Quantified Student

Technical documentation

Kane Petra (456964)

Koen Janssen (451394)

Walter Sajtos (438989)



# 

# Abstract

# The Quantified Student (QS for short) project focuses on helping students with their development and optimising 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 example, the system can show when it is the best time to work for the student.

# 

# Version History

| **Version** | **Author** | **Date** | **Description** | **Reviewers** |
| --- | --- | --- | --- | --- |
| 0.1 | K. Petra | 10.05.2022 |  | K. Janssen |
| 0.2 | K. Janssen | 12.05.2022 | Added argumentation of the framework choice.  Added updated version of Interface section. |  |
| 0.3 | W. Sajtos | 17.06.2022 | Update framework argumentation to apollo |  |
| 0.4 | K. Petra | 28.06.2022 | Added return model for API & added abstract |  |

# 

# Table of Contents

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

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

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

[**Table of Contents**](#_6rh431mk6smz) **3**

[**1 Preface**](#_1248eu7394b5) **4**

[Documents](#_z618b2q0o7v1) 4

[Standards](#_1szl4n3q7wgq) 4

[**2 System Architecture**](#_xho1soja9s1t) **5**

[2.1 Architecture](#_nmvx5kibv1tu) 5

[2.1.1 Router](#_5rsvl7igzlld) 6

[2.1.2 Third Party Controller](#_zd8m0qhwotae) 7

[2.2 Interface](#_ff82czjdo0ax) 7

[2.3 Data storage](#_g5o4ey51d5sp) 7

[2.4 Framework choice](#_sbj7qffimgnv) 7

[**3 Data Model**](#_ple17qpv2sj4) **8**

[**4 Object Model**](#_1g53l5yqd9fw) **8**

[4.1 Objects](#_dy13joz75w4h) 8

[MoreClasses…](#_7g4b92w9ium) 8

[4.2 Return Model](#_rk1xpg96xf6i) 8

[4.2.1 Example](#_4r2t76q11j3o) 9

[**5 System Functions**](#_y5xz2ubiuehg) **10**

[5.1 Post processing](#_ayyv1sailsoc) 10

[**Glossary**](#_o2sbt5v0zlk0) **11**

# 1 Preface

The Quantified Student (QS) API will act as an abstraction over QS-related data. Like health and stress levels. This is abstracted over to ensure compliance and security, finally the API will return data is an already pre-formatted way for a front end to consume.

## Documents

These documents were used when creating this one and will be referenced later on.

* QS - Analysis Document

## Standards

This document uses the following standards and expects the reader to understand them:

* Application Programming Interfaces (API)
* Entity Relationship Diagram (ERD)
* Context, Container, Components & Code model (C4 Model)

# 

# 2 System Architecture

## 2.1 Architecture

The QS API consists of multiple components that communicate with each other. An overview can be found below and a description under that.

## 

### 2.1.1 Router

The main entry point of the API servers, this component will handle routing to the different controllers and does not contain any logic beyond this.

### 2.1.2 Third Party Controller

The third party controller handles data aggregation and access to the data stored in the data warehouse. Endpoints require authentication before being usable, to authenticate see “User Controller” below.

## 2.2 Interface

For interfacing with the API, we chose the JSON data format. This allows us to easily parse the returned data into objects. Further, according to Microsoft Docs, “**JSON** is probably the most common data format for web APIs” (APA ref here: https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design) Making it easy for other developers to use and interact with the API, as the use of JSON is so commonplace in REST APIs.

## 2.3 Data storage

The QS API will not store any data of itself, the only data storage will be data cached as described in 2.4.1. User preferences will be stored on the device and will not be centrally saved.

## 2.4 Framework choice

The framework that has been chosen for the API is Apollo Server.

*“Apollo is a platform for building a* ***unified super graph****, 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”.*

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

The framework is able to retrieve data from an extensive list of data sources. Scales well both vertically and horizontally. Under the hood, the framework uses express. There is support for several [other frameworks](https://www.apollographql.com/docs/apollo-server/integrations/middleware/#all-supported-packages), including serverless options as well.

For the research that was conducted to make this decision can be found in the [Data Mediator Document](https://docs.google.com/document/u/0/d/1tpYuA7ayjZ-L3ijX95-B_2iUKWL2lcTwYDKBK816gEk/edit).

# 3 Data Model

Link Canvas API

https://canvas.instructure.com/doc/api/

# 

# 4 Object Model

## 4.1 Return Model

The QS API will return data from its endpoints in JSON format. A time series array is used as its primary format and will look like this.

* **Identifier**The identifier is used as a unique, unchangeable name to identify which application this response will be for. This is a string as value.
* **Options**The options are one or more options which decide what data is expected in the time series array. This takes the following object as value.  
   - **Axis** On what axis this option should be displayed, either x or y.  
   - **Format** Which format this option expects to be formatted as
* **Data**The data itself is on or more of the predefined options with the expected value.

### 4.2.1 Example

An example which uses an option on both the x and y axis with only one result returned in data.

{

"identifier": "unique\_identifier",

"options": {

"stress": {

"axis": "x",

"format": "int"

},

"time": {

"axis": "y",

"format": "datetime"

}

},

"data": [

{

"stress": 100,

"time": "1656405884519"

}

]

}

# 5 System Functions

## 5.1 Post processing

# 

# Glossary

| *Term* | *Explanation* |
| --- | --- |
| *API* | *Application Programming Interface* |