows the results of the data retrieval process with 17,329 rows of independent assessment data on the system using Graph-QL, while Figure 11 shows the data retrieval process using REST. The results of the experiments conducted with 17,329 rows of data in terms of REST speed were superior to Graph-QL with a ratio of 1: 5, but in terms of data size, Graph-QL was lighter than REST due to its effectiveness in data retrieval with Graph-QL.

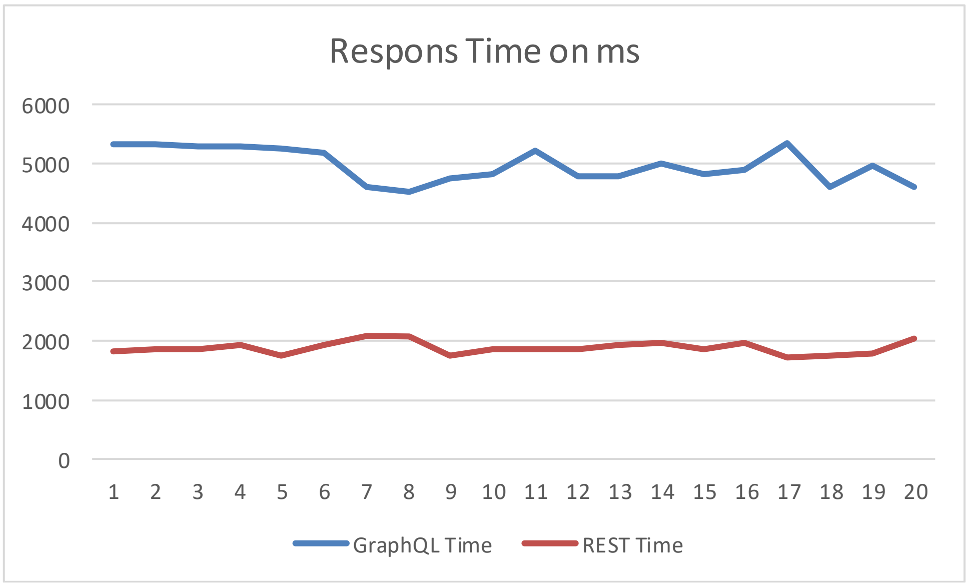


Figure 12. REST response results in asesmens

Figure 12 shows the results of 20 experiments for requests for the process of taking independent assessment data on a system with 17329 rows of data, showing that the request time with REST is faster than Graph-QL. This is still the same as previous experiments on 25 schema data with REST and Graph-QL.

4. Conclusion

From the results of the research conducted and explained, REST and Graph-QL can be implemented in the competency test information system. It shows that REST is superior in terms of time and response size if the data needed is the same compared to Graph-QL. However, REST is static, which means that the output results are in accordance with what is written in the function code, it can cause under data fetching or over data fetching. Graph-QL that takes a more dynamic approach, the output results can be modified to reduce the attributes as needed or to retrieve data in the classes that are related to the requested class, which results in, reduced demand from clients. The response size of Graph-QL will accord with the data needed. For this reason, this system is bettered suited if the API is implemented with Graph-QL because each function on the client system will have different data requirements.

References

[1] Mondaca, F., Schildkamp, P., & Rau, F. “Introducing KOSH, a framework for creating and maintaining APIs for lexical data”. *Proceedings of Electronic Lexicography in the 21st Century Conference*, *2019-October*, 2019, 907–921.

[2] Brito, G., Mombach, T., & Valente, M. T. “Migrating to GraphQL: A Practical Assessment”. *SANER 2019 - Proceedings of the 2019 IEEE 26th International Conference on Software Analysis, Evolution, and Reengineering*, (January), 2019, 140–150. https://doi.org/10.1109/SANER.2019.8667986

[3] Malakhov, K. S., Kurgaev, A. P., & Velychko, V. Y. “Modern Restful API DLS and Frameworks for Restful Web Services API Schema Modelling, Documenting, Visualizing”. *Scientific Journals Problems of Programming*, 2018, Vol. 4, pp. 059–068. https://doi.org/10.15407/pp2018.04.059

[4] Ulrich, H., Kern, J., Tas, D., Kock-Schoppenhauer, A. K., Ückert, F., Ingenerf, J., & Lablans, M. “QL 4 MDR: A GraphQL query language for ISO 11179-based metadata repositories”. *BMC Medical Informatics and Decision Making*, 2018, Vol. 19, No.1, pp. 1–7. https://doi.org/10.1186/s12911-019-0794-z

[5] Mark Logic Corp. REST Application Developer’s Guide, MarkLogic Corporation.US. 2019.

[6] Neumann, A., Laranjeiro, N., & Bernardino, J. “An Analysis of Public REST Web Service APIs”. *IEEE Transactions on Services Computing*, June 2018. Pp. 99. https://doi.org/10.1109/TSC.2018.2847344

[7] Hartina, D. A., Lawi, A., & Panggabean, B. L. E. “Performance Analysis of GraphQL and RESTful in SIM LP2M of the Hasanuddin University”. *Proceedings - 2nd East Indonesia Conference on Computer and Information Technology: Internet of Things for Industry, EIConCIT November 2018*, pp. 237–240. https://doi.org/10.1109/EIConCIT.2018.8878524

[8] Brito, G., Mombach, T., & Valente, M. T. “Migrating to GraphQL: A Practical Assessment”. *SANER 2019 - Proceedings of the 2019 IEEE 26th International Conference on Software Analysis, Evolution, and Reengineering*, January 2019, pp. 140–150. https://doi.org/10.1109/SANER.2019.8667986

[9] Hartig, O., & Pérez, J. “An initial analysis of facebook’s GraphQL language”. *CEUR Workshop Proceedings*, June 2017.

[10] Taskula, T. “Advanced Data Fetching with GraphQL: Case Bakery Service”. *Janne Kario M.Sc. (Tech.) Jukka Keski-Luopa M.Sc*, 2018, pp. 14–15.

[11] Farré, C., Varga, J., & Almar, R. “GraphQL Schema Generation for Data-Intensive Web APIs”. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11815 LNCS*, 184–194. https://doi.org/10.1007/978-3-030-32065-2_13

[12] Landeiro, M. I. F. Analysis of GraphQL performance: a case study. Springer International Publishing, 2019.

[13] Ritsilä, A. “GraphQL: The API Design Revolution”, Haaga-Helia University, 2017. Retrieved from https://www.theseus.fi/bitstream/handle/10024/141989/GraphQL- The API Design Revolution.pdf?sequence=1&isAllowed=y

[14] Ghebremicael, E. S. “Transformation of REST API to GraphQL for OpenTOSCA”. University of Stuttgart, 2017. https://doi.org/10.18419/opus-9352
[15] Ulrich, H., Kern, J., Tas, D., Kock-Schoppenhauer, A. K., Ückert, F., Ingenerf, J., & Lablans, M. “QL 4 MDR: A GraphQL query language for ISO 11179-based metadata repositories”. *BMC Medical Informatics and Decision Making*, Vol. 19, No. 1, pp. 1–7, 2019. https://doi.org/10.1186/s12911-019-0794-z
[16] Hossain, A., Nowsin, M., Sheikh, A., Halder, M., Biswas, S., & Arman, A. I. Quality of Service in Software Defined Networking Quality of Service in Software Defined Networking, September, 2018.
[17] Karakus, M., & Durresi, A. “Quality of Service (QoS) in Software Defined Networking (SDN): A survey”. *Journal of Network and Computer Applications*, Vol. 80, pp. 200–218, 2017. https://doi.org/10.1016/j.jnca.2016.12.019
[18] Febiharsa, D., Sudana, I. M., & Hudallah, N. “Information System for Batik Profession Certification Institution”. *Journal of Vocational and Career Education*, Vol. 3, No. 2, 2018. https://doi.org/10.15294/jvce.v3i2.17259
[19] Guo, Y., Deng, F., & Yang, X. Design and Implementation of Real-Time Management System Architecture based on GraphQL. *IOP Conference Series: Materials Science and Engineering*, Vol. 466, No.1, 2018. https://doi.org/10.1088/1757-899X/466/1/012015
[20] Čechák, D. Using GraphQL for Content Delivery in Kentico Cloud. *Is.Muni.Cz*. 2017. Retrieved from https://is.muni.cz/th/qm0cs/thesis.pdf
[21] Hartig, O., & Pérez, J. Semantics and Complexity of GraphQL Preprint Version *. *27th World Wide Web Conference on World Wide Web (WWW)*, (Www), 1155–1164, 2018.
[22] Nogatz, F., & Seipel, D. Implementing GraphQL as a query language for deductive databases in SWI-Prolog using DCGs, quasi quotations, and dicts. *Electronic Proceedings in Theoretical Computer Science, EPTCS*, *234*, 42–56, 2017. https://doi.org/10.4204/EPTCS.234.4