Restaurant Reservation System MongoDB API

Muyi Chen

Introduction:

In this case study we aim to construct a mongoDB based API system that takes a jason instance from the client side (simulated with PostMan in development process), process it with Node.js (which acts as the server) and stores it in local mongoDB(the local database).

1. Set up mongoDB

a. Download mongoDB here

https://www.mongodb.com/download-center/community

Remember to install mongo compass, which is the interface showing all database info.

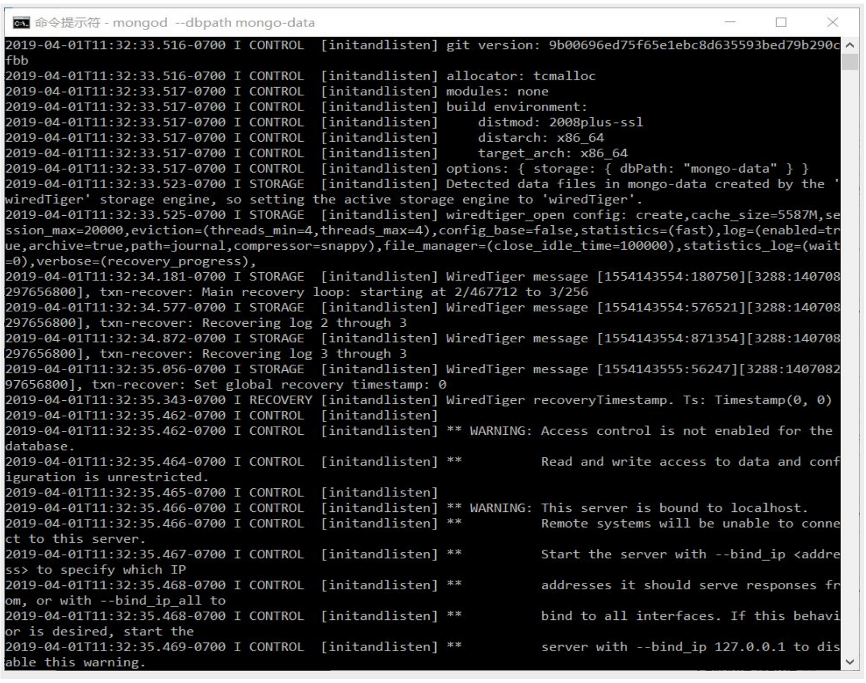
b. Download starter files from github

c. Download Node.js which will be used in the case study

https://nodejs.org/en/download/current/

After installation, use the following command in cmd after changing directory to the folder with server.js starter files:

```
$ npm install
$ mkdir mongo-data
$ mongod --dbpath mongo-data
```



```
命令提示符 - mongod  --dbpath mongo-data
2019-04-01T11:32:33.516-0700 I CONTROL  [initandlisten] git version: 9b00696ed75f65e1ebc8d635593bed79b290c
fbb
2019-04-01T11:32:33.516-0700 I CONTROL  [initandlisten] allocator: tcmalloc
2019-04-01T11:32:33.517-0700 I CONTROL  [initandlisten] modules: none
2019-04-01T11:32:33.517-0700 I CONTROL  [initandlisten] build environment:
2019-04-01T11:32:33.517-0700 I CONTROL  [initandlisten]     distmod: 2008plus-ssl
2019-04-01T11:32:33.517-0700 I CONTROL  [initandlisten]     distarch: x86_64
2019-04-01T11:32:33.517-0700 I CONTROL  [initandlisten]     target_arch: x86_64
2019-04-01T11:32:33.517-0700 I CONTROL  [initandlisten] options: { storage: { dbPath: "mongo-data" } }
2019-04-01T11:32:33.523-0700 I STORAGE  [initandlisten] Detected data files in mongo-data created by the '
wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2019-04-01T11:32:33.525-0700 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=5587M,se
ssion_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=tr
ue,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wait
=0),verbose=(recovery_progress),
2019-04-01T11:32:34.181-0700 I STORAGE  [initandlisten] WiredTiger message [1554143554:180750][3288:140708
297656800], txn-recover: Main recovery loop: starting at 2/467712 to 3/256
2019-04-01T11:32:34.577-0700 I STORAGE  [initandlisten] WiredTiger message [1554143554:576521][3288:140708
297656800], txn-recover: Recovering log 2 through 3
2019-04-01T11:32:34.872-0700 I STORAGE  [initandlisten] WiredTiger message [1554143554:871354][3288:140708
297656800], txn-recover: Recovering log 3 through 3
2019-04-01T11:32:35.056-0700 I STORAGE  [initandlisten] WiredTiger message [1554143555:56247][3288:1407082
97656800], txn-recover: Set global recovery timestamp: 0
2019-04-01T11:32:35.343-0700 I RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2019-04-01T11:32:35.462-0700 I CONTROL  [initandlisten]
2019-04-01T11:32:35.462-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the
database.
2019-04-01T11:32:35.464-0700 I CONTROL  [initandlisten] **          Read and write access to data and conf
iguration is unrestricted.
2019-04-01T11:32:35.465-0700 I CONTROL  [initandlisten]
2019-04-01T11:32:35.466-0700 I CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2019-04-01T11:32:35.466-0700 I CONTROL  [initandlisten] **          Remote systems will be unable to conne
ct to this server.
2019-04-01T11:32:35.467-0700 I CONTROL  [initandlisten] **          Start the server with --bind_ip <addre
ss> to specify which IP
2019-04-01T11:32:35.468-0700 I CONTROL  [initandlisten] **          addresses it should serve responses fr
om, or with --bind_ip_all to
2019-04-01T11:32:35.468-0700 I CONTROL  [initandlisten] **          bind to all interfaces. If this behavi
or is desired, start the
2019-04-01T11:32:35.469-0700 I CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to dis
able this warning.
```

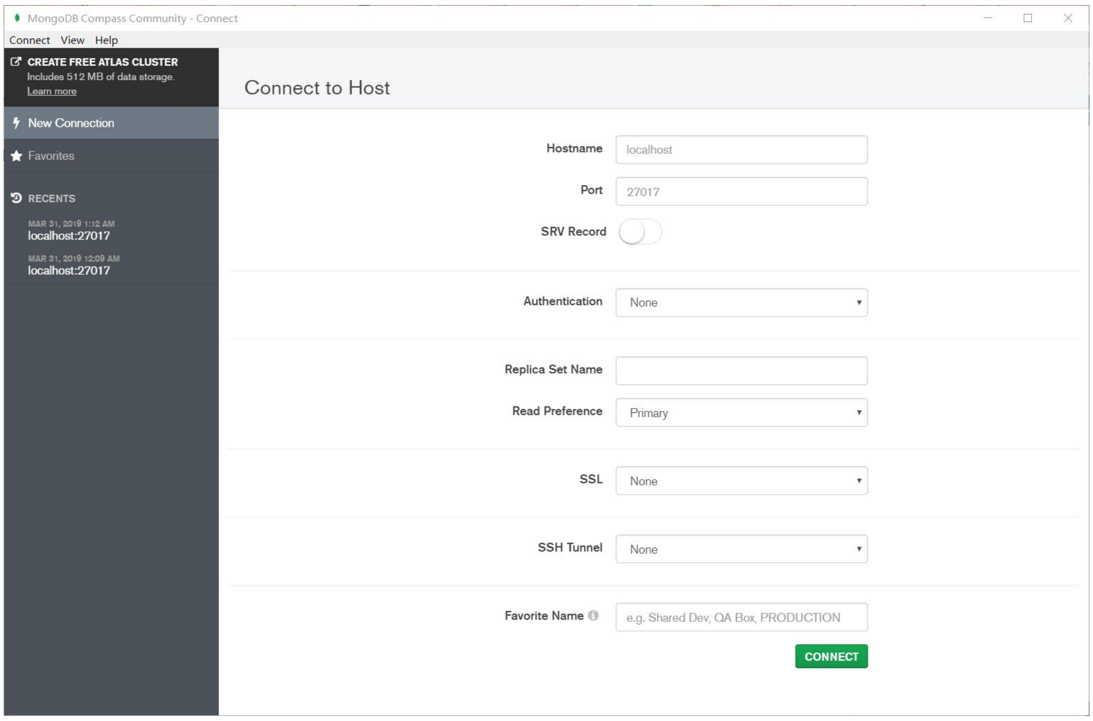
d. Download postMan, which is used to simulate the client side

https://www.getpostman.com/downloads/

PostMan sends jason requests to server.js, which then interacts with database.

e. Connecting to database

i. Start mongoDB database

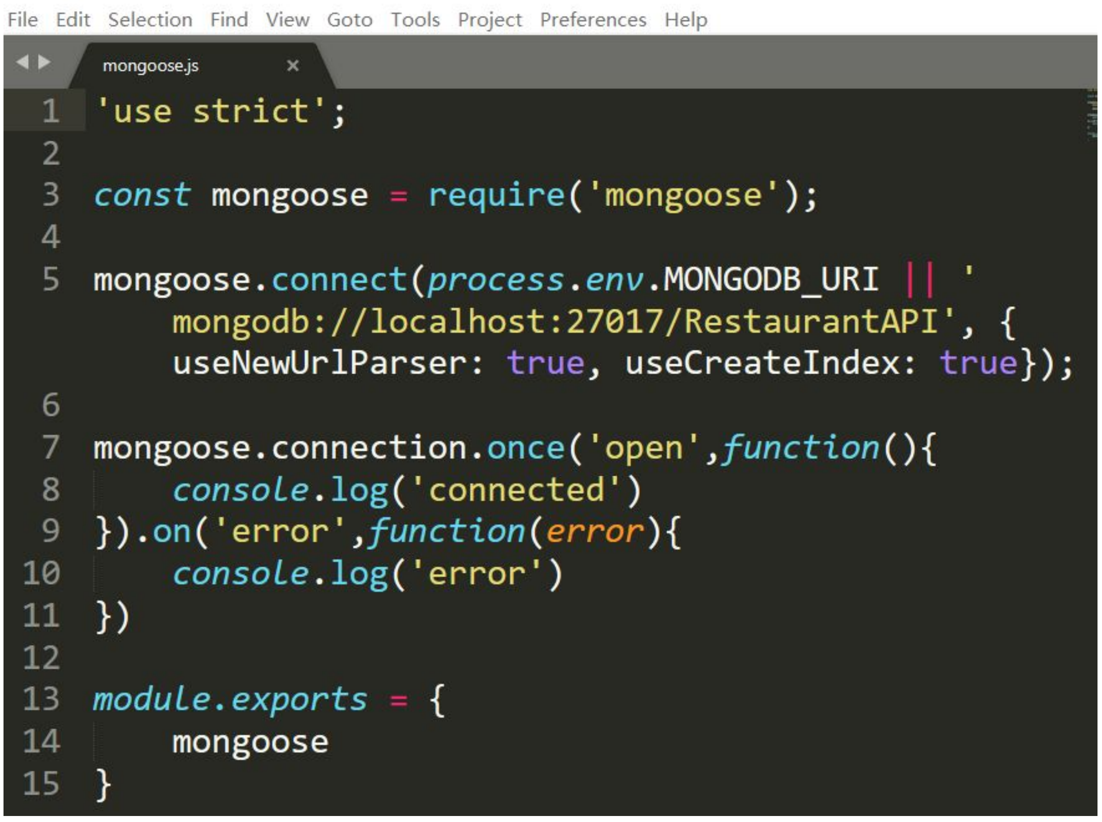


Connect to Host

Hostname: localhost
Port: 27017
SRV Record
Authentication: None
Replica Set Name:
Read Preference: Primary
SSL: None
SSH Tunnel: None
Favorite Name: e.g. Shared Dev, QA Box, PRODUCTION

Note: By default, Hostname is localhost, port is 27017. This can be customized but it has to be modified in the javascript that connects to the server as well.

ii. For this part, we need to write a js script to connect to database.


File Edit Selection Find View Goto Tools Project Preferences Help
mongoose.js

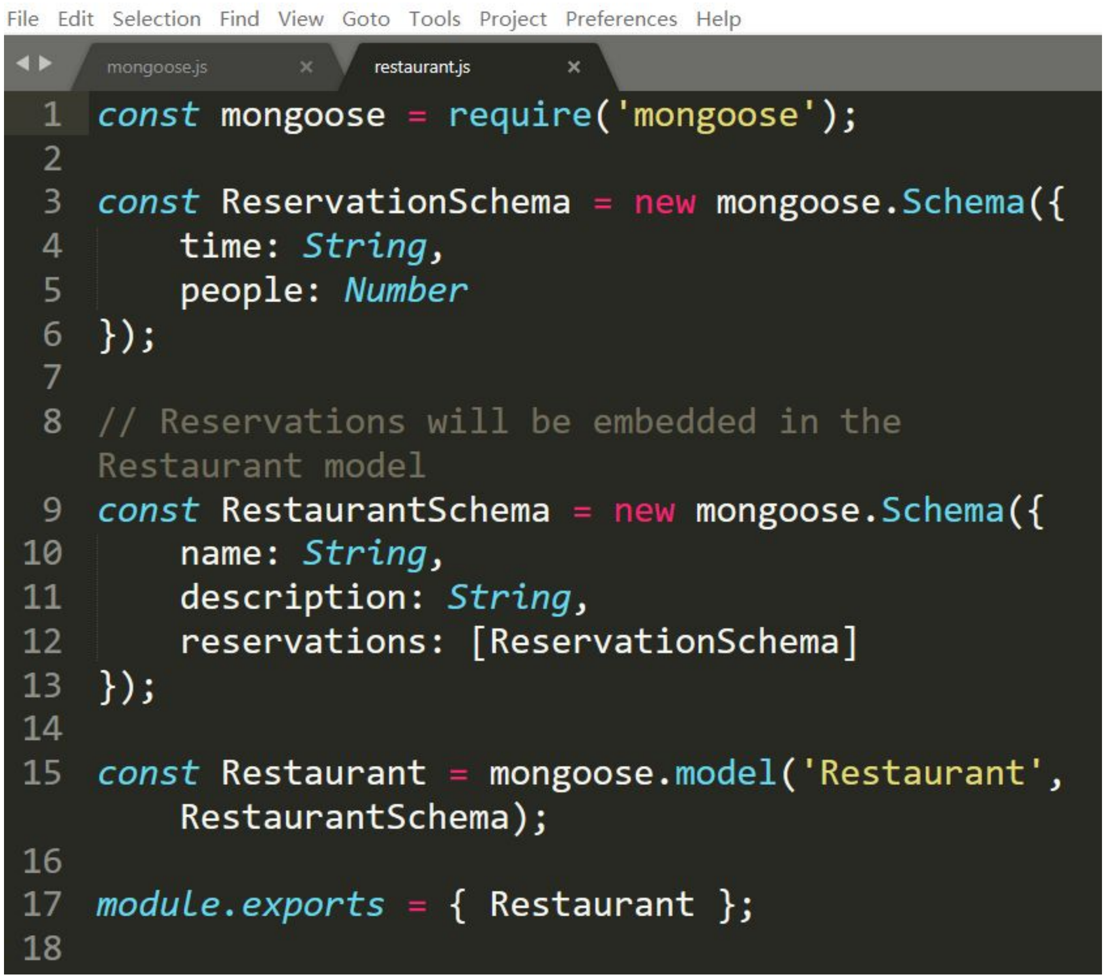
```javascript
'use strict';

const mongoose = require('mongoose');

mongoose.connect(process.env.MONGODB_URI || '
    mongodb://localhost:27017/RestaurantAPI', {
    useNewUrlParser: true, useCreateIndex: true});

mongoose.connection.once('open',function(){
    console.log('connected')
}).on('error',function(error){
    console.log('error')
})

module.exports = {
    mongoose
}
```

In the db folder we have the script mongoose.js, which connects to the database we just set up (localhost:27017) and creates a new schema called RestaurantAPI in the database. Once it is connected, we print “connected” in the cmd panel, or else “error”.

The module is exported with the name mongoose, which can be directly accessed by other js scripts.

2. Set up storage structure

The storage structure defines how the data is stored in the database.


File Edit Selection Find View Goto Tools Project Preferences Help
mongoose.js
restaurant.js

```
const mongoose = require('mongoose');

const ReservationSchema = new mongoose.Schema({
    time: String,
    people: Number
});

// Reservations will be embedded in the Restaurant model
const RestaurantSchema = new mongoose.Schema({
    name: String,
    description: String,
    reservations: [ReservationSchema]
});

const Restaurant = mongoose.model('Restaurant',
    RestaurantSchema);

module.exports = { Restaurant };
```

The structure is:

Restaurant Schema:

_id	name	description	reservation

The description block is an array with ReservationSchema inside:
Reservation Schema:

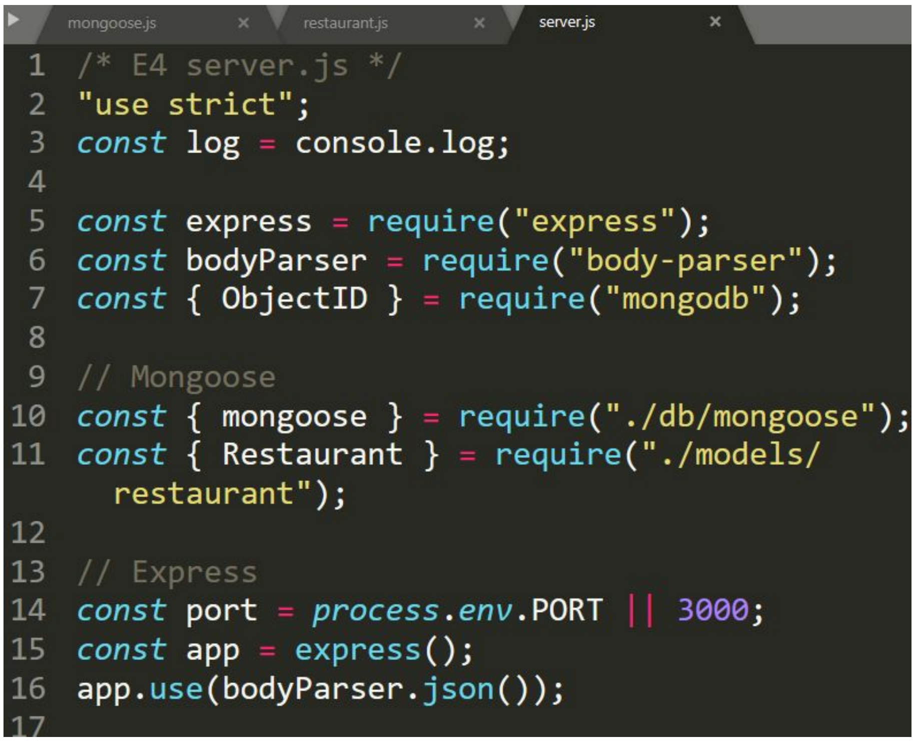
_id	time	people

3. Node.js functions & Testing with client end software PostMan

Please check the complete functions with explanations here:

https://github.com/SauryCC/personalCode/blob/master/Restaurant%20reservation%20MongoDB%20API/e4_starter/server.js

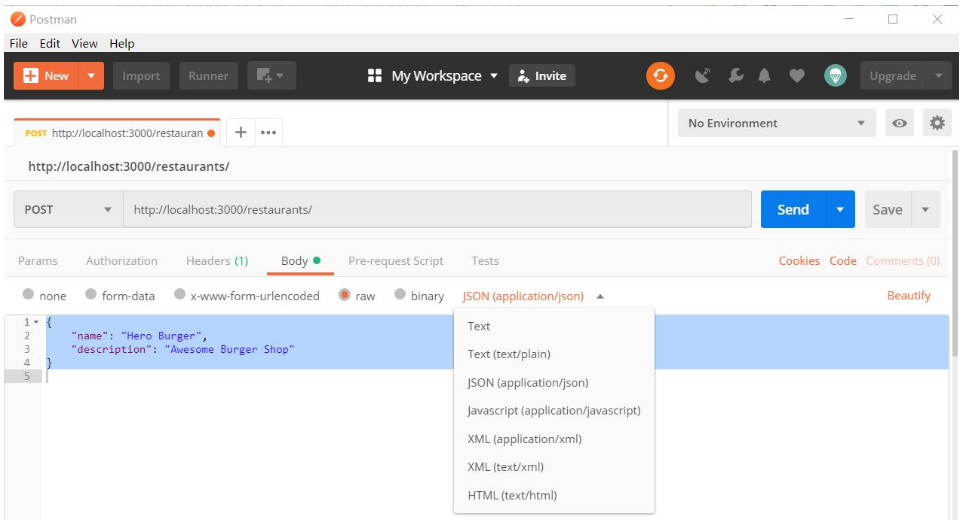
Change directory to current folder and run the file with node.js.



```
/* E4 server.js */
"use strict";
const log = console.log;

const express = require("express");
const bodyParser = require("body-parser");
const { ObjectID } = require("mongodb");

// Mongoose
const { mongoose } = require("./db/mongoose");
const { Restaurant } = require("./models/restaurant");

// Express
const port = process.env.PORT || 3000;
const app = express();
app.use(bodyParser.json());
```

This is a fraction of the script server.js
Variable mongoose extracts the mongoose.js connection file
Variable Restaurant extracts the restaurant.js schema file
Express listens to port 3000 which is where is requests from postMan is sent to.
BodyParser extracts information sent from postMan.

Data structure sent in postMan: A jason file.
Select: Body -> raw -> JSON(application/json)



POST http://localhost:3000/restaurants/

Body: raw, JSON (application/json)

```
{
    "name": "Hero Burger",
    "description": "Awesome Burger Shop"
}
```

Here is a short description of the functions:

a. **app.post("/restaurants", (req, res) => {}**

Post a jason structure received from client (postMan) of:

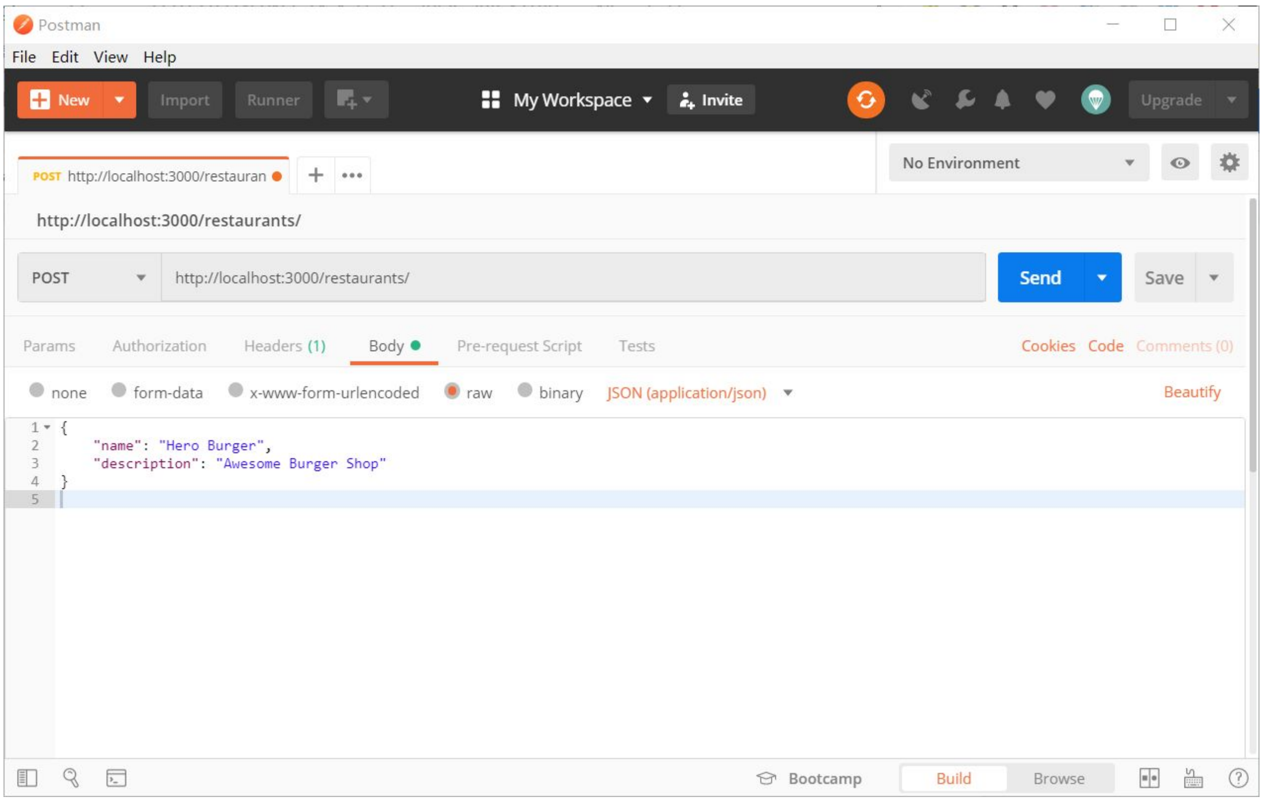
```
{
        "name": <restaurant name>
        "description": <restaurant description>
}
```

And stores in database:

_id: assigned id	name: <restaurant name>	Description: <restaurant description>	Reservation: []

Address in postMan: http://localhost:3000/restaurants/

Input in postMan:
```
{
        "name": "Hero Burger",
        "description": "Awesome Burger Shop"
}
```

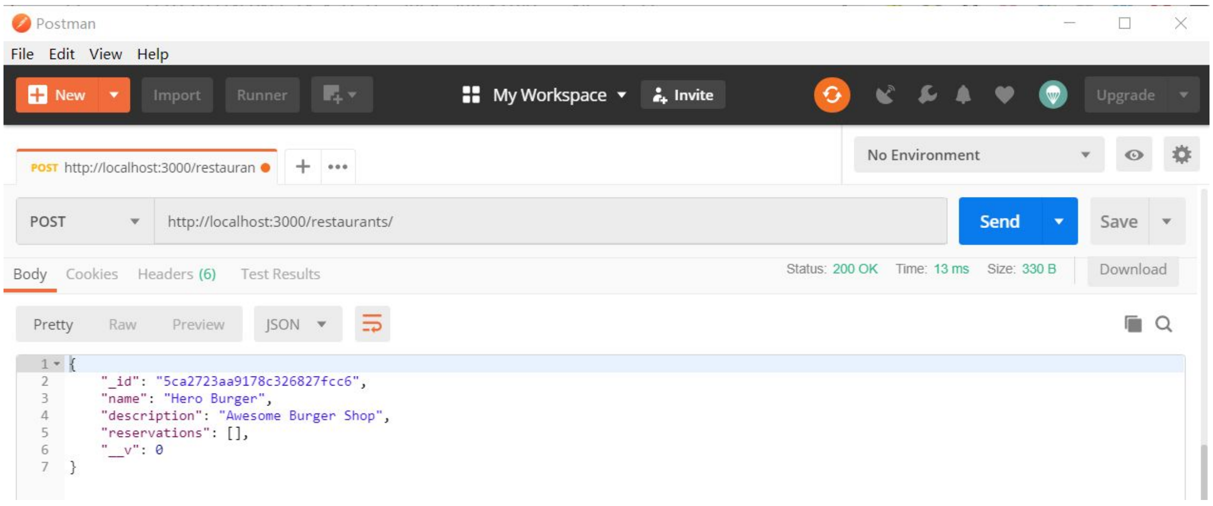


POST http://localhost:3000/restaurants/

Body: raw, JSON (application/json)

```
{
    "name": "Hero Burger",
    "description": "Awesome Burger Shop"
}
```

Reply in postMan:

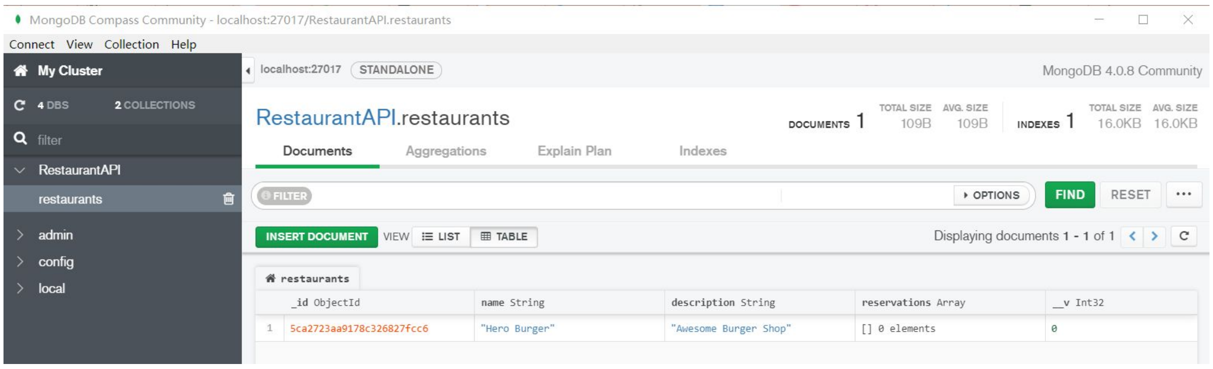


POST http://localhost:3000/restaurants/

Status: 200 OK Time: 13 ms Size: 330 B

```
{
    "_id": "5ca2723aa9178c326827fcc6",
    "name": "Hero Burger",
    "description": "Awesome Burger Shop",
    "reservations": [],
    "__v": 0
}
```

Change in database:


MongoDB Compass Community - localhost:27017/RestaurantAPI.restaurants
Connect View Collection Help
My Cluster
4 DBS 2 COLLECTIONS
filter
RestaurantAPI
restaurants
admin
config
local
localhost:27017 STANDALONE
MongoDB 4.0.8 Community
Documents Aggregations Explain Plan Indexes
FILTER
OPTIONS
FIND
RESET
INSERT DOCUMENT
VIEW LIST TABLE
Displaying documents 1 - 1 of 1

RestaurantAPI.restaurants

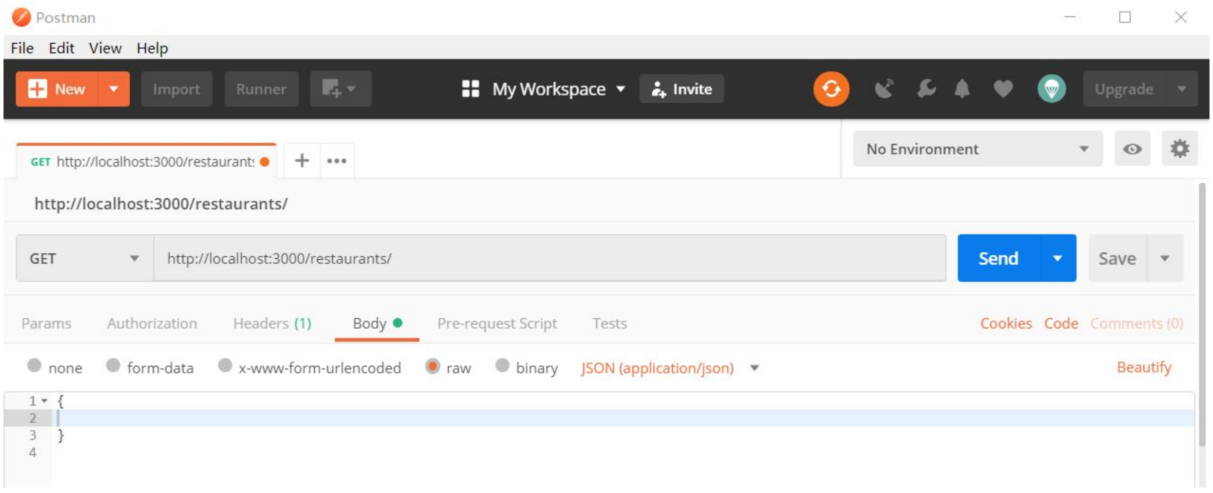
DOCUMENTS 1 TOTAL SIZE 109B AVG. SIZE 109B INDEXES 1 TOTAL SIZE 16.0KB AVG. SIZE 16.0KB

restaurants

	_id ObjectId	name String	description String	reservations Array	__v Int32
1	5ca2723aa9178c326827fcc6	"Hero Burger"	"Awesome Burger Shop"	[] 0 elements	0

b. **app.get("/restaurants", (req, res) => {}**
Get all restaurant information in the "restaurants" table.

Address in postMan: http://localhost:3000/restaurants/
Input in postMan: { }

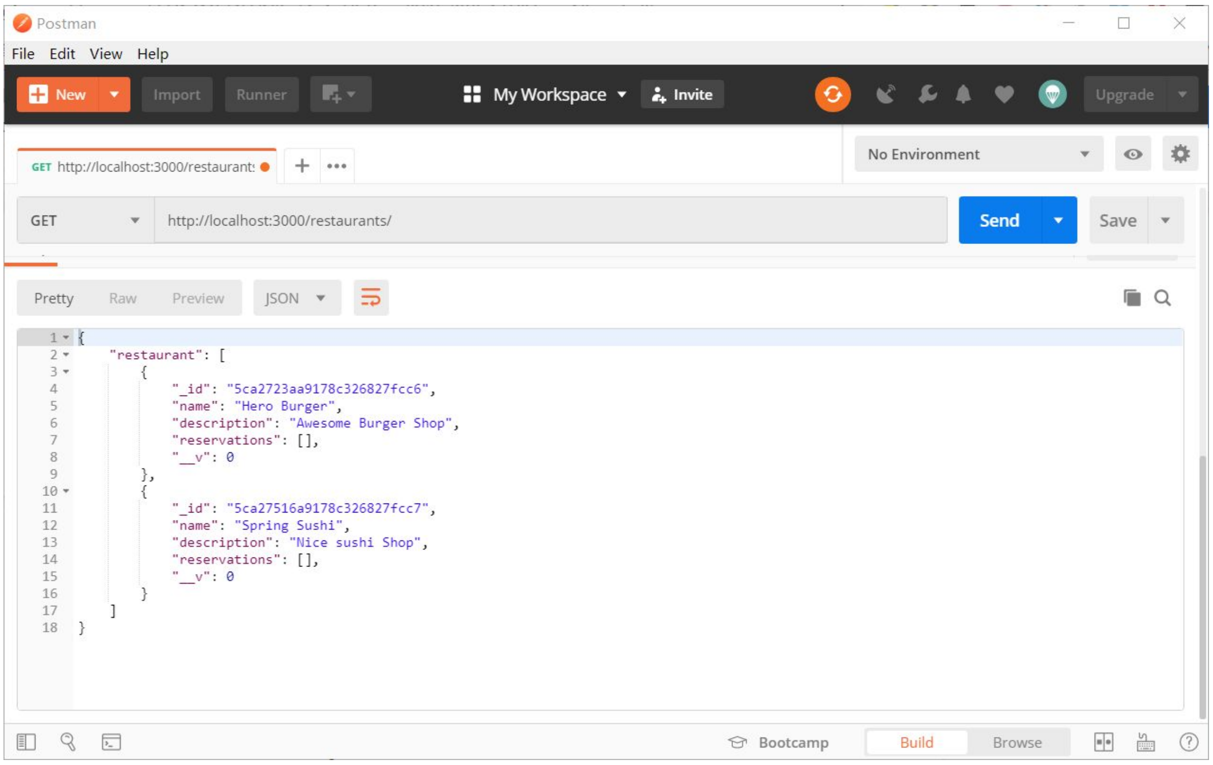

Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
No Environment
GET http://localhost:3000/restaurant
Send
Save
Params Authorization Headers (1) Body Pre-request Script Tests
Cookies Code Comments (0)
none form-data x-www-form-urlencoded raw binary JSON (application/json)
Beautify

http://localhost:3000/restaurants/

GET http://localhost:3000/restaurants/

```
{

}
```

Reply in postMan:



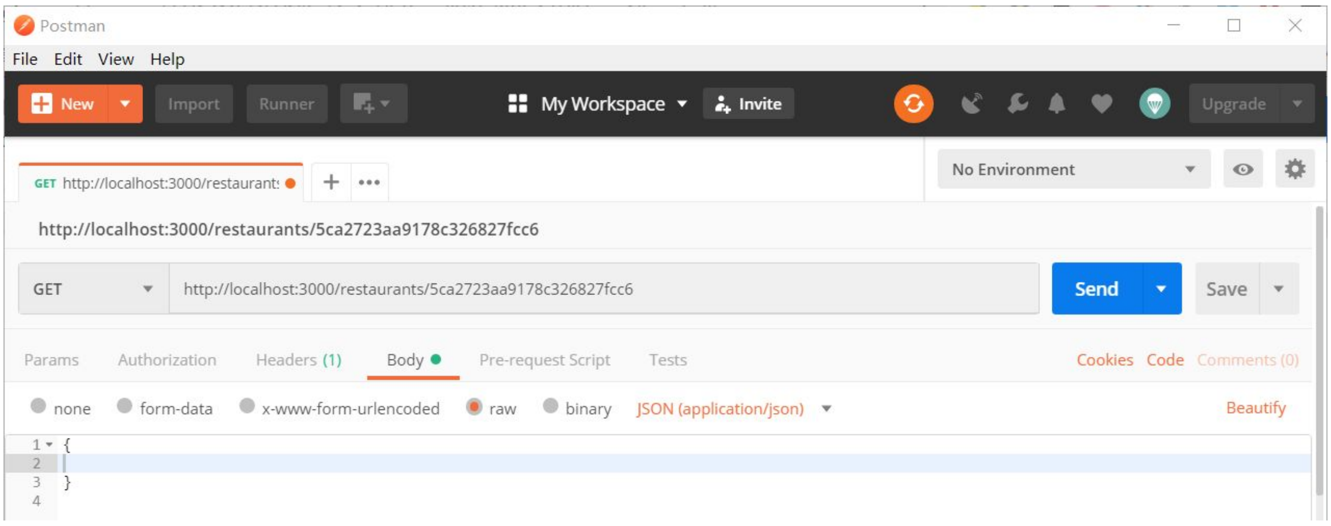
```
{
    "restaurant": [
        {
            "_id": "5ca2723aa9178c326827fcc6",
            "name": "Hero Burger",
            "description": "Awesome Burger Shop",
            "reservations": [],
            "__v": 0
        },
        {
            "_id": "5ca27516a9178c326827fcc7",
            "name": "Spring Sushi",
            "description": "Nice sushi Shop",
            "reservations": [],
            "__v": 0
        }
    ]
}
```

Change in database: none

c. **app.get("/restaurants/:id", (req, res) => {}**
   Get one specific information in the “restaurants” table.

   Address in postMan: http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6
   The address above is in format …./**restaurants/restaurants_id**
   The restaurants_id refers to “**Hero Burger**” we stored above

Input in postMan: { }

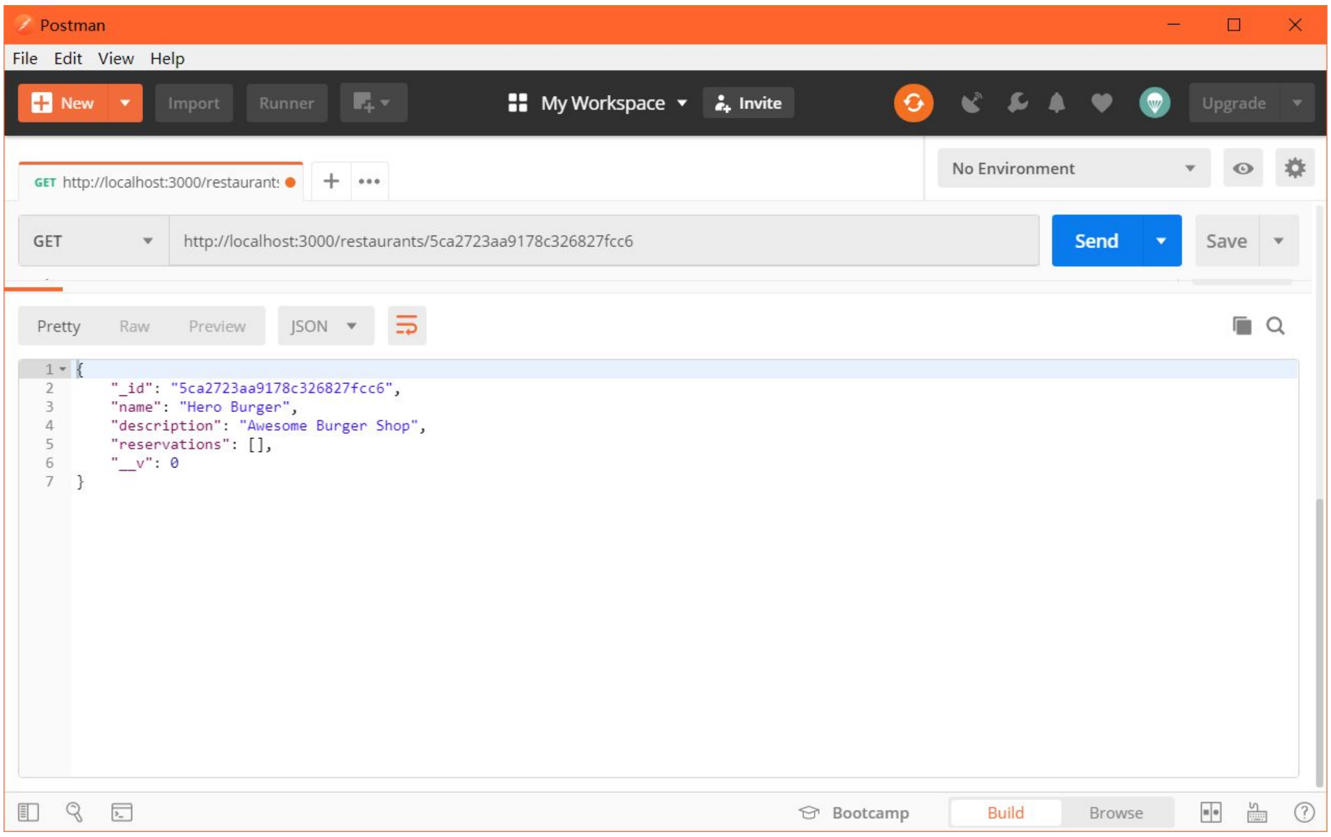


http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6

GET http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6

```
{

}
```

Reply in postMan:



GET http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6

```
{
    "_id": "5ca2723aa9178c326827fcc6",
    "name": "Hero Burger",
    "description": "Awesome Burger Shop",
    "reservations": [],
    "__v": 0
}
```

Change in database: none

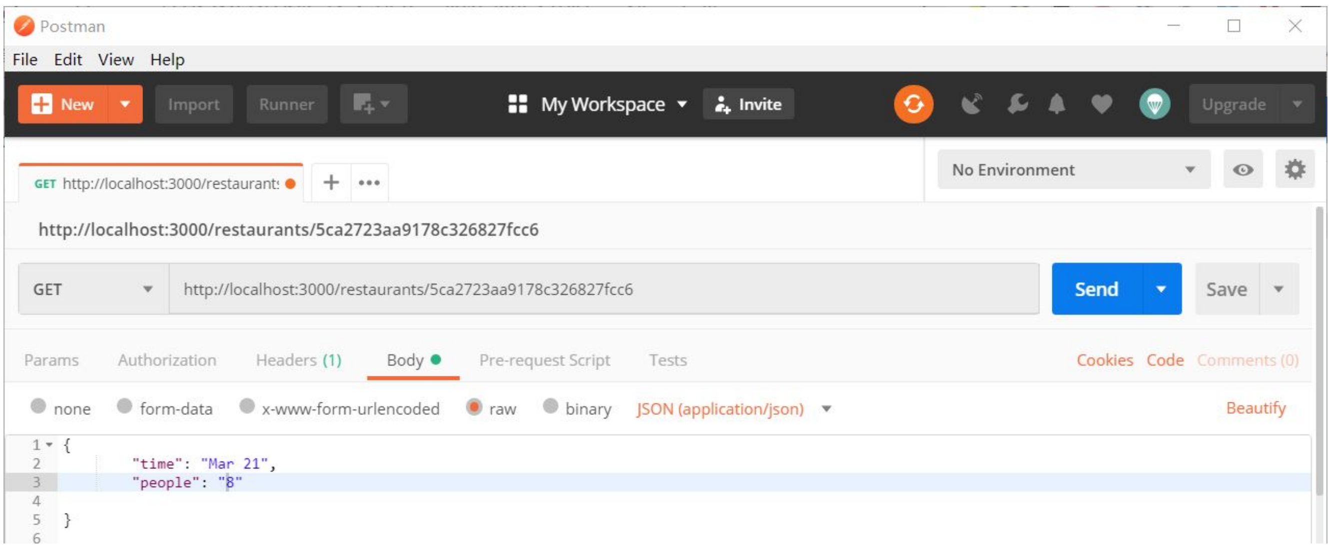
d. **app.post("/restaurants/:id", (req, res) => {}**
Posts a reservation info jason into a restaurant table.
Input format: {
"time": <time>
"people": <number of people>
}

Address in postMan: http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6

The address above is in format …./**restaurants/restaurants_id**
The restaurants_id refers to "**Hero Burger**" we stored above

Input in postMan: {
"time": "Mar 21",
"people": "8"

}

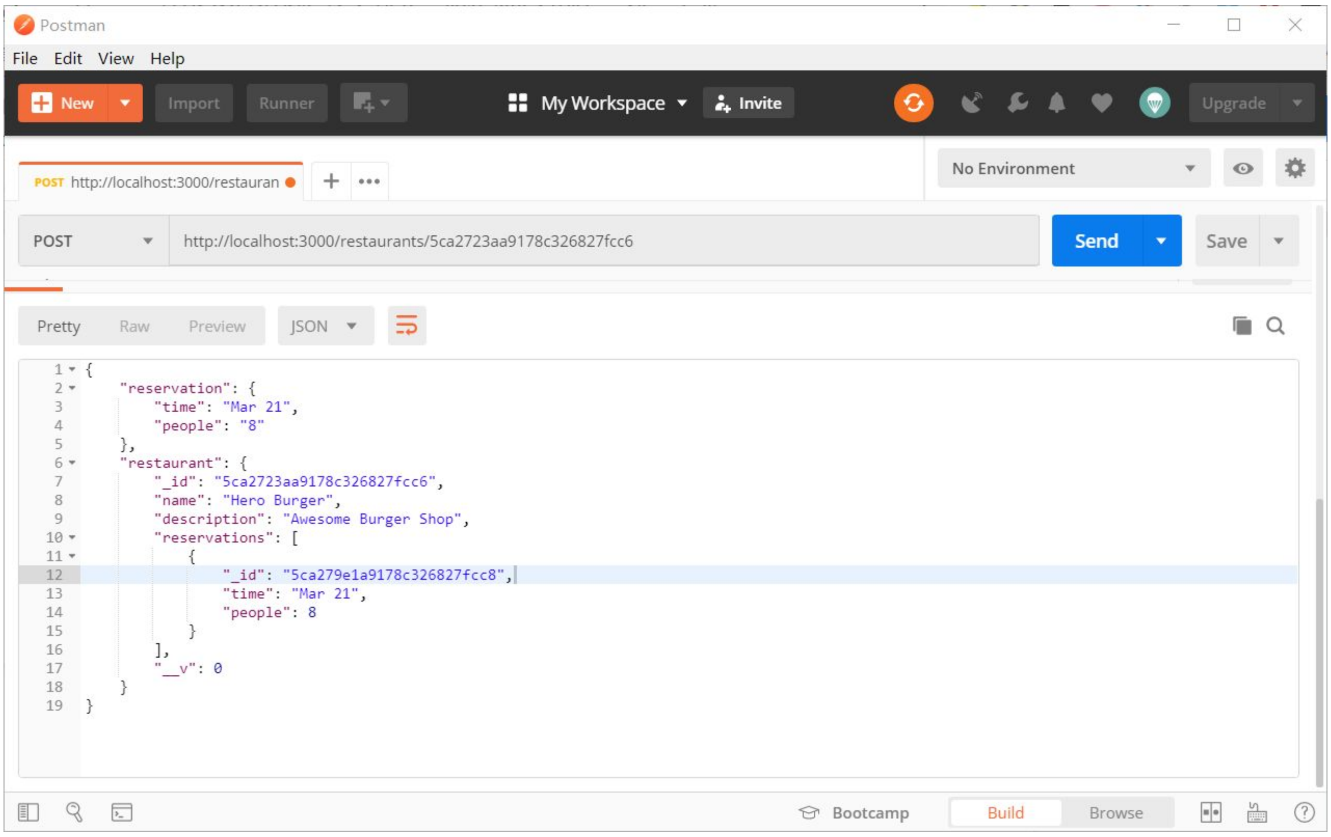


http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6

GET http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6

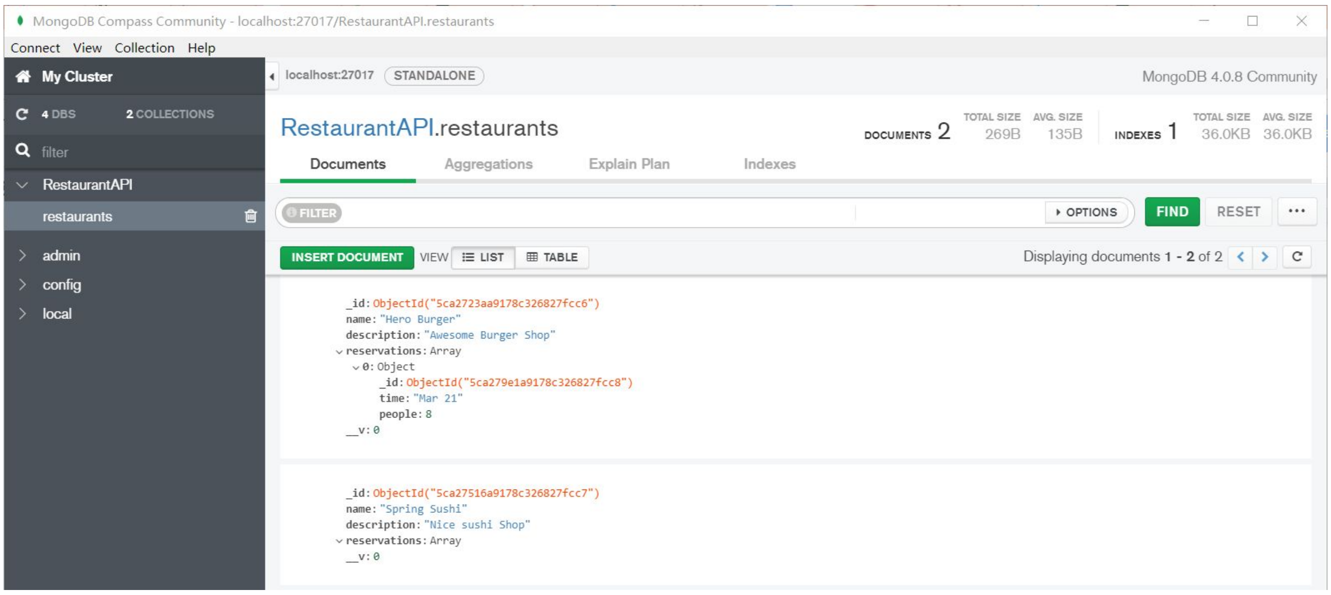
```
{
        "time": "Mar 21",
        "people": "8"

}
```

Reply in postMan:


POST http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6

```
{
    "reservation": {
        "time": "Mar 21",
        "people": "8"
    },
    "restaurant": {
        "_id": "5ca2723aa9178c326827fcc6",
        "name": "Hero Burger",
        "description": "Awesome Burger Shop",
        "reservations": [
            {
                "_id": "5ca279e1a9178c326827fcc8",
                "time": "Mar 21",
                "people": 8
            }
        ],
        "__v": 0
    }
}
```

Change in database:


MongoDB Compass Community - localhost:27017/RestaurantAPI.restaurants

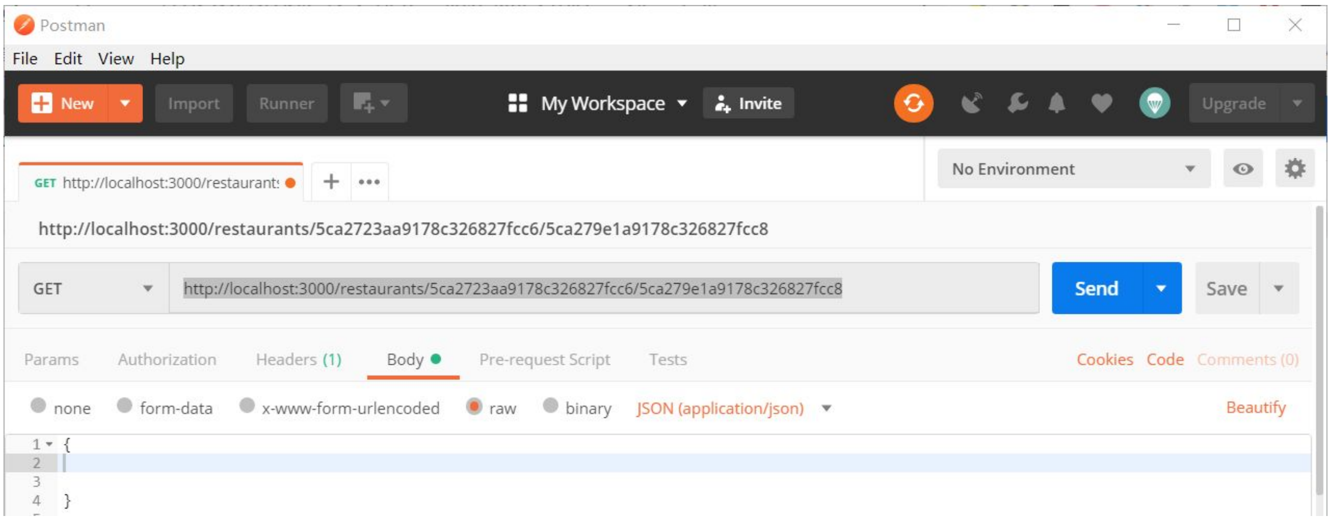
RestaurantAPI.restaurants — DOCUMENTS 2, TOTAL SIZE 269B, AVG. SIZE 135B; INDEXES 1, TOTAL SIZE 36.0KB, AVG. SIZE 36.0KB

Displaying documents 1 - 2 of 2

```
_id: ObjectId("5ca2723aa9178c326827fcc6")
name: "Hero Burger"
description: "Awesome Burger Shop"
reservations: Array
  0: Object
    _id: ObjectId("5ca279e1a9178c326827fcc8")
    time: "Mar 21"
    people: 8
__v: 0

_id: ObjectId("5ca27516a9178c326827fcc7")
name: "Spring Sushi"
description: "Nice sushi Shop"
reservations: Array
__v: 0
```

e. **app.get("/restaurants/:id/:resv_id", (req, res) => {}**
   Address in postMan:
   http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6/5ca279e1a9178c326827fcc8
   The address is in format ….**/restaurants/restaurants_id/reservation_id**
   The restaurants_id refers to "**Hero Burger**" we stored above
   The **reservation_id** refers to "**reservation 1**" we stored above

Input in postMan: {   }

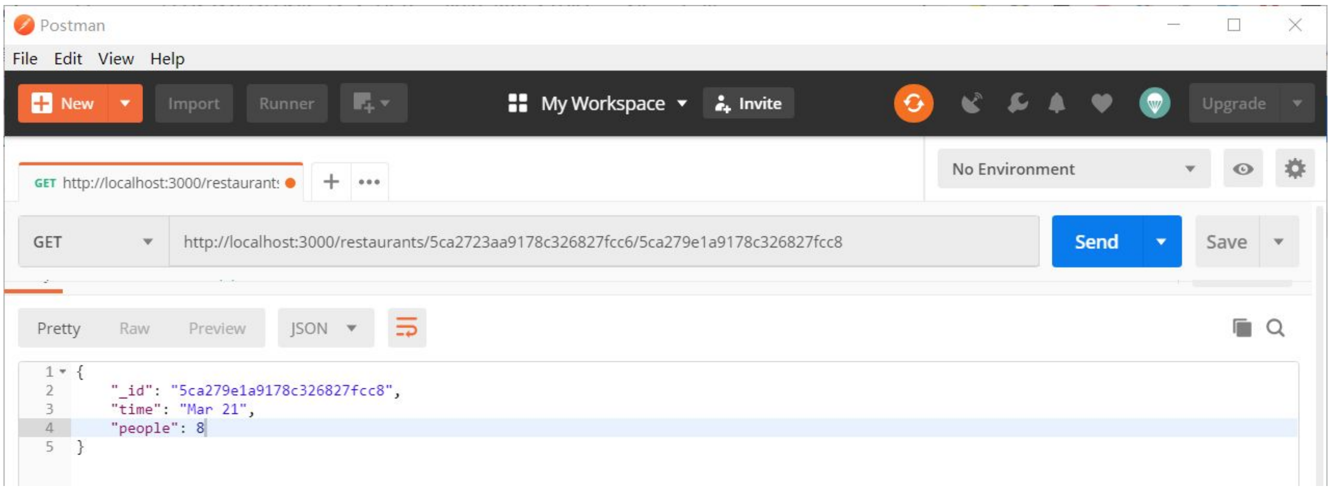

Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
No Environment
Send
Save
Params Authorization Headers (1) Body Pre-request Script Tests
Cookies Code Comments (0)
none form-data x-www-form-urlencoded raw binary
Beautify

GET http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6/5ca279e1a9178c326827fcc8

Body: raw, JSON (application/json)

```
{

}
```

Reply in postMan:



GET http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6/5ca279e1a9178c326827fcc8

```
{
    "_id": "5ca279e1a9178c326827fcc8",
    "time": "Mar 21",
    "people": 8
}
```

Change in database: none

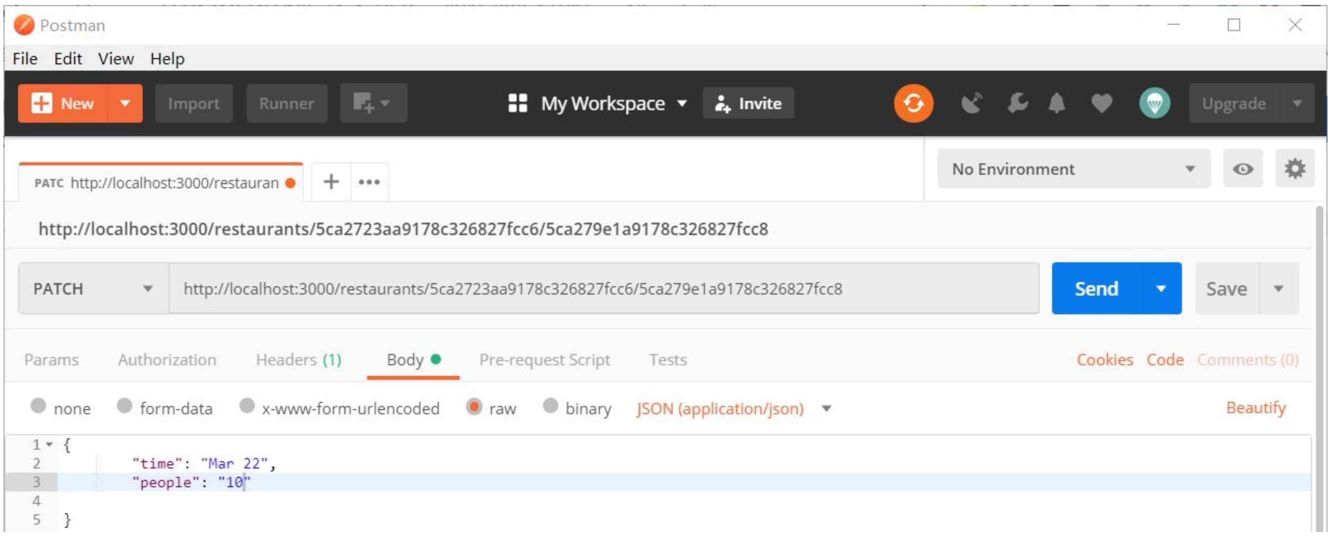
f. **app.patch("/restaurants/:id/:resv_id", (req, res) => {}**
This function changes the reservation info stored in database.

Address in postMan:
http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6/5ca279e1a9178c326827fcc8
The address is in format …./**restaurants/restaurants_id/reservation_id**
The restaurants_id refers to "**Hero Burger**" we stored above
The **reservation_id** refers to "**reservation 1**" we stored above
With info:

```
{
        "_id": "5ca279e1a9178c326827fcc8",
         "time": "Mar 21",
         "people": 8
}
```

Now let's change the reservation to Mar 22 with 10 people
Input in postMan:

```
{
        "time": "Mar 22",
        "people": "10"

}
```


Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
No Environment
Send
Save
Params Authorization Headers (1) Body Pre-request Script Tests
Cookies Code Comments (0)
none form-data x-www-form-urlencoded raw binary JSON (application/json)
Beautify

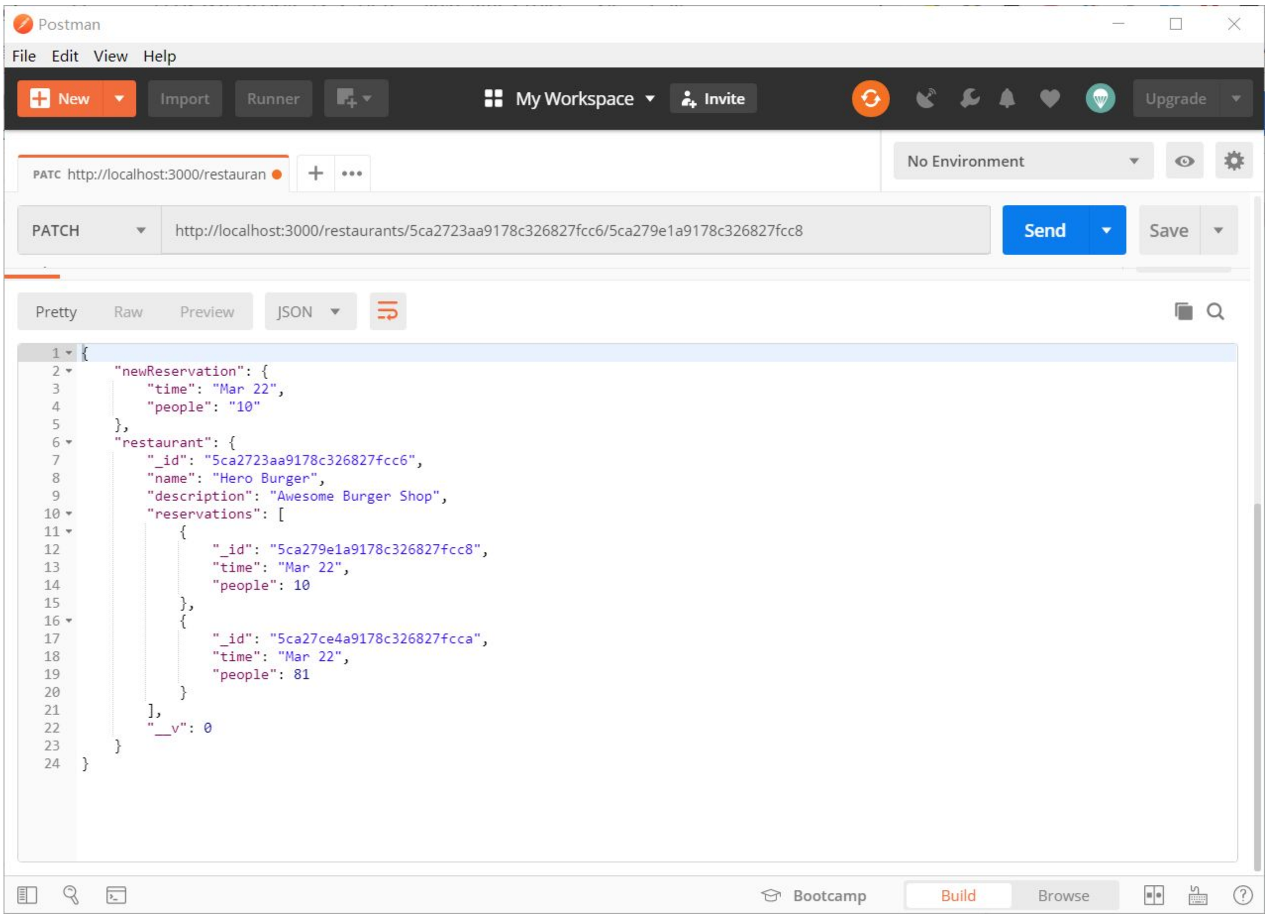
PATC http://localhost:3000/restauran

http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6/5ca279e1a9178c326827fcc8

PATCH http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6/5ca279e1a9178c326827fcc8

```
{
      "time": "Mar 22",
      "people": "10"

}
```

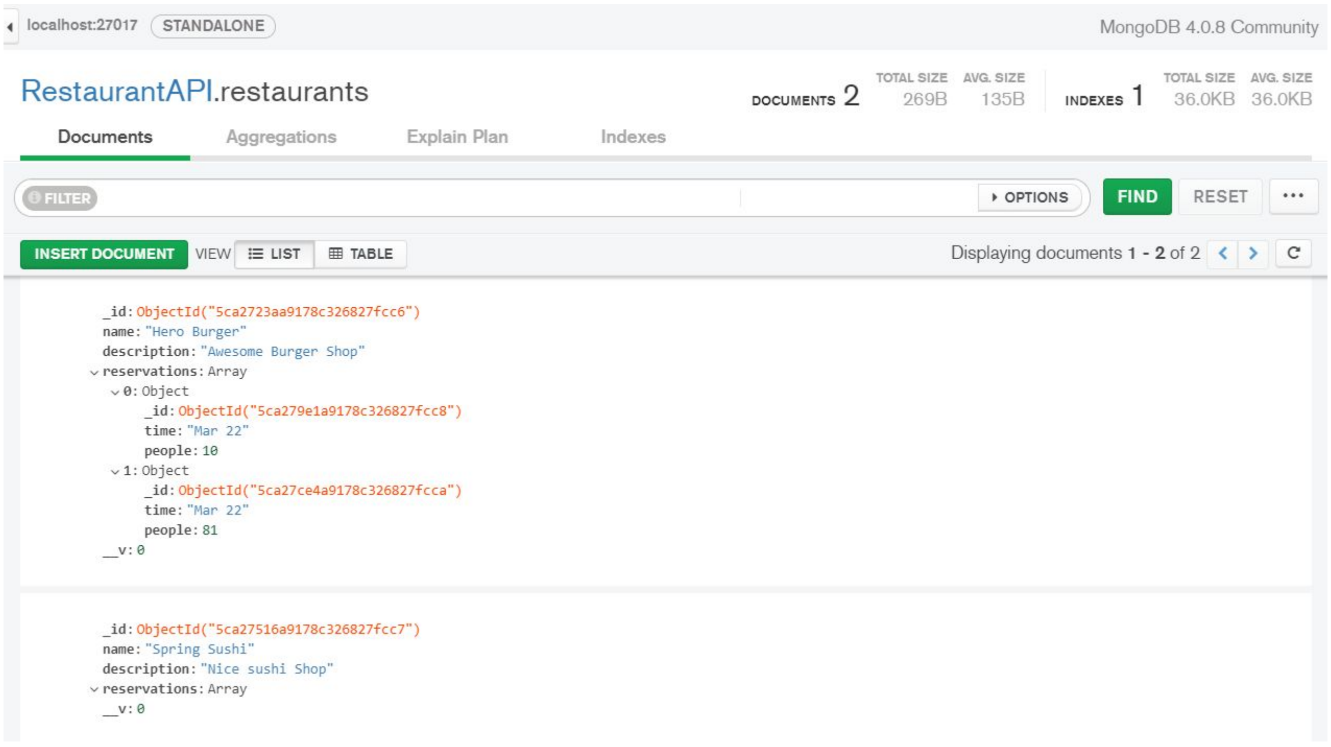
Reply in postMan:



PATCH http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6/5ca279e1a9178c326827fcc8

```
{
    "newReservation": {
        "time": "Mar 22",
        "people": "10"
    },
    "restaurant": {
        "_id": "5ca2723aa9178c326827fcc6",
        "name": "Hero Burger",
        "description": "Awesome Burger Shop",
        "reservations": [
            {
                "_id": "5ca279e1a9178c326827fcc8",
                "time": "Mar 22",
                "people": 10
            },
            {
                "_id": "5ca27ce4a9178c326827fcca",
                "time": "Mar 22",
                "people": 81
            }
        ],
        "__v": 0
    }
}
```

Change in database:


localhost:27017 STANDALONE
MongoDB 4.0.8 Community
Documents Aggregations Explain Plan Indexes
FILTER
OPTIONS
FIND RESET
INSERT DOCUMENT VIEW LIST TABLE

RestaurantAPI.restaurants

DOCUMENTS 2 TOTAL SIZE 269B AVG. SIZE 135B INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB

Displaying documents 1 - 2 of 2

```
_id: ObjectId("5ca2723aa9178c326827fcc6")
name: "Hero Burger"
description: "Awesome Burger Shop"
reservations: Array
  0: Object
     _id: ObjectId("5ca279e1a9178c326827fcc8")
     time: "Mar 22"
     people: 10
  1: Object
     _id: ObjectId("5ca27ce4a9178c326827fcca")
     time: "Mar 22"
     people: 81
__v: 0
```

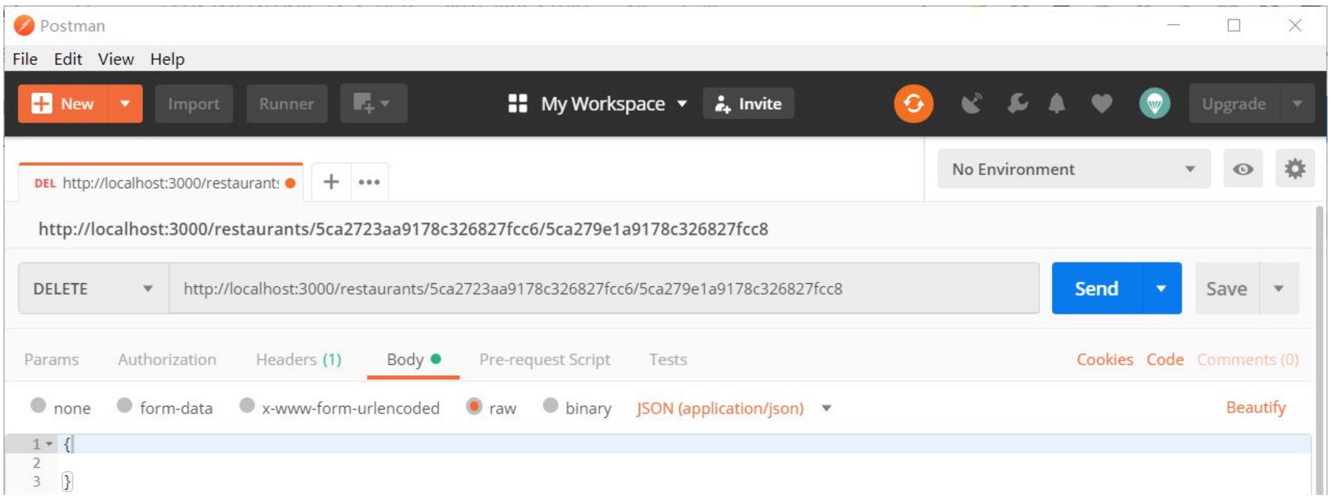
```
_id: ObjectId("5ca27516a9178c326827fcc7")
name: "Spring Sushi"
description: "Nice sushi Shop"
reservations: Array
__v: 0
```

g. **app.delete("/restaurants/:id/:resv_id", (req, res) => {}**
This function deletes selected reservation info stored in database.

Address in postMan:
http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6/5ca279e1a9178c326827fcc8
The address is in format …./**restaurants/restaurants_id/reservation_id**
The restaurants_id refers to "**Hero Burger**" we stored above
The **reservation_id** refers to "**reservation 1**" we stored above
With info modified in previous step:

```
{
        "_id": "5ca279e1a9178c326827fcc8",
        "time": "Mar 22",
        "people": "10"
}
```

Input in postMan: { }



Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
No Environment
DEL http://localhost:3000/restaurant:
Send
Save
Params Authorization Headers (1) Body Pre-request Script Tests
Cookies Code Comments (0)
none form-data x-www-form-urlencoded raw binary
Beautify

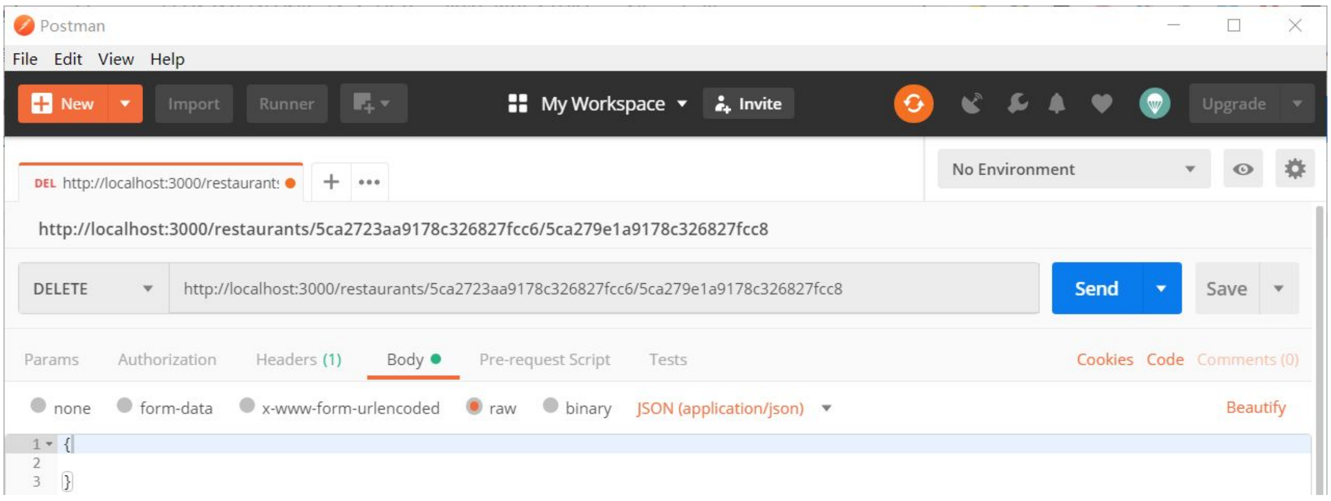
http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6/5ca279e1a9178c326827fcc8

DELETE http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6/5ca279e1a9178c326827fcc8

Body: raw, JSON (application/json)

```
{

}
```

Reply in postMan:



Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
No Environment
DEL http://localhost:3000/restaurant:
Send
Save
Params Authorization Headers (1) Body Pre-request Script Tests
Cookies Code Comments (0)
none form-data x-www-form-urlencoded raw binary
Beautify

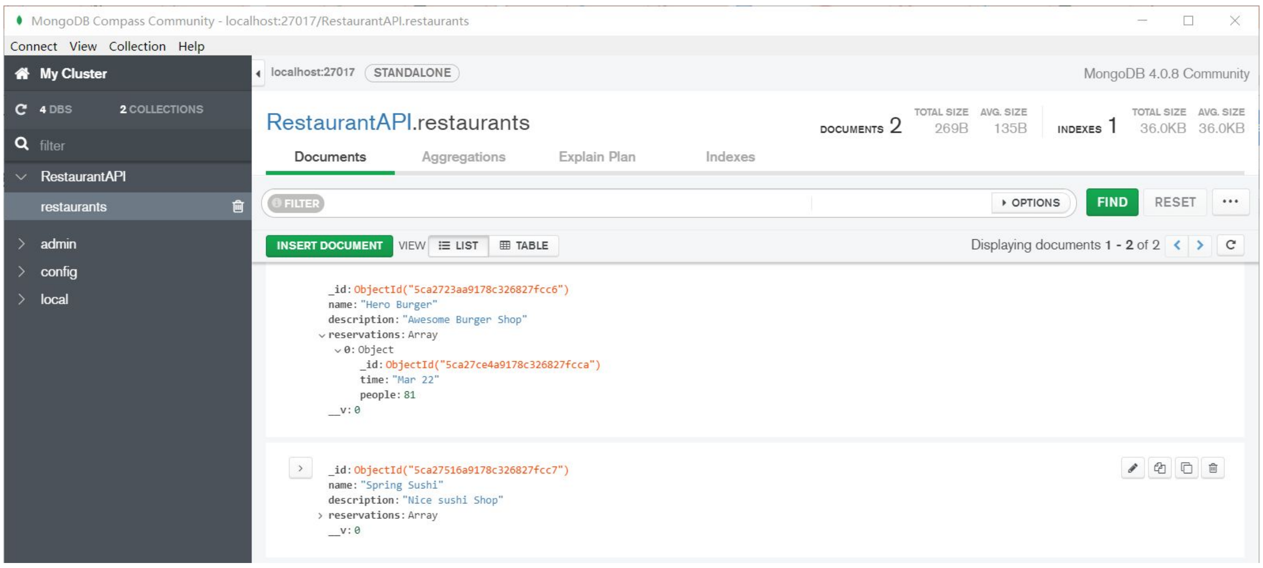
http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6/5ca279e1a9178c326827fcc8

DELETE http://localhost:3000/restaurants/5ca2723aa9178c326827fcc6/5ca279e1a9178c326827fcc8

Body: raw, JSON (application/json)

```
{

}
```

Change in database:


MongoDB Compass Community - localhost:27017/RestaurantAPI.restaurants
Connect View Collection Help
My Cluster
4 DBS 2 COLLECTIONS
filter
RestaurantAPI
restaurants
admin
config
local
localhost:27017 STANDALONE
MongoDB 4.0.8 Community
RestaurantAPI.restaurants
DOCUMENTS 2 TOTAL SIZE 269B AVG. SIZE 135B
INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB
Documents Aggregations Explain Plan Indexes
FILTER
OPTIONS
FIND
RESET
INSERT DOCUMENT VIEW LIST TABLE
Displaying documents 1 - 2 of 2

```
_id: ObjectId("5ca2723aa9178c326827fcc6")
name: "Hero Burger"
description: "Awesome Burger Shop"
reservations: Array
  0: Object
    _id: ObjectId("5ca27ce4a9178c326827fcca")
    time: "Mar 22"
    people: 81
__v: 0
```

```
_id: ObjectId("5ca27516a9178c326827fcc7")
name: "Spring Sushi"
description: "Nice sushi Shop"
reservations: Array
__v: 0
```