Quantified Student

Technical Documentation—in-memory data caching

Koen Janssen (451394)

Walter Sajtos (438989)



# 

# Abstract

For quantified student, a lot of data is needed. The majority of data is gathered through external sources like canvas. This research document aims to find the most suitable approach to link all the external sources together in a reliable and organized manner.

# 

# 

# Version History

| **Version** | **Author** | **Date** | **Description** | **Reviewers** |
| --- | --- | --- | --- | --- |
| 0.1 | Koen & Walter | 16-05-22 |  |  |
| 0.2 | Walter | 18-05-22 | Added Memcached & Apollo server | Koen |
| 0.3 | Koen | 18-05-22 |  | Walter |
| 0.4 | Walter | 19-05-22 | Write abstract & add decision matrix |  |
| 0.5 | Koen & Walter | 20-05-22 |  |  |

# 

# Table of Contents

[**Abstract**](#_8qp6tm2436fr) **1**

[**This technical**](#_x6y5kaqzq2kw) **1**

[**Version History**](#_rtysupumilb8) **2**

[**Table of Contents**](#_hioheuamors7) **3**

[**1 Introduction**](#_9hgls1ugxz6d) **4**

[**2 Research Questions**](#_9itv2aafsx7) **4**

[2.1 Main research question](#_k3r04qo9bt9k) 4

[2.2 Sub-research questions](#_6eesmvg3byya) 4

[**3 Research methods**](#_umnthlvqcvnp) **5**

[3.1 Research methods sub-question 1](#_f11ht2d9yz2g) 5

[3.2 Research methods sub-question 2](#_mt0uezmyne7j) 5

[3.3 Research methods sub-question 3](#_kkzgcvw96msu) 5

[**4 Findings**](#_qossnobweh8w) **6**

[4.1 Findings sub-question 1](#_foyimdns9tiv) 6

[**Requirements**](#_xgc4lo5677vt) **6**

[4.2 Findings sub-question 2](#_6173ejggc5iz) 7

[P1 Redis](#_g13ksa3n1u7r) 7

[P2 GraphQL with Apollo server](#_nx3p9up2zloh) 7

[4.3 Findings sub-question 3](#_wtl0rx2k5gz) 8

# 1 Introduction

The Quantified Student (QS for short) project focuses on helping students with their development and optimizing their performance with the help of collected data. The collected data will be shown in a dashboard where the student can see it. After which, the student can conclude where and how to improve their workflow.

For building a student dashboard, a lot of data is necessary. The data comes from external sources like canvas. The purpose of this research document is to find the most suitable approach to link all the external sources together in a reliable and organized manner.

# 2 Research Questions

## 2.1 Main research question

How can we build a data interface that allows an easy-to-use, well-structured communication between the dashboard and the external data sources.

## 2.2 Sub-research questions

1. What are the requirements for the In-memory data caching 1 solution?
2. What solutions exist that meet our use case?
3. Which solution fits the requirements best?
4. What are best, good and bad practices for the solution?

# 

# 

# 3 Research methods

## 3.1 Research methods sub-question 1

To assess the requirements for the In-memory data caching solution, we conducted a requirement prioritization as described in (HBO-i, 2018).

## 3.2 Research methods sub-question 2

To assess which available In-memory data caching solutions fit our requirement, we will be conducting an available product analysis as described in (HBO-i, 2018).

## 3.3 Research methods sub-question 3

To assess which In-memory data caching fits the problem best, we will be conducting multi-criteria decision-making and prototyping as described in (HBO-i, 2018).

## 3.4 Research methods sub-question 4

To assess the best and bad practices, we will be conducting best, good and bad practices as described in (HBO-i, 2018).

# 

# 4 Findings

## 4.1 Findings sub-question 1

### Requirements

To provide advice that best fits this issue. A requirements' prioritization will be made. This allows for a concrete selection of solutions and provides the best fit based on the prioritizations.

### Associated-risks

In the requirements, we took into account potential associated risks. Such as enforcing security standards on external data sources, to guarantee the end-user's privacy we encrypt user sensitive data and enforce GDPR guidelines.

| **#** | **Requirement** | **MoSCoW** |
| --- | --- | --- |
| **R1** | **Fetch data from multiple sources** | Must |
| **R2** | **Interface should not add more than 200ms latency.** | Should |
| **R3** | **Fetch and send from an encrypted connection** | Must |
| **R4** | **Load data from an SDL Data Source.** | Should |
| **R5** | **Enforces a data format standard (client-side)** | Must |
| **R6** | **JWT authorization for interacting with data sources** | Should |
| **R7** | **Infrastructure with transferability and simple setup in mind.** | Must |
| **R8** | **Allow for custom queries** | Should |
| **R9** | **Solution can be deployed within a semester** | Should |
| **R10** | **New data sources with minimal coding\*** | Should |
| **R11** | **Functionality is covered in the documentation** | Should |
| **R12** | **Solution should scale horizontally.** | Should |
| **R13** | **The used solution should be distributed under an open-source licence** | Could |
| **R14** | **In-memory caching for frequently requested data to improve performance/latency** | Should |
| **R15** | **Solution must be in line with the GDPR.** | Must |
| **R16** | **Solution must have the possibility to enforce security standards on external data sources** | Must |

*New data sources with minimal coding\*  
To prevent that the application relies on students maintaining it.*

## 4.2 Findings sub-question 2

| # | Name |
| --- | --- |
| P1 | Redis |
| P2 | GraphQL with Apollo server |
| P3 | MongoDB |
| P4 | Memcached |
| P5 | Amazon DynamoDB |
| P6 | Apache Cassandra |

### P1 Redis

*Redis is an open-source, in-memory data store used by millions of developers as a database, cache, streaming engine, and message broker.*

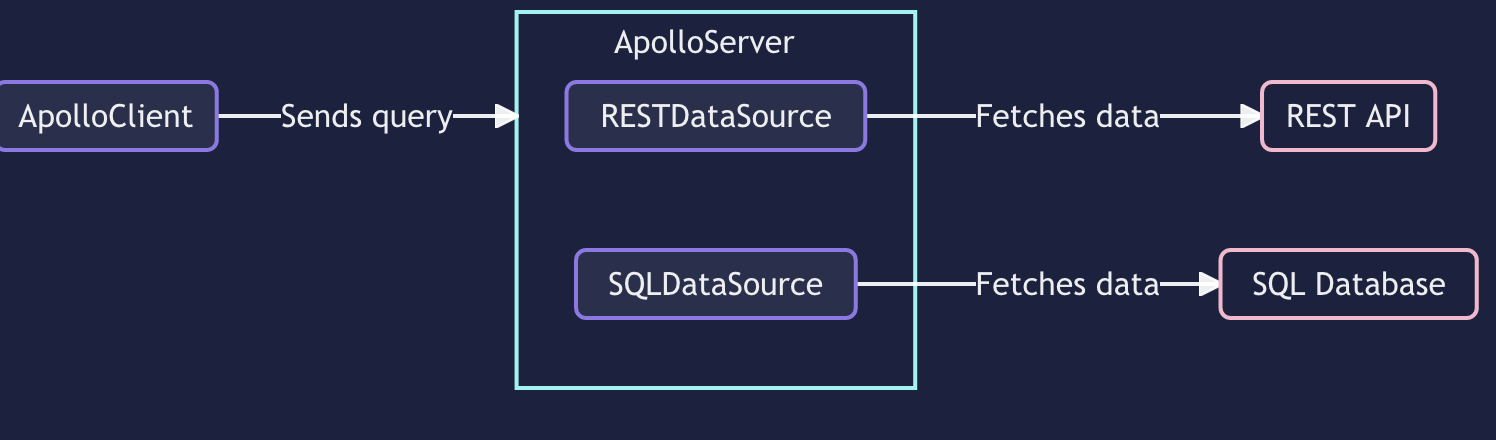
[*https://redis.io*](https://redis.io)

Redis is often used in combination with a relational database such as MySQL, MySQL, or PostgreSQL as a cache, for optimizing query speeds for frequently accessed data.

### P2 GraphQL with Apollo server

*Apollo is a platform for building a* ***unified supergraph****, a communication layer that helps you manage the flow of data between your application clients (such as web and native apps) and your backend services. At the heart of the supergraph is a query language called* ***GraphQL****.*

[*https://www.apollographql.com/docs/*](https://www.apollographql.com/docs/), (*Apollo Docs Home*, n.d.)



An interesting feature of Apollo server is the implementation of data source classes. These classes act as middlemen; they can help with caching and deduplication. There is support for standard REST APIs but also SQL, MongoDB, Cosmos, and Firestore databases. Every data source can have several endpoints, just like a regular REST API. Data can be mutated and unified in every endpoint before returning it to the consumer.

<https://www.apollographql.com/docs/apollo-server/data/data-sources/> (*Data Sources*, n.d.)

Apollo also has built-in support for using Memcached to cache recent requests.

Apollo uses an extended version of express under the hood, which can be accessed through an API. This makes it easy to implement custom middleware for data formatting or authentication.

### Privacy

Since Apollo acts as a middleman, we do not need to store any data ourselves. This makes it easy to comply with the GDPR laws.

<https://www.apollographql.com/docs/apollo-server/data/data-sources/#using-memcachedredis-as-a-cache-storage-backend> (*Data Sources*, n.d.)

### P3 MongoDB

*MongoDB is a free and open-source NoSQL database.* It automatically caches the most recently used documents.

However, this requires that the data from data sources be stored in MongoDB to use the caching functionality. This means that the data needs to be periodically fetched and stored in the database. Risking duplicate data and a lot of maintenance.

Source: <https://www.mongodb.com/document-databases> (*Document Database - NoSQL*, n.d.)

### P4 Memcached

*Free & open-source, high-performance, distributed memory object caching system, generic in nature, but intended for use in speeding up dynamic web applications by alleviating database load.*

[*https://memcached.org/*](https://memcached.org/)(Memcached, n.d.)

Its main use is to cache data to speed up the web by creating a virtual pool of memory, allowing each node to access the cache of a different node. With this approach, there is no need to take syncing the cache of your nodes into account.

Memcached by itself is not an acceptable solution for this system. It can help with caching for other systems, for example, a database. Existing solutions like Apollo server also use Memcached to cache its data.

### 

### P5 Amazon DynamoDB

*Fast, flexible NoSQL database service for single-digit millisecond performance at any scale*

<https://aws.amazon.com/dynamodb/> (*Fast NoSQL Key-Value Database – Amazon DynamoDB – Amazon Web Services*, n.d.)

However, this requires that the data from data sources are stored in DynamoDB to make use of the caching functionality. This means that the data needs to be fetched and stored in the database periodically, risking duplicate data and a lot of maintenance.

### P6 Apache Cassandra

*Apache Cassandra is an open-source NoSQL distributed database trusted by thousands of companies for scalability and high availability without compromising performance.*

### 

### 

## 4.3 Findings sub-question 3

After the available product analysis, we created a decision matrix to rank the products based on our requirements.

| ***#*** | ***Priority*** | ***Criteria / Product*** | ***P1*** | ***P2*** | ***P3*** | ***P4*** | ***P5*** | ***P6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Must | R1: | - | + | - | + | + | - |
| 2 | Must | R2: | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | Must | R3: | + | + | + | + | + | + |
| 4 | Must | R4: | + | + | 0 | + | - | - |
| 5 | Must | R5: |  | + | - | - | - | - |
| 6 | Must | R6: | - | + | - | - | - | - |
| 7 | Must | R7: | + | + | - | - | - | - |
| 8 | Should | R8: |  | + | + | - | + | + |
| 9 | Should | R9: | + | + | + | + | + | + |
| 10 | Should | R10: | - | + | - | - | - | - |
| 11 | Should | R11: | + | + | + | 0 | + | + |
| 12 | Should | R12: | + | + | + | + | + | + |
| 13 | Could | R13: | + | + | + | + | - | + |
| 14 | Should | R14: | + | + | + | + | + | + |
| 15 | Must | R15: | - | + | - | - | - | - |
| 16 | Must | R16: | - | + | - | - | - | - |
|  | Total |  | KO | 12 | KO | KO | KO | KO |

Legend

| ***Character*** | ***Value*** |
| --- | --- |
| KO | Knock-out |
| - | -1 |
| o | 0 |
| + | +1 |

## 4.4 Findings sub-question 4

After conducting library research into the best, good and bad practices for setting up GraphQL. It became clear that the same best practices apply to GraphQL for setting up a REST API, such as enforcing a multi-layered (multi-tiered) architecture.

GraphQL doesn’t have controllers, something comparable to controllers would be resolvers (GraphQL, n.d.). To apply the domain-driven design approach here, we can move business logic into separate modules and import them into the resolvers. This approach has a couple of benefits, for example, we can easily test the business logic, reuse it and avoid code duplication.

# 

# 5 Conclusion

Based on the findings from this research document, we can draw a conclusion. That GraphQL in combination with the Apollo server would be the most suitable data-mediator solution that meets all the outlined requirements. To further test whether the solution would be suitable for our use case, there was a prototype made for testing the functionality and whether it meets our needs, the outcome of this was positive.

# 

# 

# References

(n.d.). Redis. Retrieved May 20, 2022, from https://redis.io

*Apollo Docs Home*. (n.d.). Apollo GraphQL. Retrieved May 20, 2022, from https://www.apollographql.com/docs

*Data sources*. (n.d.). Apollo GraphQL. Retrieved May 20, 2022, from https://www.apollographql.com/docs/apollo-server/data/data-sources/#using-memcachedredis-as-a-cache-storage-backend

*Data sources*. (n.d.). Apollo GraphQL. Retrieved May 20, 2022, from https://www.apollographql.com/docs/apollo-server/data/data-sources/

*Document Database - NoSQL*. (n.d.). MongoDB. Retrieved May 20, 2022, from https://www.mongodb.com/document-databases

*Fast NoSQL Key-Value Database – Amazon DynamoDB – Amazon Web Services*. (n.d.). AWS. Retrieved May 20, 2022, from https://aws.amazon.com/dynamodb/

GraphQL. (n.d.). *Thinking in Graphs*. GraphQL. Retrieved June 1, 2022, from https://graphql.org/learn/thinking-in-graphs/

Memcached. (n.d.). memcached - a distributed memory object caching system. Retrieved May 20, 2022, from https://memcached.org/