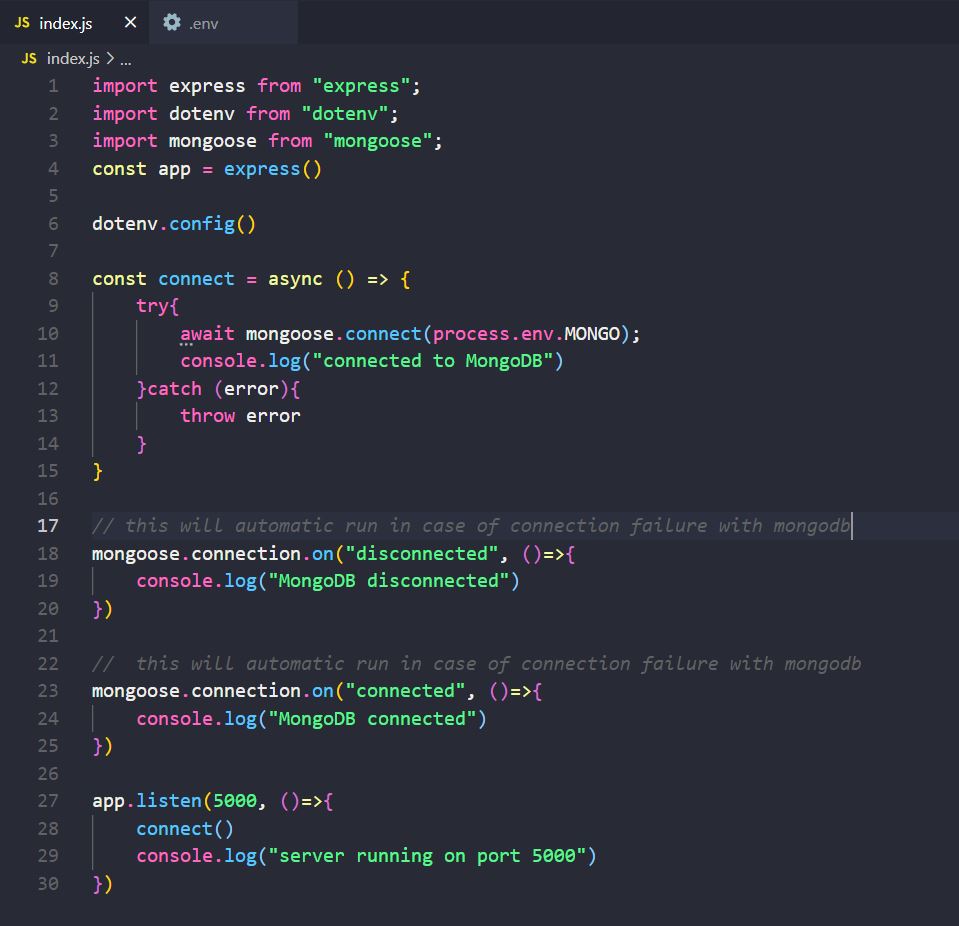
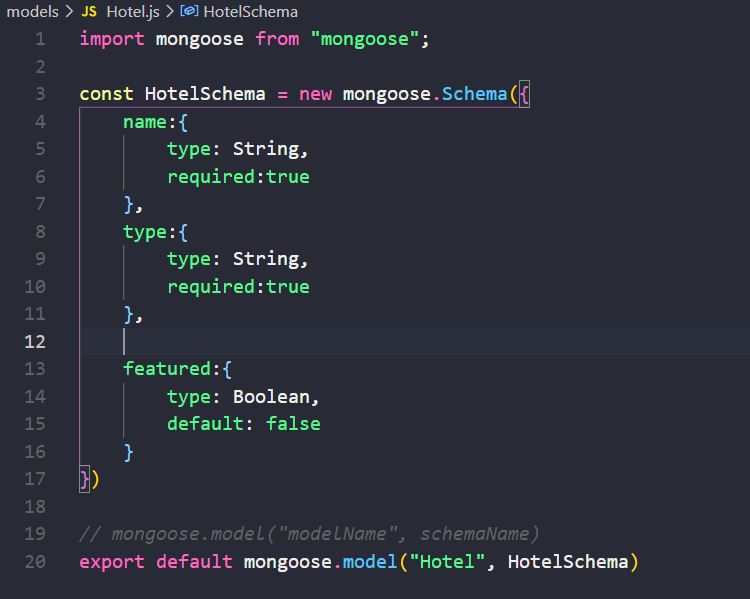
**Backend Development**

**Connection with mongodb:**

****

* dotenv is used to hide secrete data like id, password…

**Schema and model :**

****

**Send data to Mongodb using postman**

Json can’t be sent to express server, to send json data use middleware

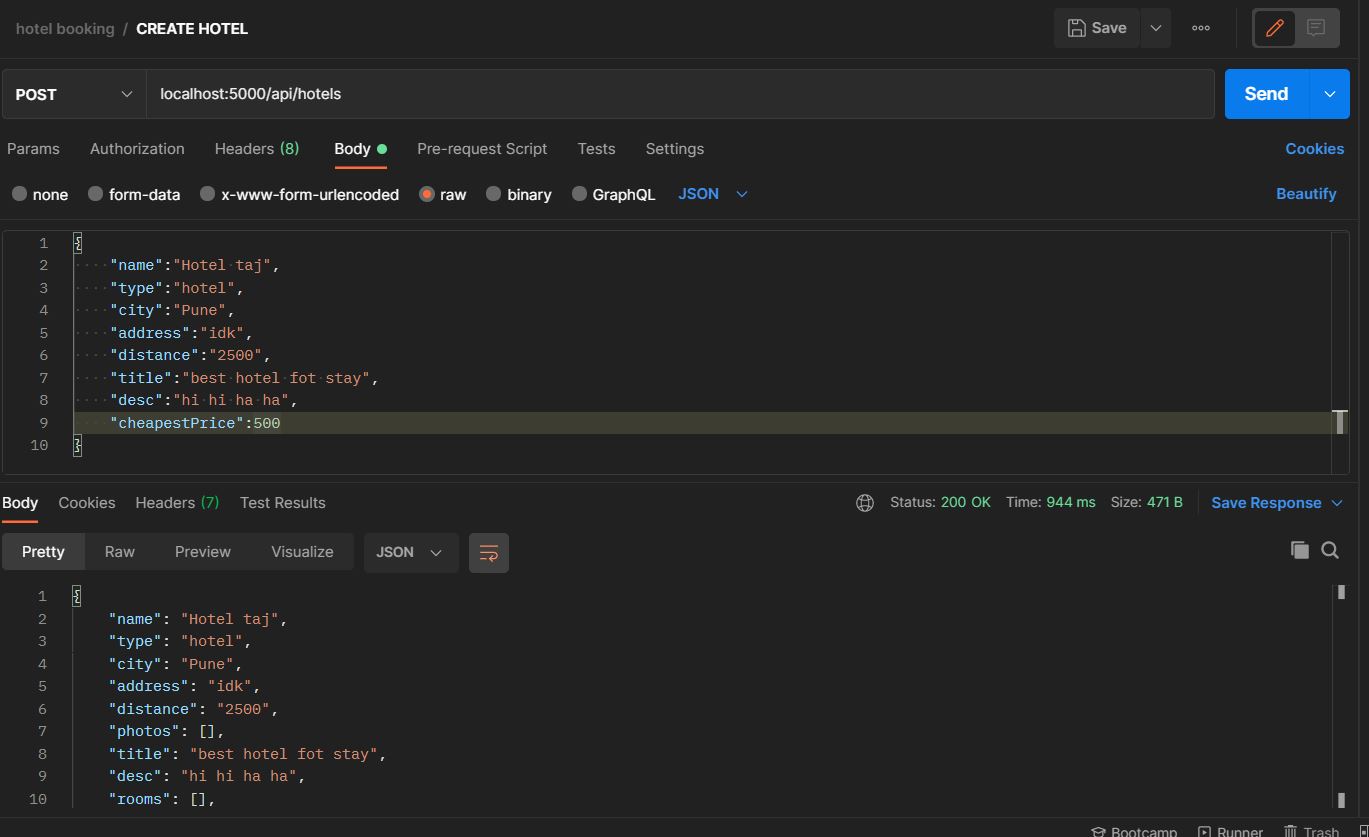
* app.use(express.json())



**CREATE in MongoDB or Send data**

****

**Sending through postman**

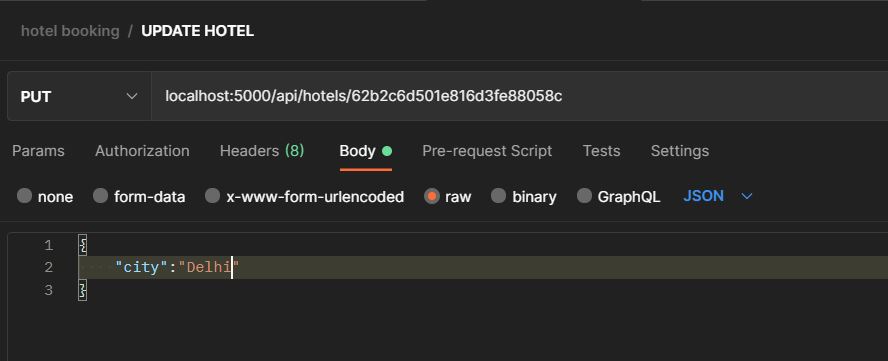


Update in mongodb

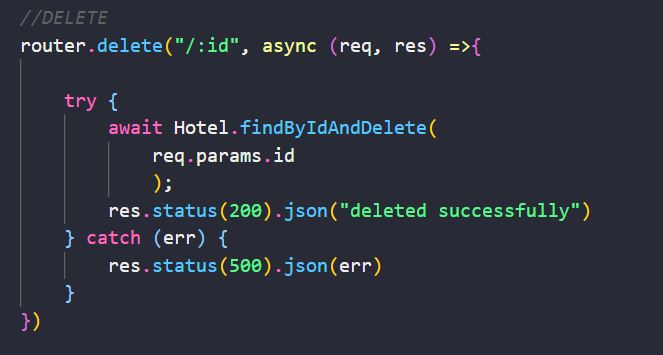


Note: if you don’t write {new:true} it will not affect database it is only return new copy to console

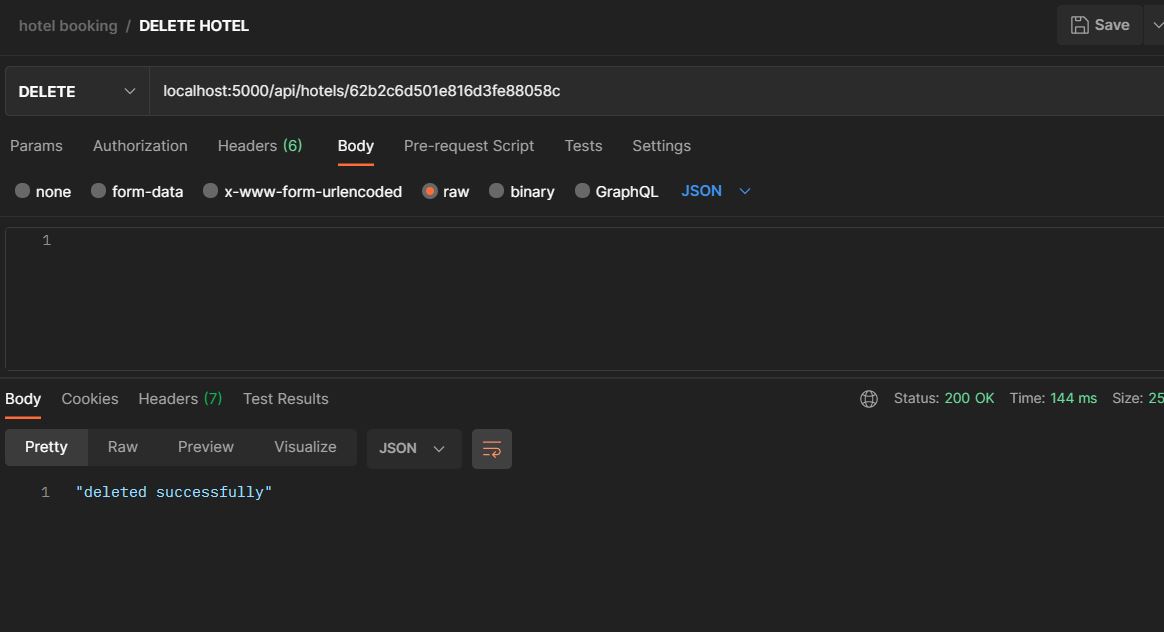
Updating through postman



Delete in mongodb



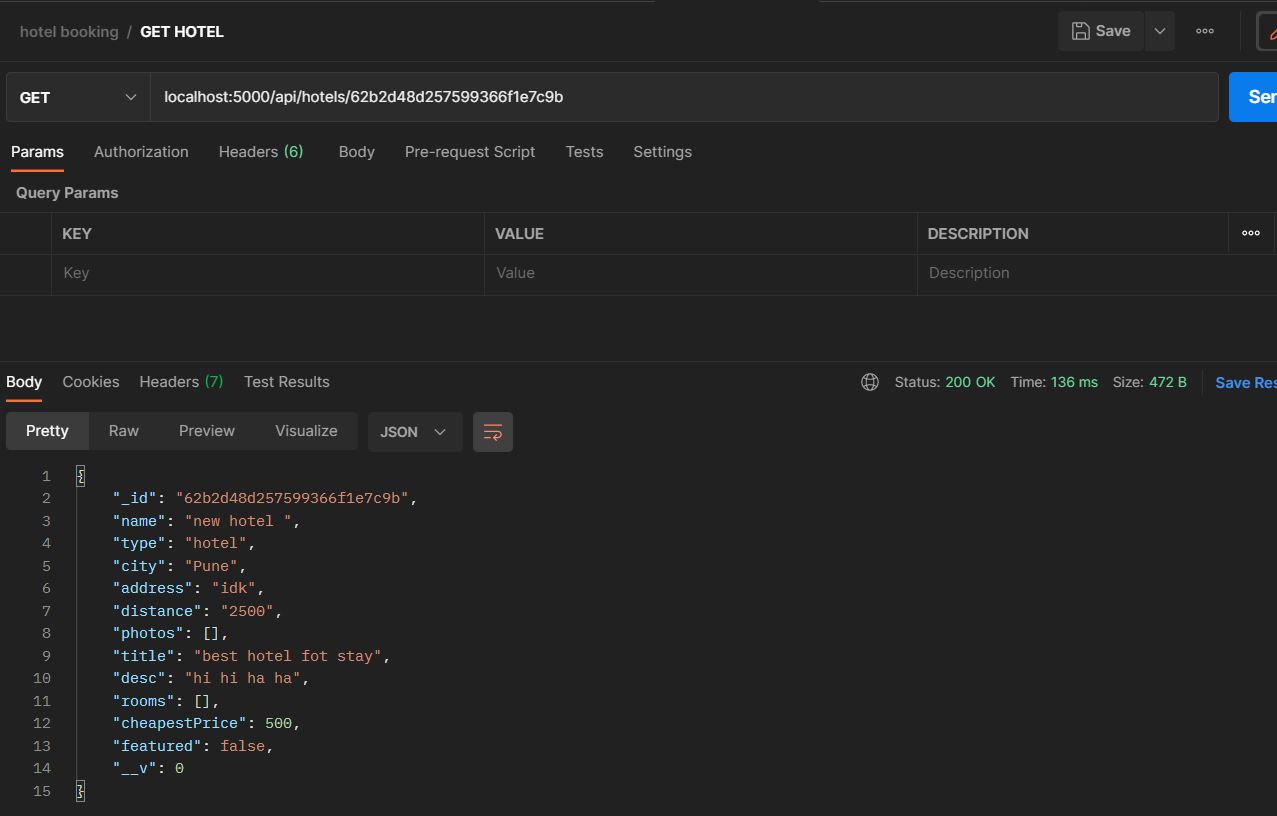
Deletion data through postman



GET or reading data in mongodb



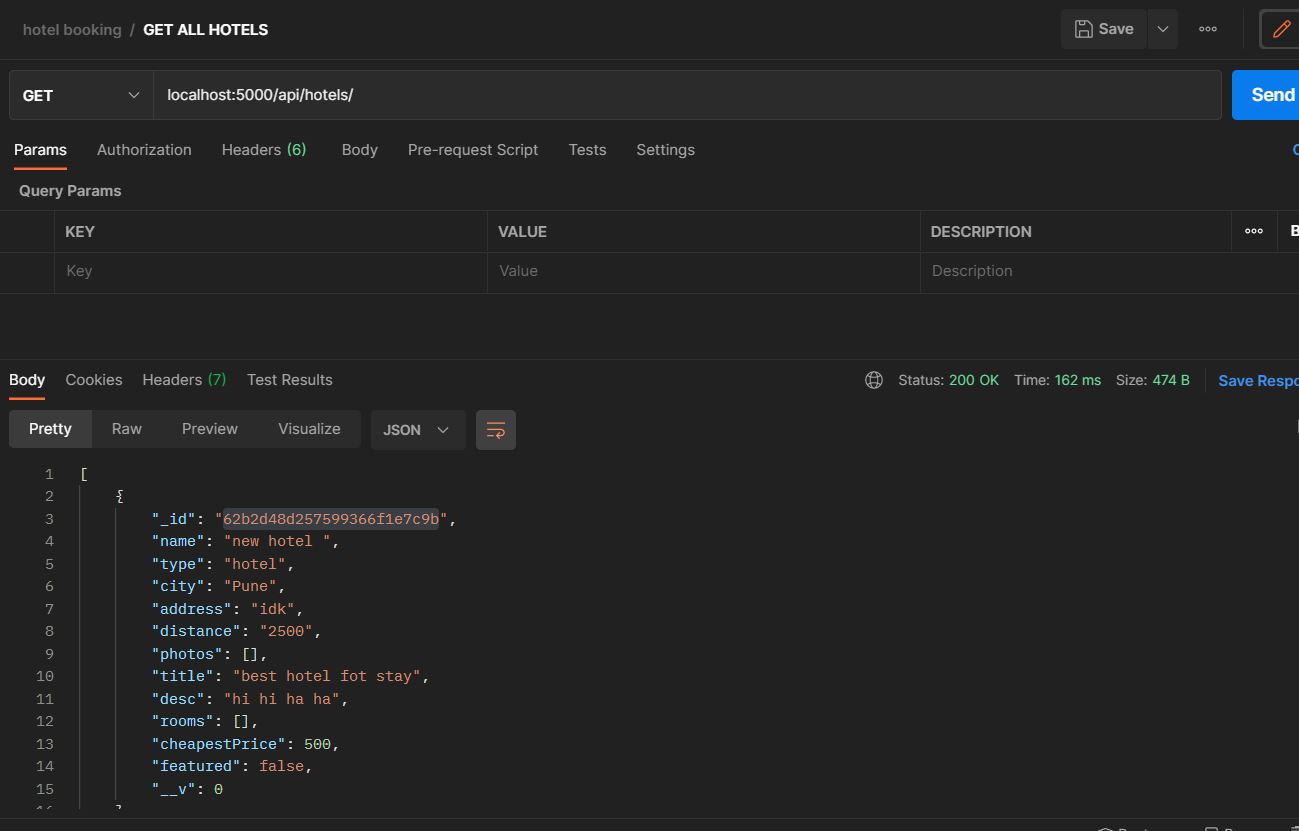
Reading data through postman



GET ALL or reading all data in mongo db:



GET ALL using postman



**Password Encoding:**

Install bcryptjs: yarn add bcryptjs or npm i bcryptjs

Import: import bcrypt from “bcryptjs”

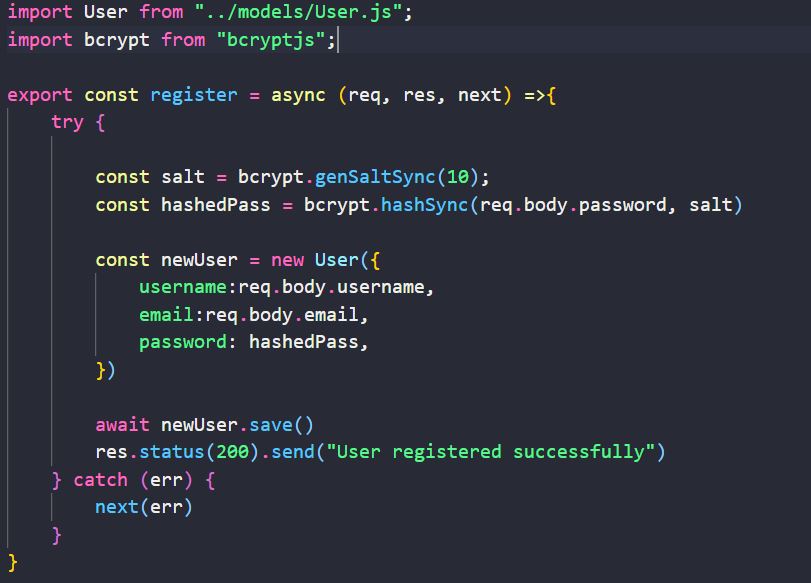
Use:

const salt = bcrypt.genSaltSync(10);

cons hashedPass = bcypt.hashSync(req.body.password, salt);

now **req.body.password** has ecrypted and become **hasedPass**

**Registration with hashed password:**



Check hashed password is same as user requested :



req.body.password ----- > password requested by user for sign-in

user.password --------- > password stored in db

**JWT with cookie:**

Install jwt – **yarn add jsonwebtoken**

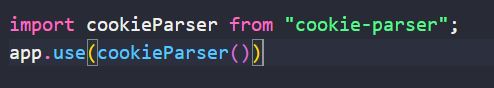
Create jwt token: 

Jwt.sign({data}, secrete\_key)

Use cookie:

Install cookie – **yarn add cookie-parser**

* Import cookie in main file(index.js) and use as middleware



* Send cookie with response



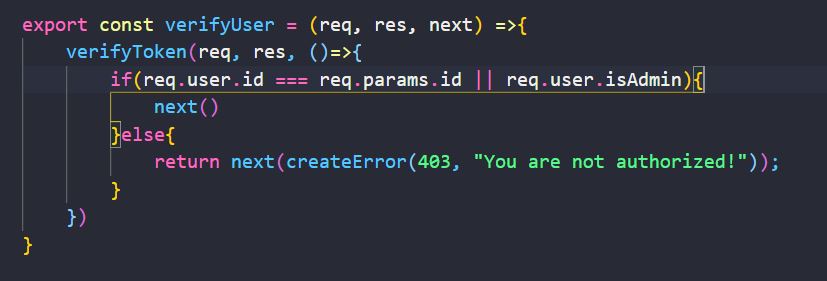
**Login with hashed password:**

****

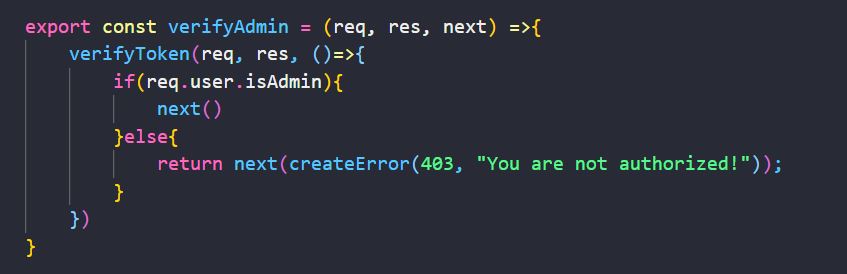
**Verify token :**

****

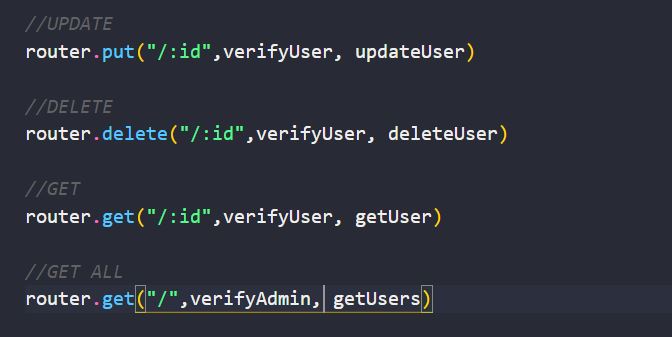
**Verify user to delete any account (they can delete only their account):**

****

**Verify admin (admin can delete any account) :**

****

**Use these verifications as middleware as below:**

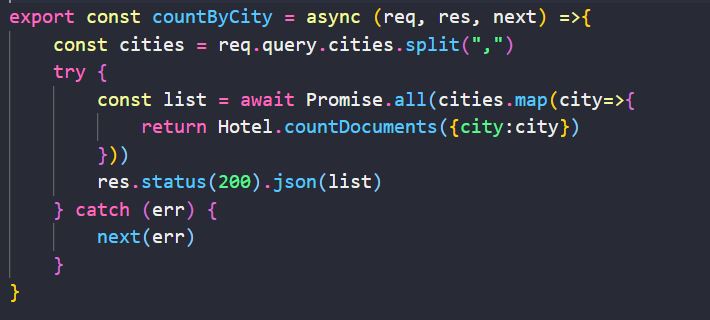
****

**Deal with query:**

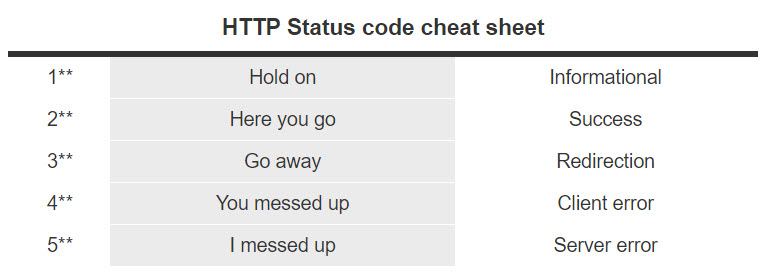
Query starts with question-mark “**?**”, then **key1=value1,value2&key2=val1,val2**

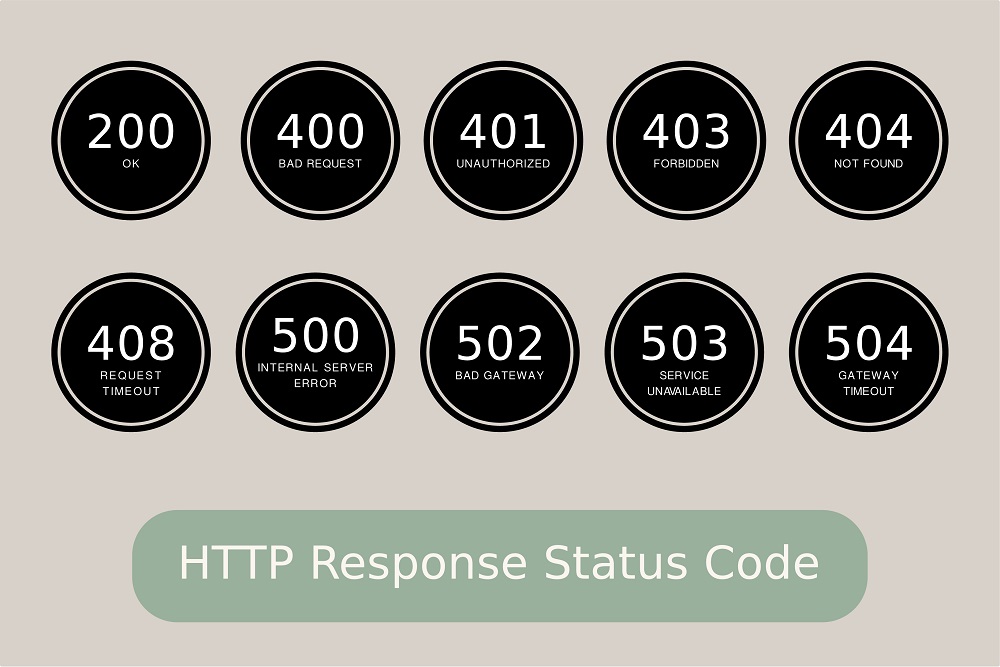
****

Accessing query:

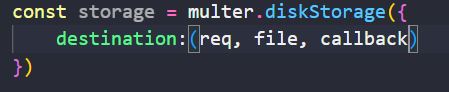


**HTTP Status Code –**





Multer





Important:

* findOne
* findByIdAndDelete vs findByIdAndRemove
* req.params vs req.query
* send vs json
* put vs fetch
* schema vs model

**MongoDB Important –**

* findOne({username: req.body.username})
* findById(req.body.id)
* findByIdAndUpdate(req.params.id, {$set: req.body}, {new: true})
* findByIdAndDelete(req.params.id)
* deleteMany({username: req.params.username})

**req.query vs req.params**

[req.params](http://expressjs.com/en/api.html#req.params) contains route parameters (in the path portion of the URL), and [req.query](http://expressjs.com/en/api.html" \l "req.query) contains the URL query parameters (after the ? in the URL).

**PUT VS PATCH**

PUT is a method of modifying resource where the client sends data that updates the entire resource. It is used to set an entity’s information completely. PUT is similar to POST in that it can create resources, but it does so when there is a defined URI. PUT overwrites the entire entity if it already exists, and creates a new resource if it doesn’t exist.

Unlike PUT, PATCH applies a partial update to the resource.

This means that you are only required to send the data that you want to update, and it won’t affect or change anything else. So if you want to update the first name on a database, you will only be required to send the first parameter; the first name.

**res.send vs res.json vs res.end** [more](https://medium.com/gist-for-js/use-of-res-json-vs-res-send-vs-res-end-in-express-b50688c0cddf#:~:text=But%20the%20main%20difference%20between%20res.json%20and%20res.send,uses%20json%20replacer%20and%20json%20spaces%20application%20settings.)

res.send

- sends response and then end the response

- it also set Content-Type according to data passed into res.send(), if body passed is buffer then Content-type will be application/octet-stream

res.json

* it is identical to res.send() when an array or object is passed, but also converts non-objects to json

res.end

* don’t return any response, it just ends the response
* it can be useful if no need to send any response like 404 page