DTT-Assignment report

**Submitted by,**

**NEBIN THOMAS**

[**nebin2001mas@gmail.com**](mailto:nebin2001mas@gmail.com)

**+37126324401**

**CONTENTS**

1. Database Setup……………………………………………………………………………………………………… Page 3
2. API Configuration…………………………………………………………………………………………………… Page 5
3. Final Report…………………………………………………………………………………………………………… Page 8

**DATABASE SETUP**

The DBMS used here is Mysql, which comes with phpMyAdmin in Wamp.

**Tables**

The database consists of 5 tables. The detailed description of the tables is given below:

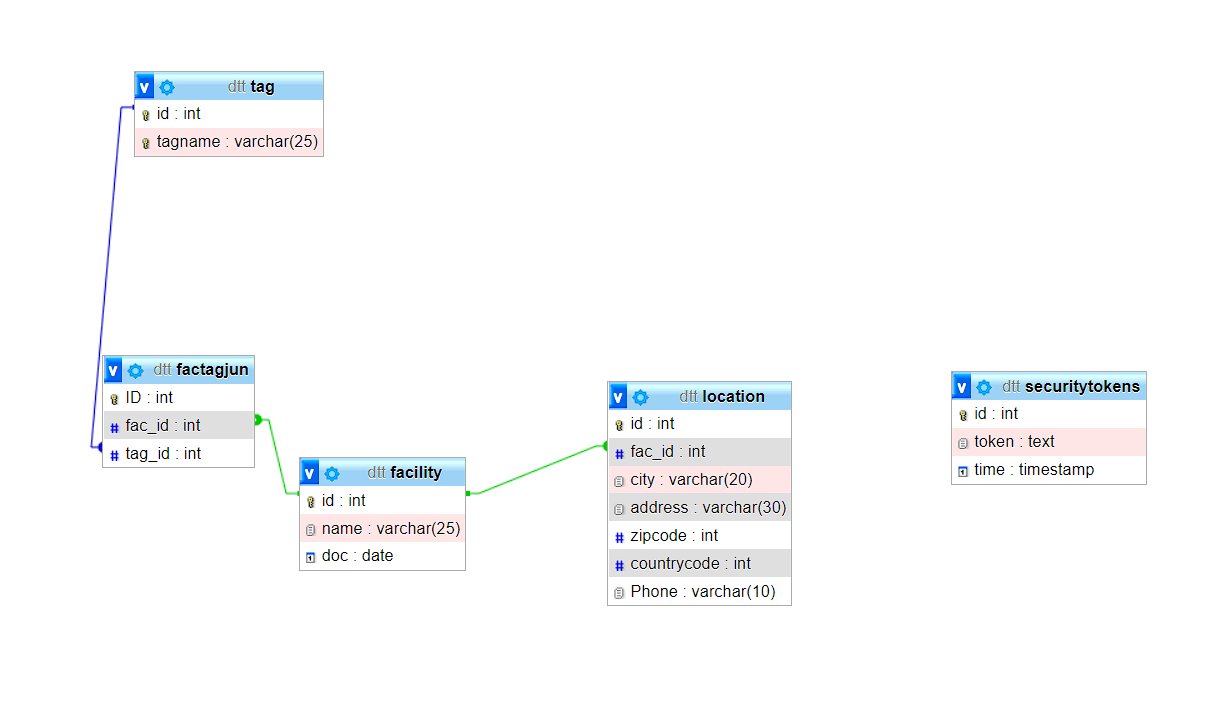
* Facility: This table has columns which are id, name and doc. This tables holds the facility information’s such as the Name and the Date of Creation.
* Tag: This table saves tag names used by the facility. The tagname are unique and reused by each facility.
* Location: This table has columns such as id, fac\_id, zipcode, countrycode, phonenumber. This table store the location of each facility.
* Factagjun: This is acronym for Facility Tag Junction. This table act as bridge between facility and tag. Also, can be called as a junction table.
* Securitytokens: This table does not have any direct relationships with other tables. This table is used to save security tokens for securing the API endpoints.

**Events**

The table has only one event. Its name is tokendeleter, which works every 1 hour and deletes the expired tokens in the database.

**Table Relationships**

The relationships of the table can be easily depicted from the pic.



* Facility->Location Relationship: The relationship can be considered as one to many since in one location there can be more than one facility too.
* Facility->Tag Relationship: The relationship between facility and tag is considered as many to many, since many facilities can have many tags. So that there is an additional junction table called factagjun, which acts as a bridge.

The rest tables do not have any direct relationships.

**API CONFIGURATION**

This API performs basic CRUD functionalities and also additionally a search functionality too. This API is secured with an authentication token which valid only for 1 hour. In-order to perform any of this functionality one must generate the token and pass it as the bearer token in the header. The usage manual of the API is already given as a documentation in the post man collection. I will brief all about this here too.

**Authentication**

URL: <http://localhost/dttsetup/auth>

PARAMS: No params is needed

Response: 200 OK

{

    "token": "b9348984d6ada378649bbb9bafb3ded8"

}

**Read**

URL: http://localhost/dttsetup/read

PARAMS: No params is needed, since it reads all the facilities and tags

Response: 200 OK

[

    {

        "id": "11",

        "Facility\_Name": "newfacility",

        "Date\_of\_Creation": "2022-04-29",

        "Tags": [

            "cool",

            "dude",

            "fun"        ]

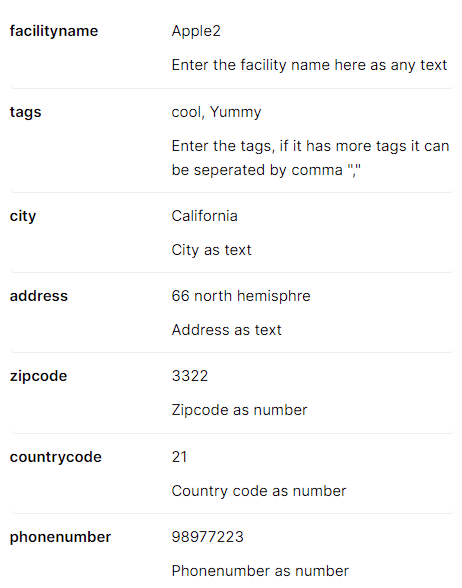
    }

]

**CREATE**

URL: http://localhost/dttsetup/create/

PARAMS: The params such as the keys and values is as of follows



Response: 201 Created

{

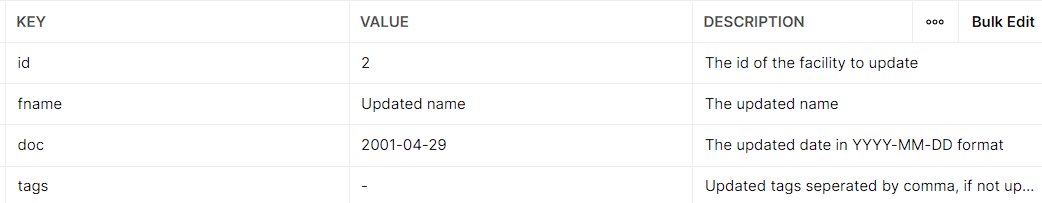
    "Success": "Facility location and tags is successfully added"

}

**Update**

URL: <http://localhost/dttsetup/update/>

Params: The parameters and its description can be seen here.

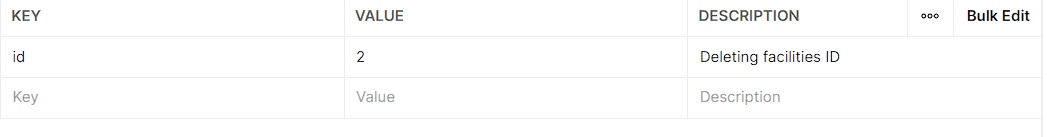


Response: 200 OK

**Delete**

URL: <http://localhost/dttsetup/delete/>

Params: The facility can be deleted with the ID



Response: No Content

**Code Description Report.**

First, I wanted to refurbish my knowledge in REST API. For that I have watched several videos in YouTube, out of which this one <https://www.youtube.com/watch?v=7YcW25PHnAA> helped me to understand the concept more clearly and very well. I also checked some of the sites which explains HTTP responses and other CRUD creations with REST API. To under some concept about the HTTP response codes I made use of the IBM site which we can see here <https://www.ibm.com/docs/en/cics-ts/5.2?topic=concepts-status-codes-reason-phrases#dfhtl_httpstatus>.

After wards I grasped and understood the real advantage API based on REST. Such that we have completely created a Backend in php or any other language. We can send an HTTP request from any environment, which means it does not matter whoever the client is, if client can use and manipulate JSON. We can also say that json can be considered as a universal language between servers and client. So, I decided to create a simple CRUD mechanism from scratch. For that I have made use of the tutorial <https://www.positronx.io/create-simple-php-crud-rest-api-with-mysql-php-pdo/>

From this tutorial I have got a clear knowledge regarding the API properties and concepts, mainly in php. I came to know more about headers and what were it’s purposes. But even though I have made a CRUD with facilities and tags, I came to a realisation that my project had many issues. The code was not consistent, robust and also it was very hard to be modified either. So, I decided to learn more advanced because I wanted to extend my knowledge from the basic methodology. So, I then checked the assignment setup which was provided by the email, I have setup the assignment in my localhost easily and smoothly because of the clear information provided. After going through the classes and its properties I have came to know that they are using special router for routing.

So, I again researched about the router and its advantages. Afterwards I came to know about BramUs router, which was currently using in the assignment setup. I checked the Github readme of the BramUs router which is here <https://github.com/bramus/router>, to understand it better. Then after more checking I came to the assignment setup which was provided already has the database connection functionalities. So, I did not need to remake another class, Instead I inherited the properties of the assignment setup for my API. So, I created an API class which inherits all the assignment properties and implemented all the CRUD, and Search functionality there. Since, I had gained a significant knowledge regarding SQL from the university, I haven’t faced that much hurdles while creating the query. The next process to add security mechanism to the API. At first, I didn’t have clear idea to secure the API. So, I again make use of the google to help me. And I found this video <https://www.youtube.com/watch?v=DanUVSlOSQQ> very helpful for understanding about the token.

So, rather than using already created Firebase/Jwt or 0Auth token, I decided to make my own security mechanism. For that I have modified my database, and created a table which holds the security tokens with an expiry time. Also, I have added an event in the database, which automatically deletes the expired after every hour. So, what I have created is that, I will generate a random hash key and store the database. For now, I have just created simple one, if necessary, I can add additional login feature for receiving the token. So, afterwards we need to pass this token in the header for achieving the goal. So, with this I was successfully able to secure the API.

So, in a nutshell I have gained a significant knowledge regarding Backend development process with this assignment. I got really interested and I was really enjoying doing this assignment. I would like to also extend my knowledge and vision with gaining experience from great firm like D-TT. This assignment was not just an assignment for me it was an addition to my knowledge and my programming skills. If, I made any errors, I would be really happy to correct it, and learn from my mistakes. So I will be waiting to hear from you.