

# DATENBANKEN UND WEBTECHNIKEN REPORT

CHIAO FAN YANG      ENROLLMENT NUMBER: 434967

ORÇUN ORUÇ          ENROLLMENT NUMBER: 321448

4<sup>th</sup> of August, 2017

# Gas Station Finder

## Table OF CONTENTS

AUTHOR: CHIAO FAN YANG

<b>1. USER INTERFACE DESIGN CONCEPT .....</b>	<b>2</b>
<b>2. FRONT END DESIGN – FRAMEWORK AND TECHNOLOGY .....</b>	<b>3</b>
2.1 <i>Bootstrap Theme Framework: Freelancer .....</i>	<i>4</i>
2.2 <i>Google Map API.....</i>	<i>5</i>
2.2 <i>Google Chart API.....</i>	<i>6</i>
<b>3. RESTFUL API REQUEST AND RESPONSE .....</b>	<b>7</b>
<b>4. VERSION CONTROL TOOL – BITBUCKET +SMART GIT.....</b>	<b>12</b>
<b>5. AGILE DEVELOPMENT APPROACH – KANBAN .....</b>	<b>13</b>
AUTHOR: ORÇUN ORUÇ	
<b>6. BACKEND TECHNOLOGIES .....</b>	<b>14</b>
6.1. <i>Node.js .....</i>	<i>14</i>
6.2. <i>Node.js Modules .....</i>	<i>15</i>
6.2.1. <i>Express Framework .....</i>	<i>15</i>
6.2.2. <i>Body Parse .....</i>	<i>16</i>
6.3. <i>Anatomy of HTTP Transaction in Node.js.....</i>	<i>16</i>
6.4. <i>API Documentation.....</i>	<i>17</i>
<b>7. DATABASE TECHNOLOGIES.....</b>	<b>19</b>
7.1. <i>Relational Database Management System of the Project.....</i>	<i>19</i>
7.2. <i>The Database Structure in Our Project .....</i>	<i>20</i>
<b>8. MOBILE TECHNOLOGIES .....</b>	<b>22</b>
8.1. <i>Android Operating System.....</i>	<i>22</i>
8.2. <i>Gas Station Finder in Android.....</i>	<i>24</i>
<b>9. CONCLUSTION .....</b>	<b>25</b>
<b>10. REFERENCE.....</b>	<b>26</b>

# Gas Station Finder

## 1. User Interface Design concept

Bootstrap – flat design + responsive

Gas station finder is a simple and useful application for people who want to have overall information of gas station, so our design goal is that user can use easily without pain and as simple as possible. We chose Bootstrap as our framework of user interface because **Bootstrap can quickly and easily prototype new designs**. In addition, our design concept is to build a responsive website which suit for various screen size. The following picture is Gas station finder's station information view. **We can see from the pictures that it suits for different screen size.**

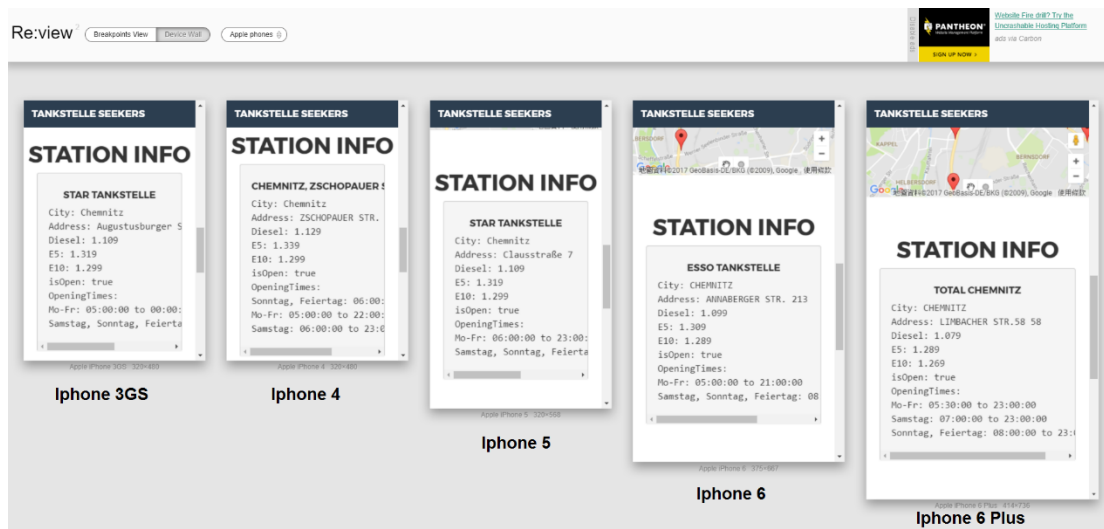


Figure 1. Station information section with various screen sizes regarding to IPHONE.

# Gas Station Finder

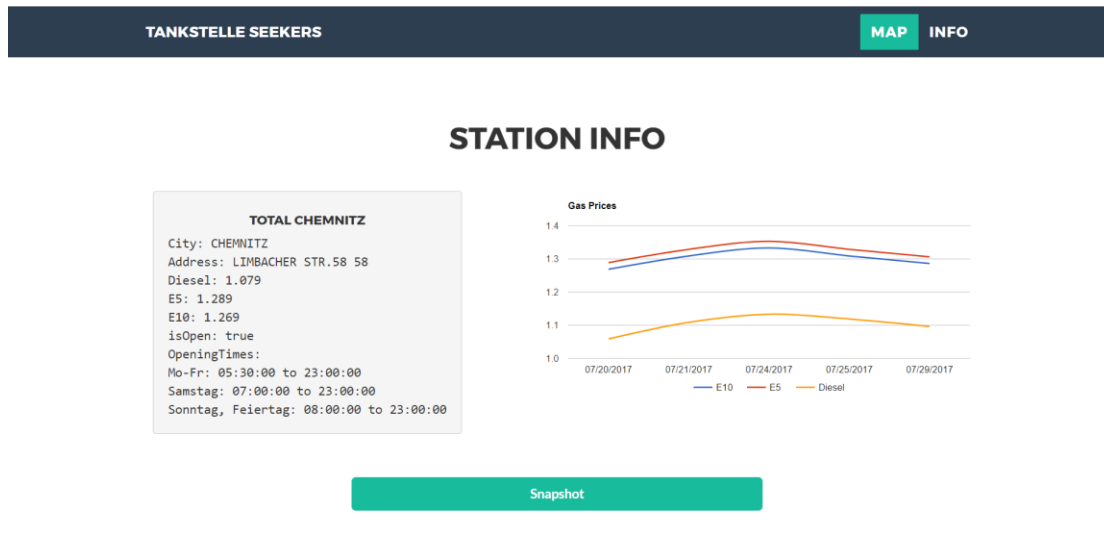


Figure 2. Station information section view by PC.

Bootstrap is a good framework for your purpose. The design element in this **freelancer theme** are **flat design and mobile friendly**. Flat design is between Material design and skeuomorphism. Flat design is a minimalistic design approach that emphasizes usability. It features clean, open space, crisp edges, bright colors and two-dimensional illustrations.

## 2. Frontend Design - Framework & Technology

We used as following languages for frontend of Gas station finder:

- HTML5
- CSS
- Javascript
- JQuery, version: 3.2.1

We used as following API/Frameworks for frontend of Gas station finder:

- Bootstrap, Freelancer theme
- Google Font API
- Google Map API
- Google Chart API
- XMLHttpRequest API

# Gas Station Finder

## 2.1 Bootstrap Theme Framework: Freelancer

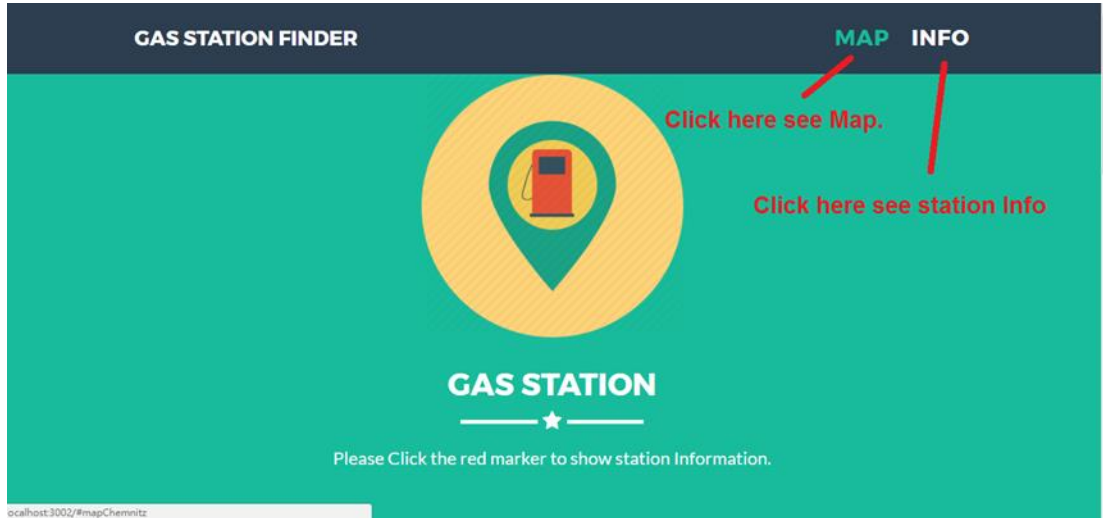


Figure 3. The cover of GAS STATION FINDER

**Framework link:** <https://startbootstrap.com/template-overviews/freelancer/>

**Theme version:** Bootstrap 4 Alpha Version

### Theme Description:

Freelancer is a One-page bootstrap theme for freelancers. One page websites require visitors to browse through information in a linear fashion as opposed to clicking and exploring around page to page. Our purpose for the website is simple and easy to understand, so we chose this theme to provide a better User experience.

### Theme figures:

- Flat icons by flaticons.com
- LESS files and compiled CSS included
- Custom outline button style
- Mobile friendly

# Gas Station Finder

## 2.2 Google Map API

**Purpose:** Showing the map. Let users know the location of gas station and user can choose the circle area to see the information of station.

**Google Map API Version:** The experimental version

**Google Map API Type:** Google Maps JavaScript API

**Framework link:** <https://developers.google.com/maps/>

### Introduction:

Google APIs is a set of application programming interfaces (APIs) developed by Google which allow communication with Google Services and their integration to other services. Examples of these include Search, Gmail, Translate or Google Maps. Third-party apps can use these APIs to take advantage of or extend the functionality of the existing services. We used Google Maps Javascript API for your gas station finder.

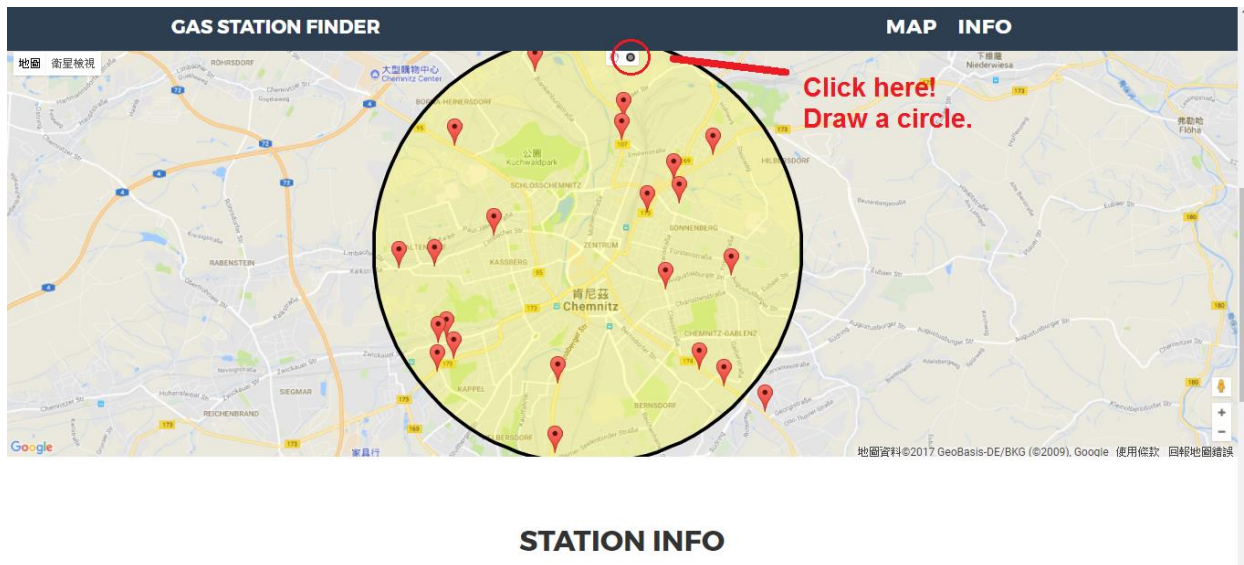


Figure 4. Map section of website.

Click the little circle to draw the area of station you want to know. After drawing area and click hand sign, we can choose a station we want to know.

In the beginning, we need to apply a API\_KEY from google by email, then put the following code in your html <head>.

**Applying API KEY link:**

<https://developers.google.com/maps/documentation/directions/get-api-key>

# Gas Station Finder

```
<script async defer
  src="https://maps.googleapis.com/maps/api/js?v=3.exp&
    key=YOUR_API_KEY&call back=initMap">
```

We utilized Google API DrawingManager to increase the user experience of gas station finder.

**Google Map API DrawingManager tutorial:**

<https://developers.google.com/maps/documentation/javascript/drawinglayer>

## 2.3 Google Chart API

**Purpose:** For each gas station there has to be a graphical representation over the price trend available to show users the current price change.

**Chart Type:** Line Chart

**Load the chart library:**

```
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
```

This tag can be either in the `head` or `body` of the document, or it can be inserted dynamically into the document while it is being loaded or after loading is completed.

**Framework link:** <https://google-developers.appspot.com/chart/>

**Data Structure:**

```
var getPriceTrendData = [['Time', 'E10', 'E5', 'Diesel'],
  [ '24/07/2017', 1.4, 1.6, 1.5],
  [ '25/07/2017', 1.5, 1.7, 1.3],
  [ '26/07/2017', 1.8, 1.6, 1.0],
  [ '27/07/2017', 1.1, 1.4, 1.4],
  [ '28/07/2017', 1.3, 1.1, 1.6]];
```

# Gas Station Finder

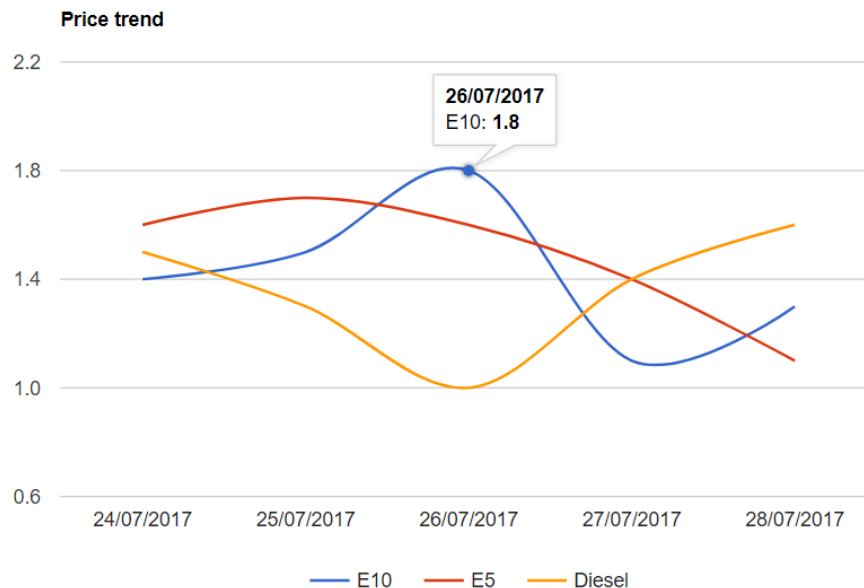


Figure 5. Google Chart API Price Trend example

## 3. Restful API request and response

**Purpose:** We use **XMLHttpRequest** API for sending HTTP GET, POST to our own API.

**Introduction:** XMLHttpRequest (XHR) is an API in the form of an object whose methods transfer data between a web browser and a web server. The object is provided by the browser's JavaScript environment. XHR support various data type not only XML but also plain text and JSON.

We used XMLHttpRequest to send GET request, then receive JSON file from server and present them in our user interface.

**Restful request:**

**HTTP GET:** Fetch gas station data (prices, names, address, brand, Opening hours) of a given geographical position and radius.

**GET /StationInformation/?lat=50.8357&lng=12.92922&rad=4 HTTP/1.1**

**Host: localhost:3002**

**Accept: Application/JSON**





# Gas Station Finder

```
{
  ok: true,
  license: "CC BY 4.0 - https://creativecommons.tankerkoenig.de",
  data: "MTS-K",
  status: "ok",
  stations:
[
  {
    id: "9516ea04-a21b-478c-aaa6-f69e690a28f8",
    name: "CHEMNITZ - DRESDNER STR. 84",
    brand: "Agip",
    street: "Dresdner Str.",
    place: "Chemnitz",
    lat: 50.841591760457,
    lng: 12.935632657089,
    dist: 0.8,
    diesel: 1.079,
    e5: 1.319,
    e10: 1.299,
    isOpen: true,
    houseNumber: "84",
    postCode: 9130
  },
  And more stations .....
]}
}
```

**HTTP GET:** Fetch each gas station “opening hours” (prices, names, address, brand, Opening hours) of a given “station ID”.

**GET** /getPriceTrend/9516ea04-a21b-478c-aaa6-f69e690a28f8 **HTTP/1.1**

**Host:** localhost:3002

**Accept:** Application/JSON

**Server Response:**



# Gas Station Finder

```
{
  ok: true,
  license: "CC BY 4.0 - https://creativecommons.tankercoenig.de",
  data: "MTS-K",
  status: "ok",
  station:
  {
    id: "0ededd2c-381c-45fc-830e-efe32fba2bc2",
    name: "Esso Tankstelle",
    brand: "ESSO",
    street: "ANNABERGER STR.213",
    houseNumber: " ",
    postCode: 9120,
    place: "CHEMNITZ",
    openingTimes:
    [
      {
        text: "Mo-Fr",
        start: "05:00:00",
        end: "21:00:00"
      },
      {
        text: "Samstag, Sonntag, Feiertag",
        start: "08:00:00",
        end: "21:00:00"
      }
    ],
    overrides: [],
    wholeDay: false,
    isOpen: true,
    e5: 1.309,
    e10: 1.289,
    diesel: 1.079,
    lat: 50.803587,
    lng: 12.912419,
    state: null
  }
}
```

# Gas Station Finder

```
}  
}
```

HTTP GET: Get price trend for a given gas station.

GET /getPriceTrend/9516ea04-a21b-478c-aaa6-f69e690a28f8 HTTP/1.1

Host: localhost:3002

Accept: Application/JSON

Server Response:

```
[  
  {  
    ID: "9516ea04-a21b-478c-aaa6-f69e690a28f8",  
    E5: 1.299,  
    E10: 1.279,  
    DIESEL: 1.079,  
    DATE: "07/18/2017"  
  },  
  {  
    ID: "9516ea04-a21b-478c-aaa6-f69e690a28f8",  
    E5: 1.289,  
    E10: 1.269,  
    DIESEL: 1.059,  
    DATE: "07/20/2017"  
  },  
  {  
    ID: "9516ea04-a21b-478c-aaa6-f69e690a28f8",  
    E5: 1.349,  
    E10: 1.329,  
    DIESEL: 1.129,  
    DATE: "07/21/2017"  
  },  
  {  
    ID: "9516ea04-a21b-478c-aaa6-f69e690a28f8",  
    E5: 1.3034444444444444,  
    E10: 1.2834444444444444,  
    DIESEL: 1.0834444444444442,
```



# Gas Station Finder

```
DATE: "07/24/2017"
},
{
  ID: "9516ea04-a21b-478c-aaa6-f69e690a28f8",
  E5: 1.359,
  E10: 1.339,
  DIESEL: 1.149,
  DATE: "07/25/2017"
},
{
  ID: "9516ea04-a21b-478c-aaa6-f69e690a28f8",
  E5: 1.322125,
  E10: 1.302125,
  DIESEL: 1.112125,
  DATE: "07/29/2017"
}
```

]

**HTTP POST:** Make a snapshot of gas station prices of all gas station in a given geographical position.

**POST** /makeSnapshot HTTP/1.1

**Host:** localhost:3002

**Content-type:** Application/JSON

**HTTP POST Content body:**

```
{"stations":[
  {"id":"9516ea04-a21b-478c-aaa6-f69e690a28f8","e5":1.319,"e10":1.299,"diesel":1.079,"date":"07/30/2017"},
  {"id":"005056ba-7cb6-1ed2-bceb-be1b10ed4d4e","e5":1.309,"e10":1.289,"diesel":1.069,"date":"07/30/2017"},
  {"id":"005056ba-7cb6-1ed2-bceb-831997b4cd2d","e5":1.309,"e10":1.289,"diesel":1.069,"date":"07/30/2017"},
  {"id":"688962a2-3dd0-458f-bcac-a4886e12da99","e5":1.309,"e10":1.289,"diesel":1.069,"date":"07/30/2017"},
  .... and more
]}
```



## 4. Version control tool – Bitbucket + Smart Git

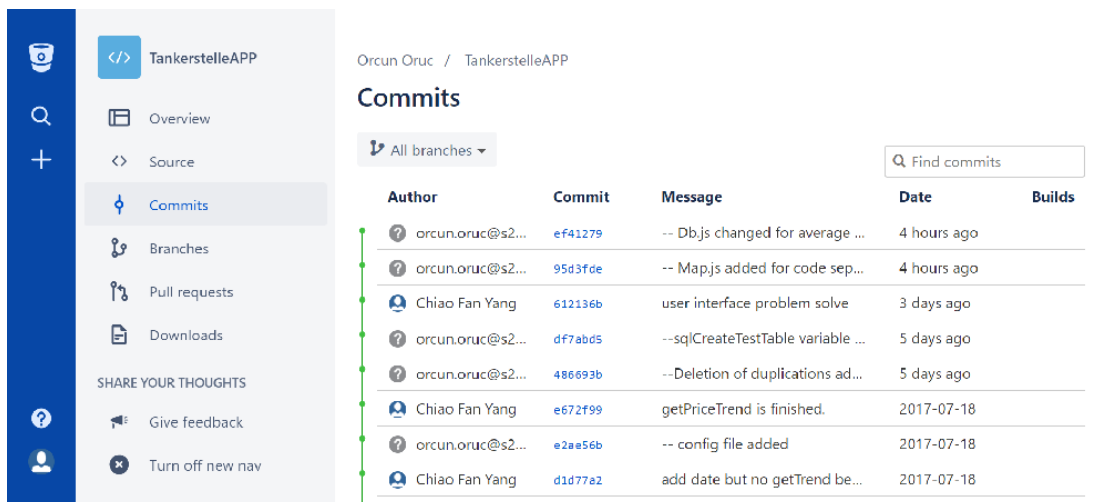
**Purpose:** Our development team consist of 2 students. As long as team work in software development, Version Control is an important topic! We used Bitbucket + Smart Git tools to solve the problem of Version sync.

The benefits of using those tools for our project:

- **Backup** – if another member loses the code by accident, all you need for recovery is one of your teammates' local Git repository.
- **Understanding What Happened** - we can see what exactly was changed in the file's content.
- **Restoring Previous Versions& Storing Versions (Properly)** - Being able to restore older versions of a file (or even the whole project) effectively.

Tool link for Bitbucket: <https://bitbucket.org/>

Tool link for Smart Git: <http://www.syntevo.com/smartgit/>



Author	Commit	Message	Date	Builds
orcun.oruc@s2...	ef41279	-- Db.js changed for average ...	4 hours ago	
orcun.oruc@s2...	95d3fde	-- Map.js added for code sep...	4 hours ago	
Chiao Fan Yang	612136b	user interface problem solve	3 days ago	
orcun.oruc@s2...	df7abd5	--sqlCreateTestTable variable ...	5 days ago	
orcun.oruc@s2...	486693b	--Deletion of duplications ad...	5 days ago	
Chiao Fan Yang	e672f99	getPriceTrend is finished.	2017-07-18	
orcun.oruc@s2...	e2ae56b	-- config file added	2017-07-18	
Chiao Fan Yang	d1d77e2	add date but no getTrend be...	2017-07-18	

Figure 6. Bitbucket Commits shows the project progress and save files in different version.



# Gas Station Finder

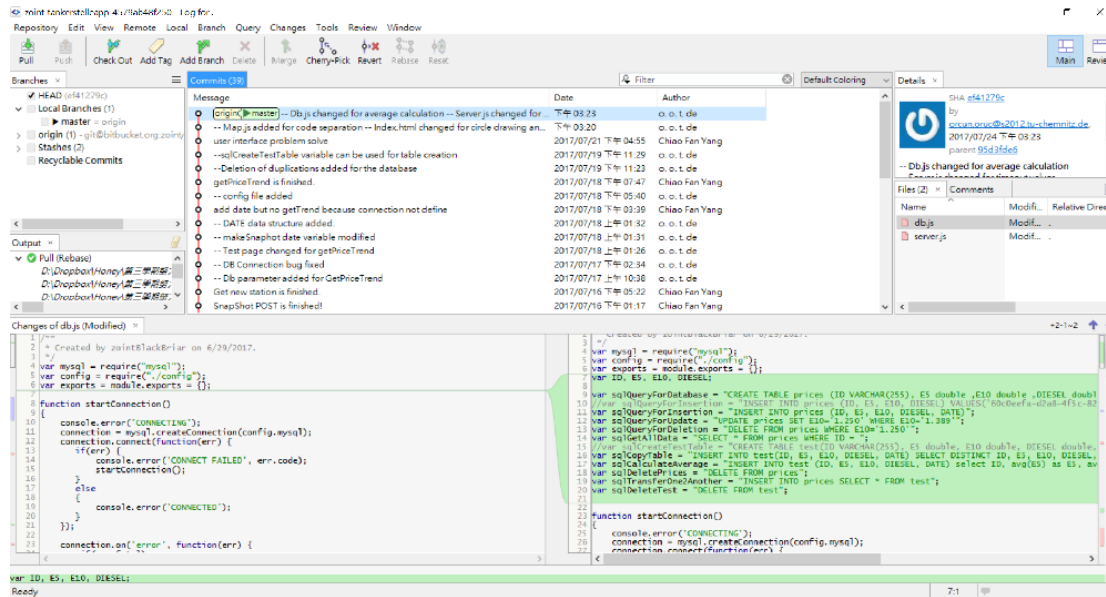


Figure 7. SmartGit can show where did you change the code and where did your team members change the code.

## 5. Agile Development Approach – Kanban

**Purpose:** Agile development reduces risk and increase productivity. It helps us to arrange the process of development. Each sprint we clearly decide our goal and communicate with each other frequently.

**Introduction:** Agile Methodology is a type of project management approach. Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end.

**Kanban Tool Introduction:** Kanban Tool is a visual process management solution that helps teams to work more efficiently, visualize workflow, analyze and improve business processes in line with Kanban method.

Kanban Tool link: <http://kanbantool.com/>

# Gas Station Finder

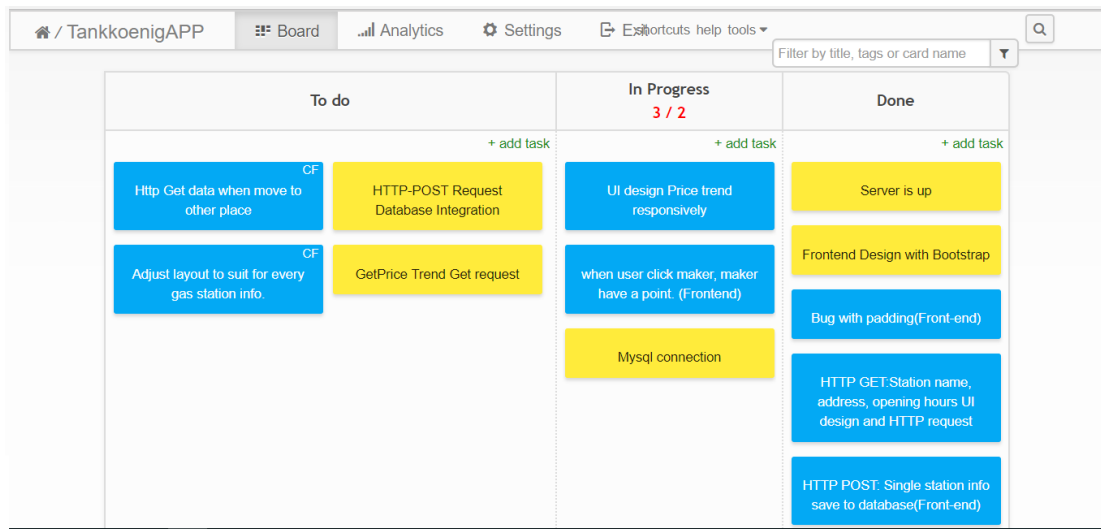


Figure 8. Kanban Tool screen shot

## 6. BACKEND TECHNOLOGIES

### 6.1. NODEJS

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine) [1]. Node.js is a new technology that provides a quick, scalable and integrable solutions to the software teams. Although it is a new technology, it is will be a trending development framework in the near future. Node.js parse the request up through the HTTP headers and provide them as part of the request object. But Node.js does not start parsing the body of the request until the callback has been fired. This is different from other server-side frameworks, like PHP, where both the headers and the body of the request are parsed before your application logic runs.

Node.js provides this lower-level interface so you can handle the body of data as it is being parsed, if desired. Node's core APIs are always lightweight and low-level. This leaves opinions, syntactic sugar, and specific details up to the community modules. Node's core APIs are always lightweight and low-level. This leaves opinions, syntactic sugar, and specific details up to the community modules [2].

# Gas Station Finder

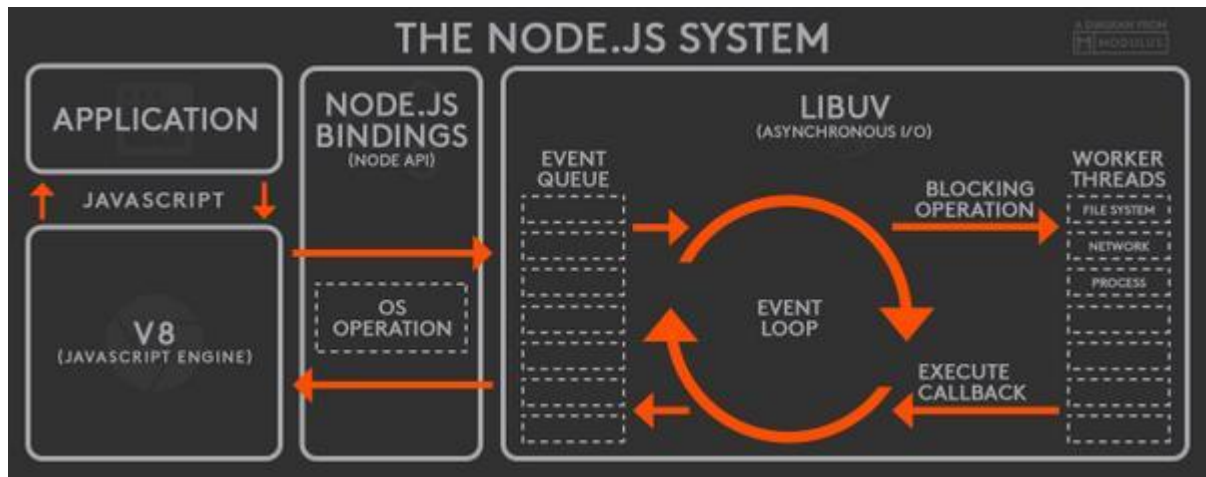


Figure 9: Node.js Basic Architecture [3]

The main feature of Node.js is being an asynchronous I/O framework which means that you can use single thread environment to implement your project without a thread overhead. It uses V8 Google's open source JavaScript engine that resides in Chrome browser. Besides V8 virtual engine provides a safe-sandbox environment for the applications, it translates your JavaScript code into machine code so that is blazing fast. V8 is written in C++. Our Node.js modules (libraries) and local programs written in JavaScript resides in Application section. In **Binding**, the system basically is a wrapper around a library written in one language and expose the library to codes written in another language so that codes written in different languages can communicate [4].

Last but not least, Node.js provides us a good communication environment in our project. We have finished our project less-sized code and without multi-thread overhead. Moreover, by parsing our REST-API calls in JavaScript, we leverage the data-object consistency in our project. Nowadays, Node.js has many modules (libraries) written in JavaScript and we used many of them in the project.

## 6.2. NODE.JS MODULES

### 6.2.1. Express Framework

Express.js is the best fit and simple web application development framework compatible with Node.js.

It is a minimal and flexible Node.js web application framework, providing some features for building single page and hybrid web applications [5]. Express framework let you easily extend with templates engines, and many things easily, which are often that people like to use. Moreover, it is a tiny module of objects related to the web usage. It is basically for the object; application, request, response, and router. Express framework resides community modules are where Node thrives. Each of them with several properties and methods all related with basic



# Gas Station Finder

web operations so that you have them under your development tools. Following are some of the core features of Express framework [6].

- Allows to set up middleware to respond to HTTP Requests.
- Defines a routing table which is used to perform different actions based on HTTP Method and URL.
- Allows to dynamically render HTML Pages based on passing arguments to templates.

## 6.2.2. Body Parser

When the request comes, we need to parse this request before event-handler takes. It provides a body parser in node.js. Basically, it parses the HTTP request body. This is usually necessary when we need to know more than just the URL we hit, particular in the context of a POST or PUT, PATCH, HTTP request where the information we want to contain the body. More importantly, it allows a middleware for parsing JSON, plain text, or returning a raw Buffer object for you to deal with as you require.

Middleware is any number of functions that are invoked by the Express.js routing layer before your final request handler is, and thus sits in the middle between a raw request and the final intended route [7]. For example, we want to upload index.html page at the very beginning. We need to send a request “localhost: 3002/” to the server and the server will send an answer back to us whether we can reload index.html or not. Then we can send a request for price trends or station information. Suppose we want to use aforementioned request. Express framework will add the following route to its own logging logic.

/ - index.html reload

/ StationInformation – get station data from API

/ getPriceTrend – fetch data from database

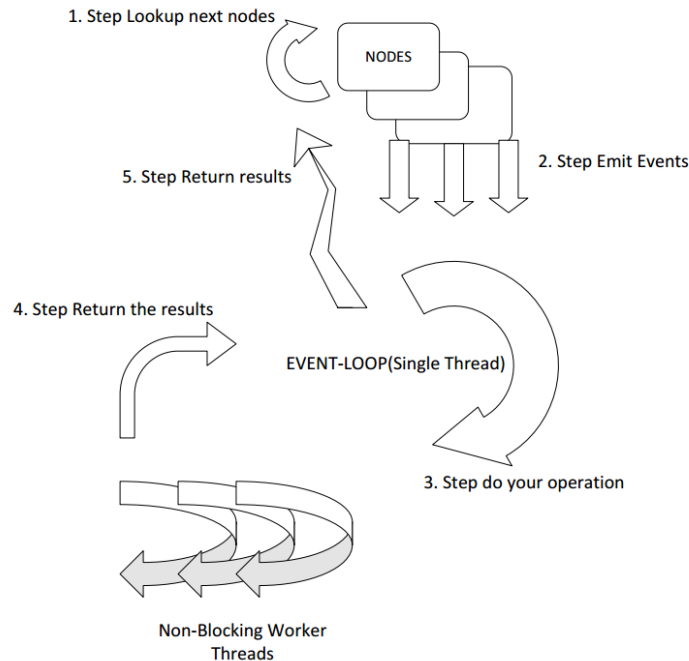
The above-mentioned request will be parsed first and then routed with “next” member respectively. If there would be an error, next member of routing will route the error page.

## 6.3. ANATOMY of HTTP TRANSACTION IN Node.js

After we create a server, we have to handle HTTP requests in Node.js. It works with Event Emitters and Streams. Every HTTP request that against the server so called **request handler**. Server object will send an Event Emitter and the Emitter will emit the respective events. In the figure 1.2, Node.js architecture can be seen clearly. Libraries of Node.js (as depicted **nodes** keyword) installed respectively in the very beginning. When event triggered by HTTP request, the server object will emit events to the event loop by adding listener (event handler) itself. In the single thread environment, multiple request will be evaluated in an

# Gas Station Finder

asynchronous way. Non-blocker worker threads deliver the results of the function in accordance with middleware routing.



**Figure 10:** Architecture of the Node.js Program

When handling a request, we need to extract two things which are the method and URL. We will get the server, protocol and port value with “URL” parameter. The method parameter shows us which verb will implement in the request. If we have “POST” verb in our request, the request body might be important to your application. Moreover, an error in the request stream presents itself by emitting an “error” event on the stream [8]. If you don’t have a listener for that event, the error will be thrown, which could crash our Node.js program.

## 6.4. API Documentation in Backend Service

In the following part, we will describe how to use API. We mainly created for two types of API for each requests.

REST CALL	PARAMETER AND VERB	SAMPLE ANSWER	NOTES



# Gas Station Finder

<b>/StationInformation/lat=[lat]&amp;lng=[lng]&amp;rad=[rad]</b>	@VERB : GET @PARAM : Lat, lng, rad	"stations":[ { "id": id, "name":name, "brand":brand, "street":street, "place": "Berlin", "lat": lat, "lng":lng, "dist":dist, "diesel":diesel, "e5":1.339, "e10": e10 ..	A call for fetching Station Data from API.
<b>/test/StationInformation/lat=[lat]&amp;lng=[lng]&amp;rad=[rad]</b>	@VERB : GET @PARAM : Lat, lng, rad	"stations":[ { "id": id, "name":name, "brand":brand, "street":street, "place": place, "lat": lat, "lng":lng, "dist":dist, "diesel":diesel, "e5":1.339, "e10": e10 ..	A test call for test.html and Postman. Tested error types.
<b>/OpeningHours/{ID}</b>	@VERB : GET @PARAM : ID	"stations":[ { "id": id, "name":name, "brand":brand, "street":street, "openingtimes" : [ {"text": "Mon-Fr", "start": "06:00:00", "end": "22:30:00"}, ...	A call for fetching Opening Hours regarding the specific station
<b>/test/OpeningHours/{ID}</b>	@VERB : GET @PARAM : ID	"stations":[ { "id": id, "name":name, "brand":brand, "street":street, "openingtimes" : [ {"text": "Mon-Fr", "start": "06:00:00", "end": "22:30:00"}, ...	A test call for Opening Hours in test.html
<b>/getPriceTrend/{ID}</b>	@VERB : GET @PARAM: ID	[{"ID":ID, "E5":1.304, "E10":1.284, "DIESEL":1.839, "DATE":07/20/2017}, ..]	A call for price trend data that located in the database
<b>/test/getPriceTrend/{ID}</b>	@VERB: GET @PARAM: ID	[{"ID":ID, "E5":1.304, "E10":1.284, "DIESEL":1.839, "DATE":07/20/2017}, ..]	A test call against the database
<b>/test/makeSnapshot</b>	@VERB: POST @PARAM: No parameter. Only has request body	ERROR OR SUCCESSFUL ENTRY	A test call against the database for test.html and Postman
<b>/makeSnapshot</b>	@VERB: POST @PARAM: No parameter. Only has request body	ERROR OR SUCCESSFUL ENTRY	A call that sends package data about stations and its daily prices

## 7. DATABASE TECHNOLOGIES

### 7.1. RELATIONAL DATABASE MANAGEMENT SYSTEM OF THE PROJECT

MySQL database has been selected for the project. It is a true multi-threaded database server that excels at retrieving information [12]. We need a reliable and robust system and need to connect the data in a proper time interval to fetch data from Tankerkönig API. We have worked mainly a Linux-based operating system and this forced us to use a software which complies with a Linux-based operating system.

As compared with other competitors, MySQL database is really fast according to I/O speed. We have tested this feature when we implemented a query with a database query instructor. We didn't choose NoSQL schema-less based database because we don't have many complicated and undefinable type structure in the database. We have observed String value is enough for ID, Names, and Date and double value for Prices (e.g. prices table). We have defined database name as "tankkoenigDB" and username is "tankkoenig". We have used a Node.js module which provides an asynchronous promise-based client library for the version of MySQL 5.7.12. [12]

MySQL database has some advantage over the counterparts which are;

**Operational System Compatibility:** We can use the MySQL database in any environment as we wish. Node.js application integrated to MySQL database as well as PHP applications did.

**Cost:** We are cost dependent when we created an application with MySQL database. This feature make it easier our life if would create the application from local environment to potentially shippable product.

**Reliability and Scalability:** By using terminal, we can reach MySQL database environment very quickly. It has one of the most reliable structure in the database world.

**Use different architecture in the same database:** We can use InnoDB and MYISAM storage engines in MySQL. These two differ on their locking implementation: **InnoDB** locks the particular row in the table, and **MYISAM** locks the entire MySQL table. It can be specified while creating a table in DB [9].

We could have chosen other database type like MongoDB. But the main reason that we did not choose because we had concrete data types in the modelling phase. We do not have to use a document-oriented database types. We have many data in "prices" and "locations" table which would led to unstable schema and redundant data duplication.

# Gas Station Finder

The other feature of relational database systems that can handle with complex transaction. So that we can provide an atomic transaction support for the system. The reason that we choose MySQL is given in the following key features, which are:

- a) When the data is fetching, the database provides high performance
- b) When we want to show the graphs about price trends to the end-user, we need a solution that would be highly available.
- c) We had structured and simple data types in the schemas.
- d) We need to be sure about proper data indexing.

MySQL helped us to meet the above-mentioned challenges.

## 7.2. THE DATABASE STRUCTURE IN OUR PROJECT

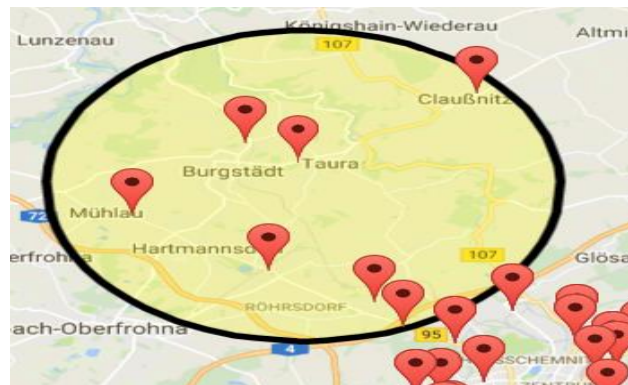
**Setup a User and Access Grant Type:** Database needs to be inserted with the following parameters in the Ubuntu 16.04 LTS system. This stage is very important that we able to reach database properly.

**Prices Table:** We have used a table for price trends. Every prices that we have chosen the station with ID saved into this table:

**Locations Table:** All the locations related to REST API registered into this database.

**Opening Hours Table:** All the opening hours of the selected stations will be shown in this table.

We have tried to keep simple the tables as much as possible. Data is registering into the database in every call with the circle after clicking make snapshot button as depicted in Figure 1.3.



**Figure 11:** Send data to the database with UI



# Gas Station Finder

ID	E5	E10	DIESEL	DATE
00016899-3247-4444-8888-acdc00000007	1.479	1.4589999999999999	1.269	07/27/2017
00060029-0003-4444-8888-acdc00000003	1.309	1.289	1.069	07/23/2017
00060055-0001-4444-8888-acdc00000001	1.309	1.289	1.069	07/27/2017
00060063-0001-4444-8888-acdcffffff	1.289	1.269	1.079	07/27/2017
00060117-0001-4444-8888-acdcffffff	1.399	1.3789999999999998	1.179	07/27/2017
00060157-0001-4444-8888-acdc00000001	1.369	1.349	1.159	07/27/2017
00060182-0001-4444-8888-acdc00000001	1.309	1.289	1.069	07/23/2017
00060385-0001-4444-8888-acdc00000001	1.3390000000000002	1.319	1.109	07/27/2017
00060386-0002-4444-8888-acdc00000002	1.3390000000000002	NULL	1.109	07/27/2017
00060562-0001-4444-8888-acdc00000001	1.417	1.397	1.2170000000000003	07/27/2017
00060613-0004-4444-8888-acdc00000004	1.309	1.269	1.069	07/23/2017
00060711-0001-4444-8888-acdc00000001	1.324	NULL	1.0839999999999999	07/23/2017
00060731-1059-4444-8888-acdc00000001	1.2490000000000003	1.229	1.079	07/27/2017
00060909-b59e-4444-8888-acdc00000609	1.2989999999999997	1.279	1.079	07/23/2017

Figure 12. Sample Locations table

As you can see in Figure 1.4, the prices table stores the data in accordance with ID and DATE. If two rows randomly come from the UI side in an equal way (ID and DATE), we calculate the average price of the certain date and register into the database.

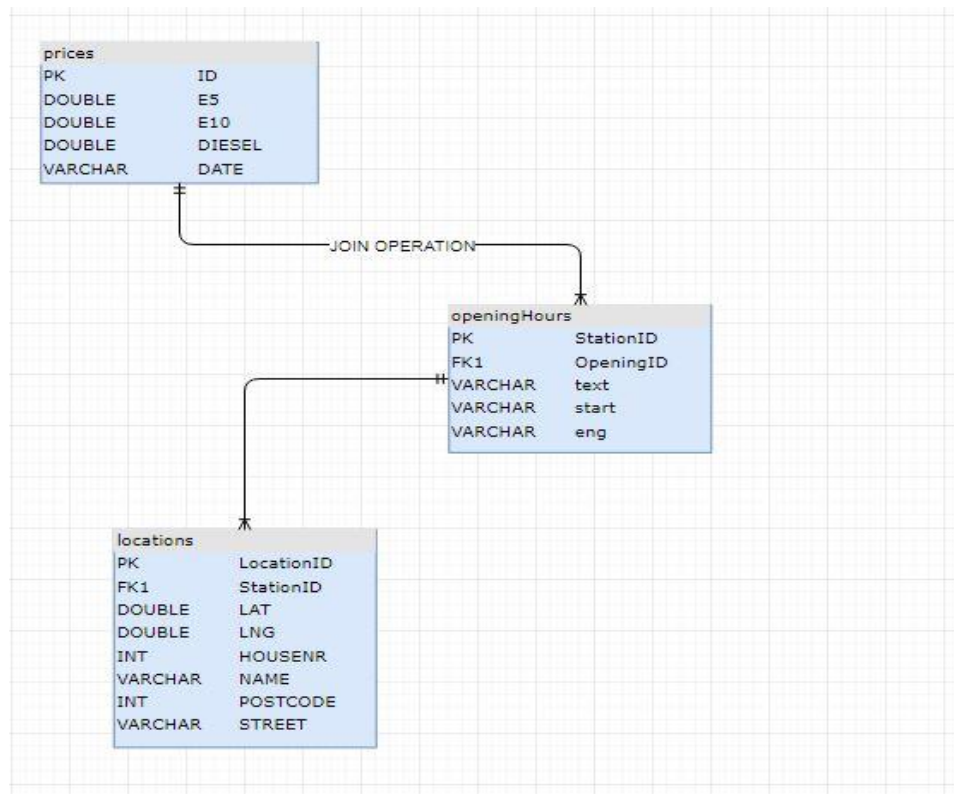


Figure 13. Planned Database Structure

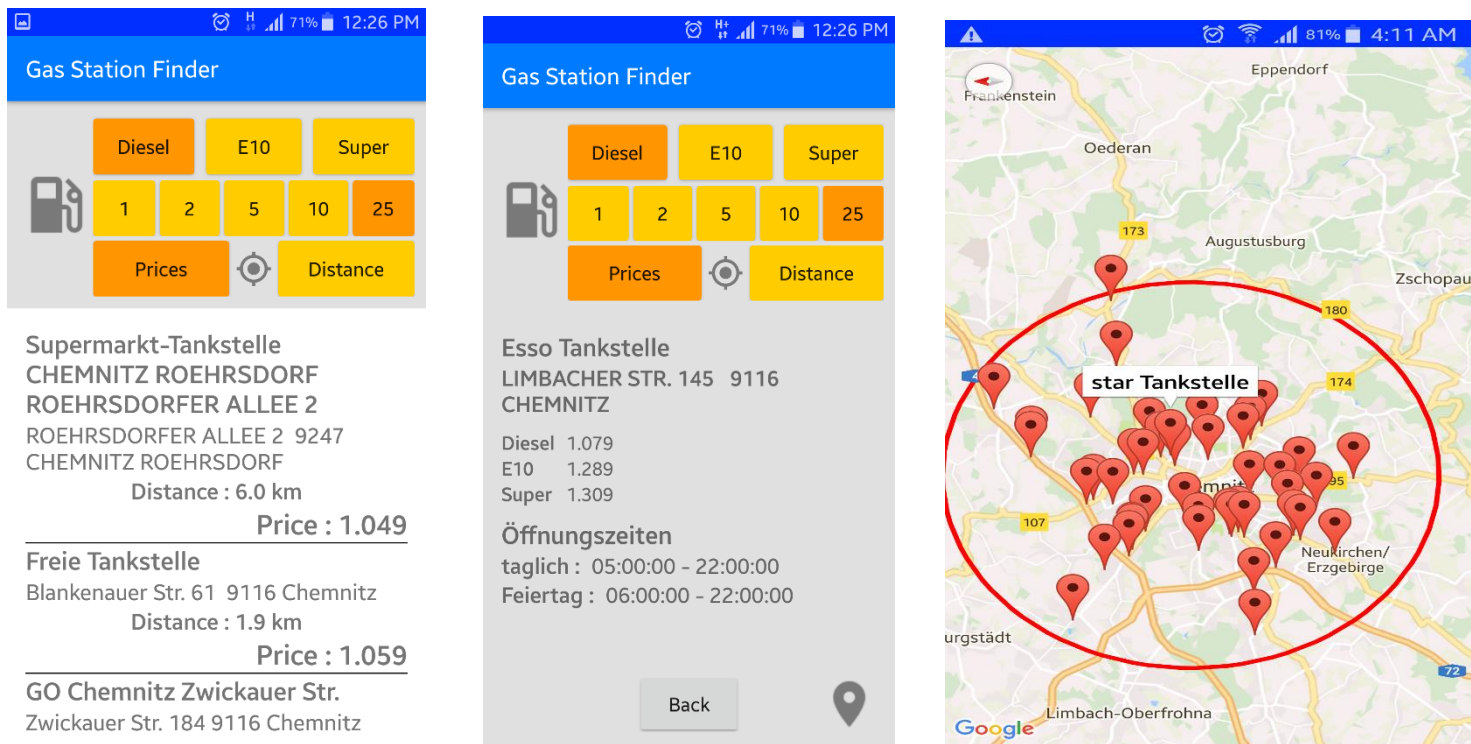


# Gas Station Finder

## 8. MOBILE TECHNOLOGIES

### 8.1. GAS STATION FINDER IN ANDROID

We have designed an Android program for the project. An android program designed in Android Studio IDE. For better User experience, the buttons and text view simplified as much as possible.



**Figure 14:** Gas Station Finder

Mainly, it uses a bunch of technologies. Model – View – Presenter approach has been selected for this application, which is a slightly derived approach from Model – View – Controller (MVP) an aspect of architectural pattern. First of all, REST Request send to the Tankerkönig API to get station information. We can choose the radius via the button numbered 1, 2, 5, 10, 25. That shows us the radius of our searching circle.

The end user can search the prices and distance in ascending order. We can use the buttons named Prices and Distance for this purpose. Then it represents some parameters related to gas station name, address, distance, and price.

If we would tap one of the stations that has listed, we will see a screen like in the middle of image Figure 1.6. We can see all prices (Super, E10, Diesel) regarding station. We can use the button named back to

# Gas Station Finder

return our main screen or we can use the button shaped rain drop. If we tap this, we will see the screen of left-most in Figure 1.6. You will see the name of the station on the marker.

As a consequent, our application is easy-to-use for the end-user who wants to gain benefit from it. We are planning to upload this application to Google Store.

## 8.2 ANDROID TECHNOLOGIES IN THE PROJECT

### Dagger 2

It is a dependency injection framework. It is based on Java Specification Request. It uses code generation and is based on annotations. The generated code is very easy to read and debug. Instead of having our objects creating a dependency or asking a factory object to make one for them, we pass the needed dependencies in to the object externally. One of the major advantages of dependency injection is that it can make testing really easier [12].

### Retrofit

Retrofit is a REST Client and a type-safe HTTP client for Android and Java by Square. It makes relatively easy to retrieve and upload JSON (or other structured data) via a REST based Web Service. In Retrofit you configure which converter is used for serialization [13]. The type of data are typically JSON and GSON but we can add custom converters to process XML or other protocols. We can consider this library like small-packed Express Framework. Retrofit would parse URL via annotations and it uses the OkHttp library for HTTP Request. This technology represents the Model part of Model-View-Presenter approach.

### RxJava

In most Android applications, we are reacting to user interactions (clicks, swipes and etc.) while doing something else in the background (networking). Orchestrating all of this is a hard thing and could quickly turn into an unmanageable code mess. In addition, RxJava presents us a new programming paradigm called reactive programming. If we would give an example about this, let assume the following scenario we have "Send a request to a Tankerkönig API over network and after that completes start fetching its station info and its prices at the same time" [14]

If we dissect this, there will be these main parts that all happen in the background:

- 1) Fetch a station from API.
- 2) Fetch price information related to the station at the same time.
- 3) Combine results from both responses into one.

To do same in Java SE and Android, we are going to need to use: [14]

1. Different AsyncTasks



# Gas Station Finder

2. Create a Semaphore structure, that will wait until both of the requests (station and price) will complete.
3. Object – Level fields to store the results.

It can be already seen, it provides us a functional programming paradigm and asynchronous non-blocking IO.

## Butterknife

It is a binding library that uses annotation in User Interface Side and it generates code for us. It makes your code less and more clear. It is time-saving activity when we use this library. To avoid repetitive code just like “findViewById(R.id.yourview)” in XML pages, the library helps us to binds fields, method, and views for us [15].

## 9. Conclusion:

We used Node.js to implement our own API and HTML5, CSS, Javascript to implement our backend. In this project, we get known to lots of API technology and learn how to implement it. In addition, we learn how to work together as a team.

There are lots of challenge of this project. Tankerkönig-API can't support too many requests in a period of time. In the user interface, we provide a better experience to solve this problem, so we utilized Drawing Manager Google API. Users can draw a circle according to their Location and Radius. In the meanwhile, we try to save data to database when users request the same data, it can directly get from Database. It has lots of challenges such that how can server get know the price is changed or are they some overlay of gas stations.

Thank for everyone who help us for the project.

## 10. REFERENCES

- [1] Digital Ocean, How to Create a New User and Grant Permissions in MySQL. [Online] Available: <https://www.digitalocean.com/community/tutorials/how-to-create-a-new-user-and-grant-permissions-in-mysql>, Last Access: (29.06.2017)
- [2] Mike Cantelon, Marc Harter, T.J. Holowaychuk, Nathan Rajlich, Node.js in Action pp. 72
- [3] Stackoverflow Question, Aren Li [Online] Available: <https://stackoverflow.com/questions/36766696/which-is-correct-node-js-architecture>, Last Access: (26.07.2017)
- [4] Mike Cantelon, Marc Harter, T.j. Holowaychuk, Nathan Rajlich. Node.js in Action. Pp. 94.
- [5] Pro Node.js for Developers, Colin J. Ihrig pp. 189
- [6] Tutorialspoint, Node.js – Express Framework [Online] Available: [https://www.tutorialspoint.com/nodejs/nodejs\\_express\\_framework.htm](https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm), Last Access: (29.06.2017)
- [7] Roman Shtylman, O'reilly Safari [Online] Available: <https://www.safaribooksonline.com/blog/2014/03/10/express-js-middleware-demystified/>, Last Access: (29.07.2017)
- [8] Linux Foundation [Online] Available: <https://nodejs.org/en/docs/guides/anatomy-of-an-http-transaction/#method-url-and-headers> Last Access: (29.07.2017)
- [9] Session 1, Why MySQL pp. 8. [Online] Available: <http://catdir.loc.gov/catdir/samples/wiley031/2001093437.pdf>
- [10] Github documentation, MySQL Connector/Node.js with XDevAPI [Online] Available: <https://github.com/mysql/mysql-connector-nodejs>, Last Access: (29.06.2017)
- [11] Data Realm, Five Advantages and Disadvantages of MySQL: <https://www.datarealm.com/blog/five-advantages-disadvantages-of-mysql/>, Last Access: (29.07.2017)
- [12] Vogella, Using Dagger 2 for dependency injection in Android – Tutorial, [Online] Available: <http://www.vogella.com/tutorials/Dagger/article.html>, Last Access: (29.07.2017)
- [13] Lars Vogel, Simon Scholz, David Weiser, Using Retrofit 2.x as REST Client – Tutorial, Vogella, <http://www.vogella.com/tutorials/Retrofit/article.html>, Last Access: (28.07.2017)
- [14] Tadas Subonis, Feedpresso, Why should use RxJava in Android a short introduction to RxJava, [Online] Available: <http://blog.feedpresso.com/2016/01/25/why-you-should-use-rxjava-in-android-a-short-introduction-to-rxjava.html> Last Access: (26.07.2017)
- [15] Parany Patel, ButterKnife – A viewbinding library for Android (Beginner Guide), <https://medium.com/@pranaypatel/butterknife-a-viewbinding-library-for-android-beginner-guide-fd92caf8e505>, Last Access: (25.07.2017)