READ

The next routes we write will handle fetching resources from our database. In particular we will want to write two routes, one to handle fetching all of our cities and another to fetch a single city. Understanding fetching specific documents with dynamic route parameters will be essential when we get to updating and deleting documents as well.

Part 1: Fetching Multiple Documents

Step 1: Begin writing our route method.

```
router.get('/', (req, res) => {
```

```
// Similarly to our create route we pass in the two required arguments but this time we are making a “get” request. Since our HTTP methods are different we can use the same endpoint “localhost:5000/cities/“.
```

Step 2: Fetching all cities in our collection.

```
City.find()
```

```
// We use the mongoose .find( ) method on our City Model and pass in no parameters. This will return all the documents in our cities collection in our database in JSON format.
```

Step 3: Sorting our results.

```
    .sort({ name: 1 })
```

```
//Optionally we can attach a .sort( ) method on to the end and pass in an object specifying the order in which we would our documents returned. Here I chose name and set its value to ‘1’ so that the cities are return alphabetically.
```

Step 4: Returning our array in the response.

```
    .then(cities => {
      res.send(cities)
    })
    .catch(err => res.send(err.message))
})
```

```
//The .find() method will return a promise, if fulfilled we will get an array of our cities. If rejected we can debug depending on our error message.
```

Part 2: Testing on Postman

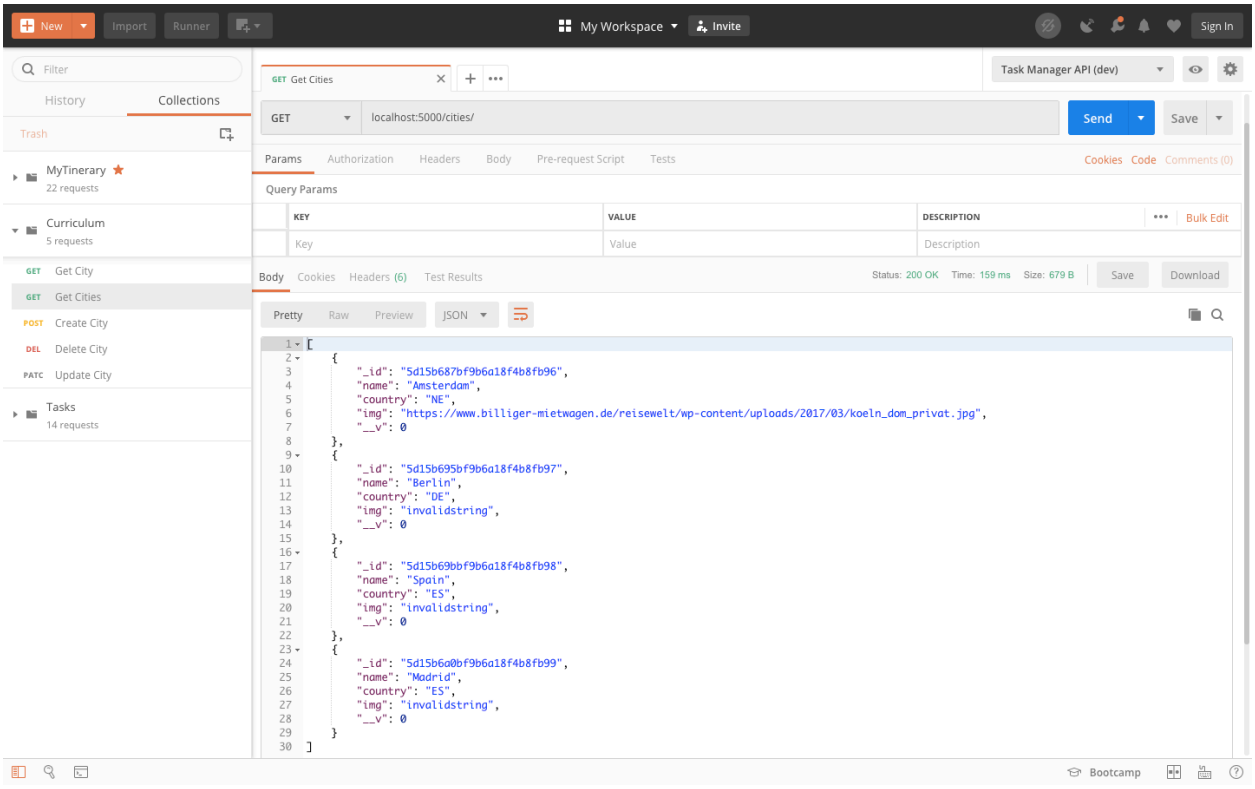
Step 1: Set the HTTP method to a “GET” request and set the end point to “localhost: 5000/api/cities”

Step 2: Since we aren’t posting any data we simply need to fire off our request and await our response. Depending on how many cities are in your database your response should look similar to figure 1.

Step 3: If you receive an error message as a response, check to make sure your server is running correctly, if so it is most likely an issue with your route method.

Step 4: Upon a successful response copy an object _id from one city in the array that was returned to your clipboard as we will be using it in the next exercise.

Figure 1:



Part 3: Fetching Single Documents

Step 1: Begin writing our route method.

```
router.get('/:id', (req, res) => {
```

```
// Here instead of simply '/' we have this syntax with '/:id'. The colon makes the parameter of the endpoint dynamic. Each document in our collection will have its own endpoint corresponding to the document id which we will pass in with our request.
```

Step 2: Fetching City document by id.

```
  City.findById(req.params.id)
```

```
// We use the mongoose .findByID( ) method on our City Model and pass in 'req params id' as our only argument. This is referring to the parameter in our route.
```

Step 3: Checking with conditional logic.

```
    .then(city => {
      if (!city) {
        return res.status(400).send({ msg: 'City not found' })
      }
```

```
// Even if the document is not found, it does not necessarily mean that our route handler did not do its job correctly. Incase the document does not exist in our database, we use some conditional logic here to return out of the function and send back a custom message.
```

Step 3: Receiving our document.

```
      res.send(city)
    })
    .catch(err => {
      res.send(err.message)
    })
})
```

```
// If all goes well we will receive a single object which will represent our document.
```

Part 4: Testing on Postman

Step 1: Set the HTTP method to a “GET” request and set the end point to “localhost: 5000/api/cities/‘

Step 2: Paste the object id from your clipboard which corresponds with a city in your database to the end of your route.

Step 3: Send the request and await the response, if successful you should receive a single object. (See figure 2 for reference)

Step 4: Now replace a character in the object id route parameter with an incorrect one to make sure you receive the custom error message from Step 3.

Step 5: If you receive your error message. You are all done replace with the correct character and save this route to your Postman collection under ‘Get City’

Figure 2: