Github: <https://github.com/limooo196/VDObject-API/tree/main>

**VDObject-API**

This is Node JS application that a web-server that handles certain objects with various HTTP request that connects with MySQL database.

To run this application  
- adjust the .env files to match your database  
- npm install  
- nodemon app.js

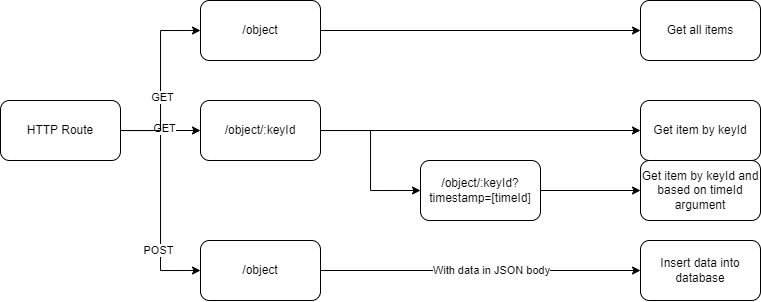
**File structure**

**App.js**app.js is the main file of the application, where it will handle the main route of the server and the listener. For this web server I turned on **bodyParser JSON** and **urlencoded**, so our application can submit through JSON body and also can insert argument into the url.

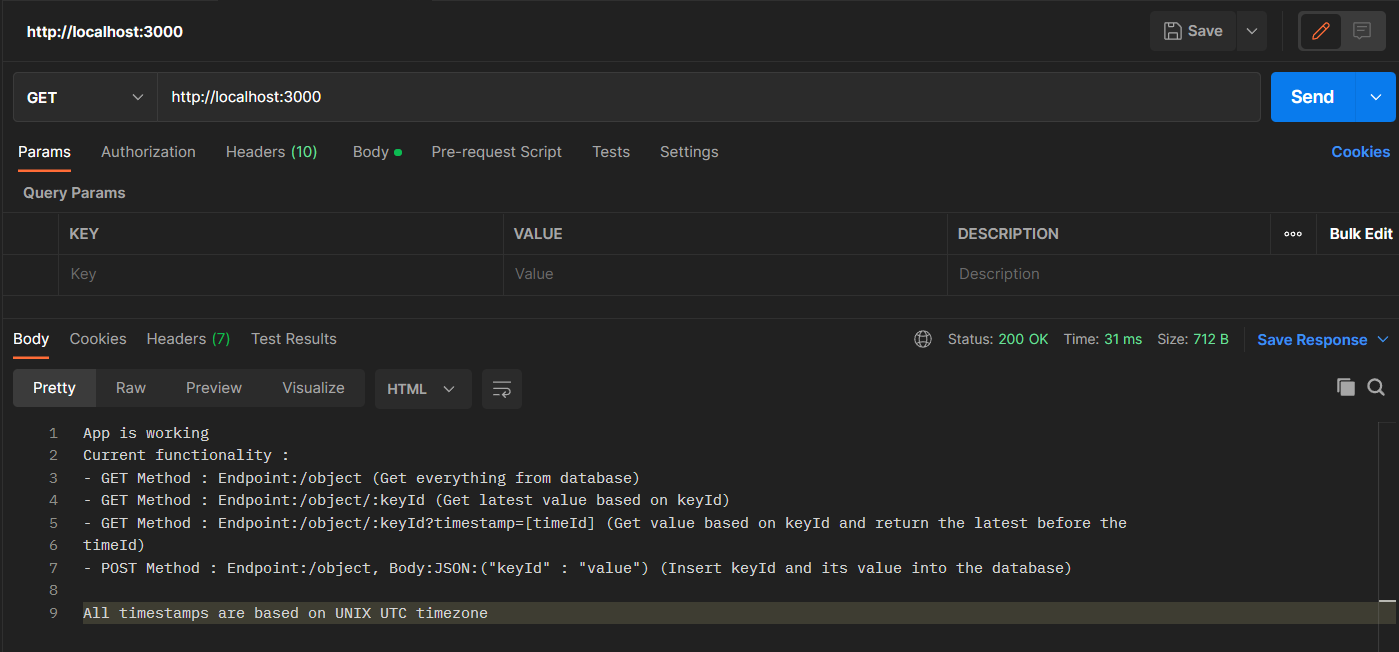
It will direct the “/object” route to the objectRoute, “/” which is the root url, it will explain some of the functionality and endpoint and last “\*” it will return some manual on the methods and paths that is functional in case user entered the wrong or undefined path.

**DB.js**Db.js Is a file that handles connection to the mysql database, where it will create connection to the mysql server. **Model/object.js**this file create the object before we store it into the database, it has the basic constructor and also save function where, it will execute mysql query to insert the object user entered into the database. **Controller/objectController.js**in this file we have 3 main function, where the most logic of the server resides. First function is to create the object and call the save function to insert the data into the database. Second function is to return all items in the database. Third item is where the server handles the Get Method by keyId or timestamp.

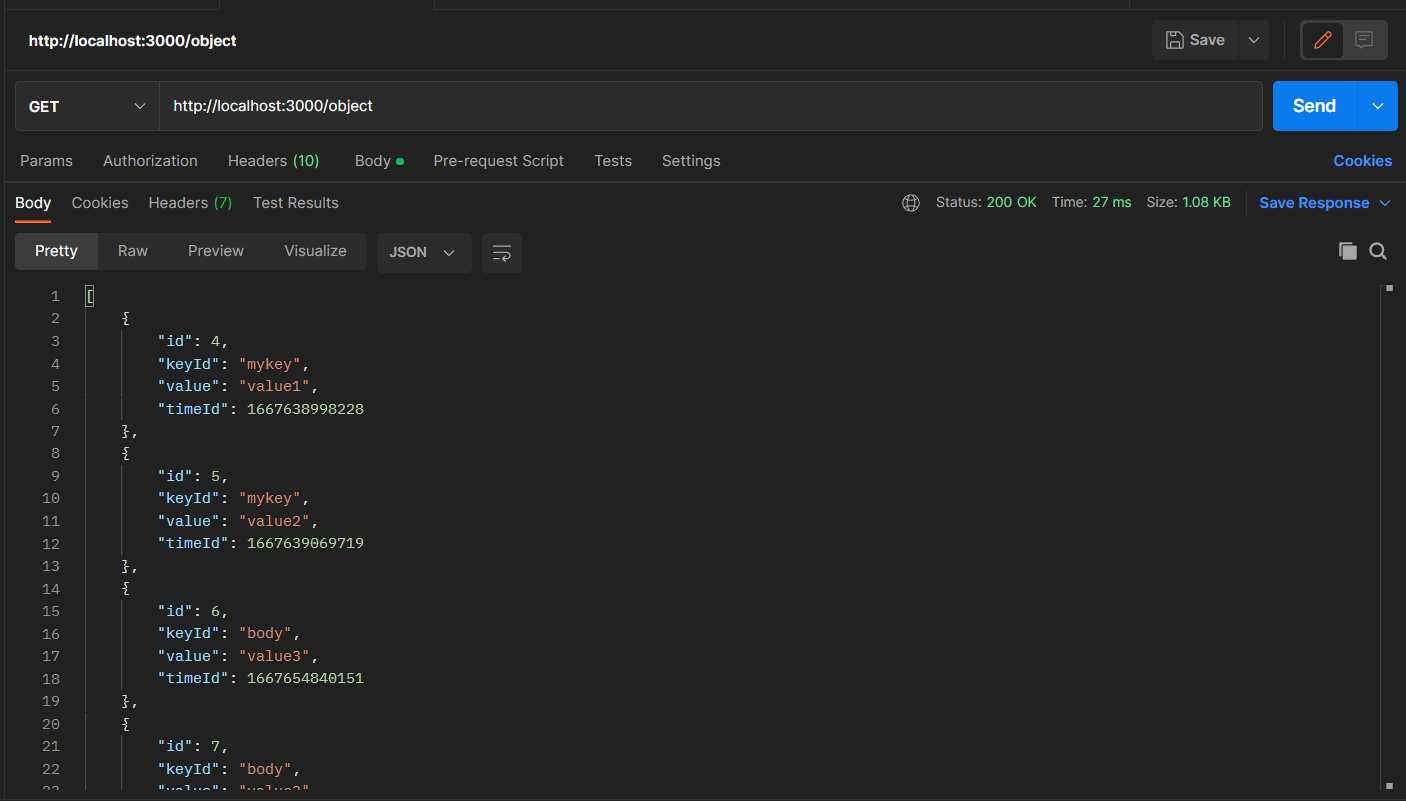
**Route/objectRoute.js**This file handles all the HTTP methods of “/object” which redirected from the app.js. in this route we only defined two paths with HTTP methods, first path is get method of the “/” path where it will return all items in the database and post method of this path will create new object if user entered the json into the body. Second path is “/:keyId”, this path is handling the keyId user entered and also it handles the timestamp if user entered the timestamp arguments.



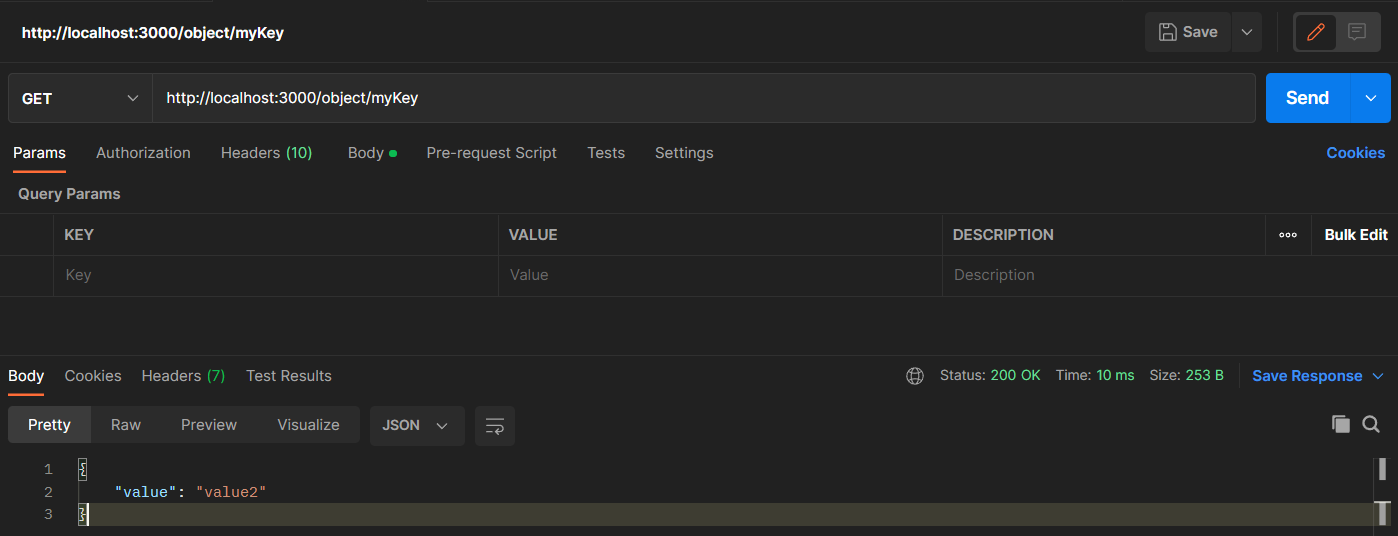
**Functionality**

**Get Method in “/”**

**Get Method in “/object”**

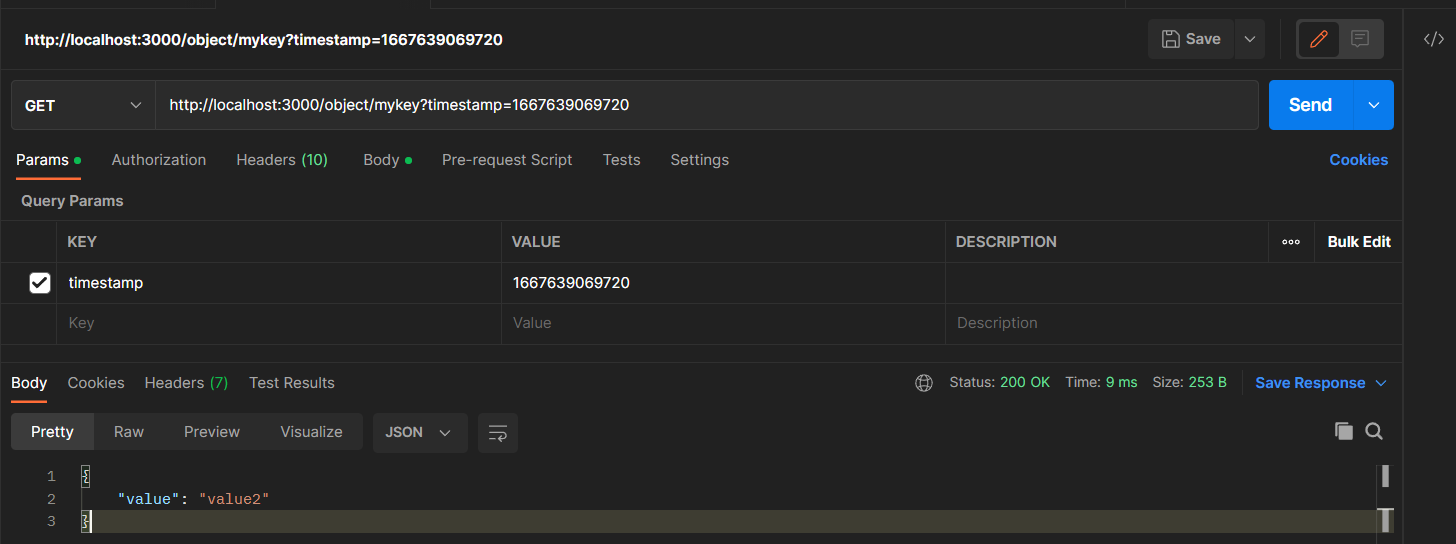
****

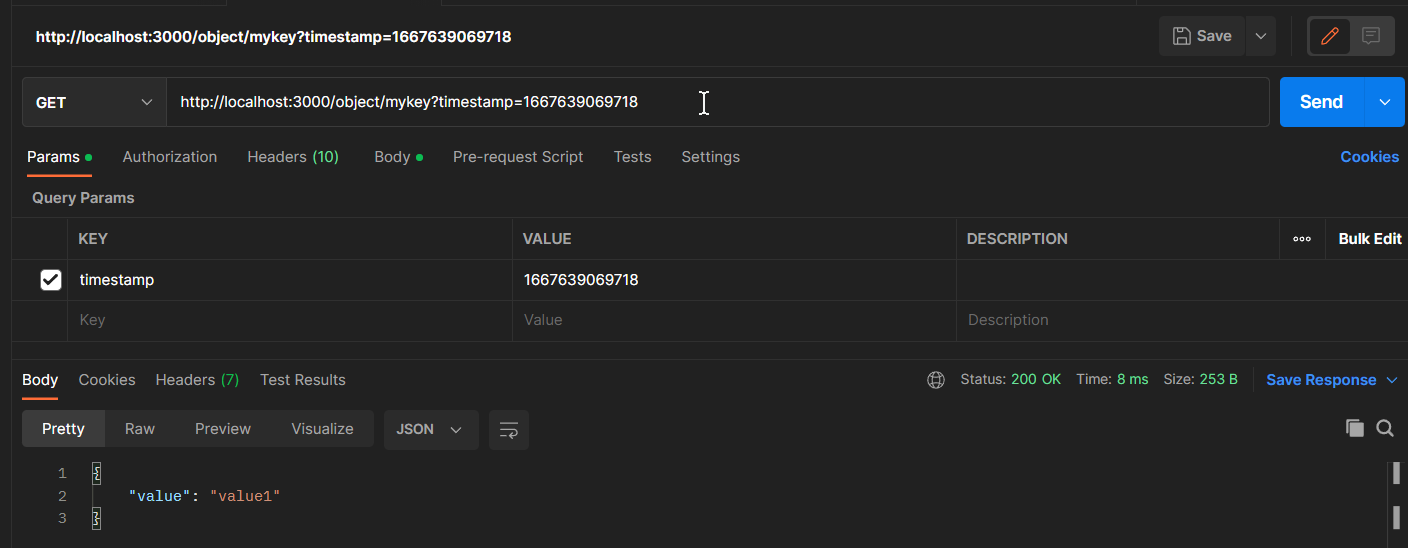
**Get Method in “/object/:keyId”**

****

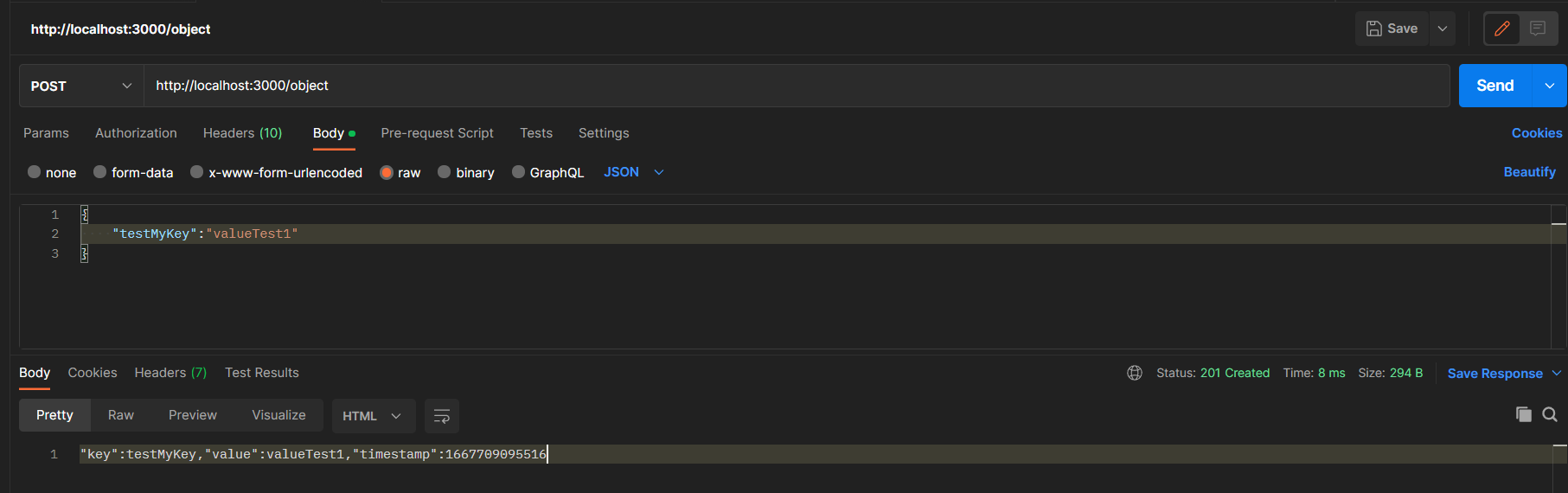
**Get Method in “/object/:keyId?timestamp=[timeId]”**

“value2” is created in 1667639069719

****

****

**POST Method in “/object”**

****

Explain assumptions