# Documentation

### Rest API created by Sytze Westerdijk

#### API

The API is created using NodeJS, the following was carried out to prepare the environment:

1. Create a folder, name it API.
2. Run NodeJS command prompt and type ‘npm init -y’ to create the package.json, containing things like the name and version of your API.
3. Install the required packages:

npm install express –save

npm install mysql –save

npm install body-parser

npm install js2xmlparser

1. Create a new file called server.js in the API folder.
2. Refer to server.js for explanation on what each line of code means, this is all commented and explained.

#### Database/API data

The database used was created using mysql and is called api. The structure of this database can be determined by viewing api.sql.

Essentially 3 tables are created: store, product and sales\_fact.

Store: storeID and storeName

Product: productID and productName

Sales\_fact: storeID, productID, sales, cost, profit

Basically, different stores and products are created, whereas sales\_fact records sale tranactions of products using the productID and storeID, giving the cost, price sold at and profit in return.

#### Consuming the API/Visualization

The visualization of the API is done using Javascript, which turned out to be mistake that was realized too late. This, because validation of the XML/JSON data using schemas is not possible on the client side in Javascript, as requested by the assignment. The inclusion of PHP is, therefore, used to validate using the schemas: Draft-07 for JSON and XSD for XML.

Check consumer.js for commented code explaining each line or segment of code and its purpose.

Using visual studio code, install the extension: consumer

In the terminal, while in the API folder run:

composer require justinrainbow/json-schema

The [justinrainbow/json-schema](https://github.com/justinrainbow/json-schema) package allows one to define a schema for what’s allowed in the JSON file, and then validate against it.

Check index.php for commented code explaining each line or segment of code on how the XML and JSON data are validated against the schemas.

Create a schema.json file and paste the result of the following link and using example json data to generate the schema to compare to:

<https://jsonschema.net/>

Do the same for XML: schema.xsd using the Russian doll design:

<https://www.freeformatter.com/xsd-generator.html>

#### Why the languages were chosen:

NodeJS for the API:

NodeJS is a JavaScript runtime environment that runs server-side, meaning XAMPP must be used to test the code. As an infrastructure, NodeJS is designed for building scalable network applications, meaning it is perfect for an API, which may scale over time as more data is added. It is quite easy to setup to run on a local machine and only requires a couple of lines of code to create the API itself. Routes are also easy to define. When changes are made, you can simply stop and start the API within seconds, making it ideal to develop with. NodeJS is also asynchronous.

JavaScript for the visualization:

In the end, this turned out to be a mistake, but due to time constraints I continued with JavaScript. It is an ideal language to call an API and makes it very easy to visualize the data using libraries such a chart.js, which was used to create a simple yet snappy and esthetically pleasing. Its easy to code, meaning less code can be used to achieve a lot when compared to other languages. I should have done research beforehand, but it turns out JSON and XML schemas can not be validated within JavaScript, this is why PHP is introduced in index.php to validate the provided data against the schemas.