Basic CRUD: Blog App

- In an empty directory run $ monk project Blog
- In the root of the folder, you will see package.json and tsconfig.json files being created. Also an “src” folder with the project main directory i.e “Blog” being created.
- Open package.json file and edit the name and also the start script:

```
"name": "blog",
"scripts": {
   "start": "nodemon './src/Blog/Server.ts'",
   "build": "tsc -p ."
 },
```

- Run $ npm install
- Now, let’s create a post app. Run: $ monk app post
- You will see a new app being created inside the src folder with the name of Post. The folder contains subdirectories viz. Controllers, Models, Routes, and Services. Each folder has @_filename to export from a single point.
  In the routes folder, there’s an additional file called PostRoutes. This file contains the router for the post-app. Ps. the file also shows how to create routes for the app.

- Now let’s create a basic controller for the post-app.
- Go inside the Post/Controllers and create a new file “PostControllers.ts”
- Inside the file:

```
import {Request, Response} from 'express'
import {ApiListController} from 'bluemonk'



class PostList extends ApiListController{
    // here you will write the logic to list post
    // but for now we will just send a dummy data
    get(req:Request, res:Response){
    const posts = {_id: '123', title: 'a dummy post'}
    return res.status(200).send({data: posts})
    }
}
export {
    PostList
}

```

- ApiListController is among various other built-in controllers in Bluemonk. We will go in-depth into using all of the controllers later on.
- Now import the PostList controller in the Post/Routes/PostRoutes.ts

```
import express from 'express'
import {run} from 'bluemonk'

import {PostList} from '../Controllers/PostControllers'

const app_router = express.Router()
app_router.get('/list', (req, res)=> run(PostList, req, res))

export default app_router
```

- You can simply create the post router as shown above.
- Now in the main directory routes file i.e Blog/Routes.ts. You can create the main route to point to the app router.

```
import express from 'express'
import post_router from '../Post/Routes/PostRoutes'
const router_main = express.Router();



//All routes
router_main.use('/post', post_router)

export default router_main;

```

- Now you can run $ npm start to start the server. Open Postman
- Run get method in the postman: http://localhost:5000/post/list. You should be able to get back the dummy data.

Mongo Atlas:

- Before we move forward you need to create a MongoDB Atlas account. Go to https://cloud.mongodb.com/ to create a new mongo atlas account or login into your account. We will not discuss how to create a MongoDB Atlas account in this tutorial.
- In the root of the project ie. where the package.json file is located. Create a new file simply ".env".
- Inside the .env file:

DB_CONNECT = mongodb+srv://<username>:<password>@cluster0.zdr8r.mongodb.net/<dbname>?retryWrites=true&w=majority

Create the DB_CONNECT = provide your atlas credentials.

Post Model

- Lets' create a post model. Go to Post/Models and create a new file PostModels.ts
  Add the following to the file:

```
import {model, Schema} from 'mongoose'

const postSchema = new Schema({
    title: {
      type: String,
      required: true,
    }
  });

export default model('Post', postSchema)
```

- Now let's modify the code in Post/Controllers/PostControllers.ts to create a CRUD that creates, retrieves, updates, and deletes from a real database.

Post Controllers

- In Post/Controllers/PostControllers.ts:

```
import {Request, Response} from 'express'
import Post from '../Models/PostModel'
import {ApiListController, ApiCreateController, ApiUpdateController,
ApiRetrieveController, ApiDestroyController} from 'bluemonk'

class PostCreate extends ApiCreateController{

   async post(req:Request, res:Response){
        const post_title = req.body.title
        const post = new Post({
            title: post_title,
          });

        const savedPost = await post.save();
        return res.status(200).send({data: savedPost})
```

```
    }
}

class PostList extends ApiListController{
    async get(req:Request, res:Response){
        const posts = await Post.find();
        return res.status(200).send({data: posts})
    }
}


class PostUpdate extends ApiUpdateController{
    async put(req:Request, res:Response){
        const post_id = req.params.id
        const post_ins = req.body;
        const updatedPost = await Post.updateOne(
            { _id: post_id },
            { ...post_ins }
        );
        return res.status(200).send({data: updatedPost})
    }
}

class PostRetrieve extends ApiRetrieveController{
    async get(req:Request, res:Response){
        const post_id = req.params.id
        const post = await Post.findById(post_id)
        return res.status(200).send({data: post})
    }
}

class PostDelete extends ApiDestroyController{
    async delete(req:Request, res:Response){
        const post_id = req.params.id
        const post = await Post.deleteOne({ _id: post_id });
        res.status(200).send({data: `Post ${post_id} deleted`})
    }
}
```

```
export {
    PostCreate,
    PostList,
    PostUpdate,
    PostDelete,
    PostRetrieve
}
```

- The methods needn't be asynchronous but it fits our need right now.

Post Routes

- And in the Post/Routes/PostRoutes.ts
  Add the following:

```
import express from 'express'
import {run} from 'bluemonk'
import {PostCreate, PostList, PostUpdate, PostRetrieve, PostDelete}
from '../Controllers/PostControllers'

const app_router = express.Router()
app_router.get('/list', (req, res)=> run(PostList, req, res))
app_router.post('/create', (req, res)=> run(PostCreate, req, res))
app_router.get('/:id', (req, res)=> run(PostRetrieve, req, res))
app_router.put('/:id', (req, res)=> run(PostUpdate, req, res))
app_router.delete('/:id', (req, res)=> run(PostDelete, req, res))


export default app_router
```

- Run $ npm start

- CRUD test in postman

PostMan

- To create a new post: (make sure to choose JSON(application/json))

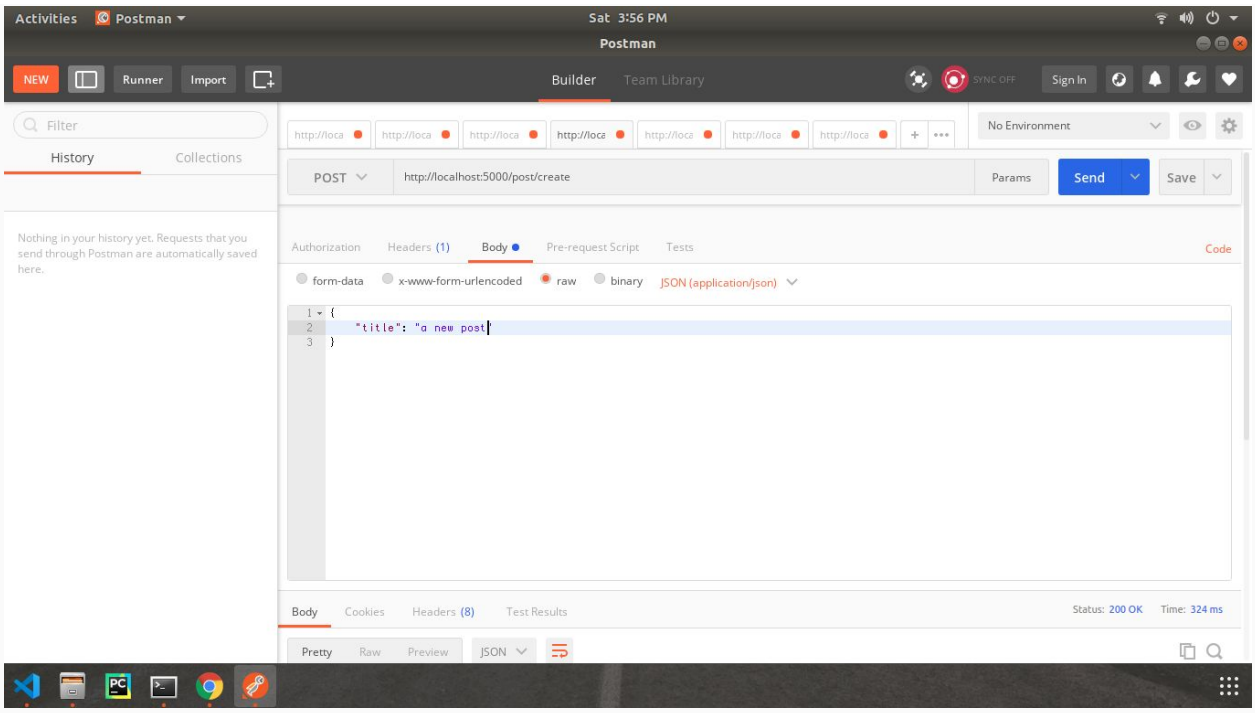
  In http://localhost:5000/post/create make a POST method request with the body

  {

  “title”: “ a new post”

  }

-



If you scroll down you will see the response object

Also, you can see the post in the Atlas collection post.

- To list all the post :

  In http://localhost:5000/post/list make a GET method request

  You may get back an object :

  {

  "data": [

  {

  "_id": "5fc20db3e8e81619d9130d29",

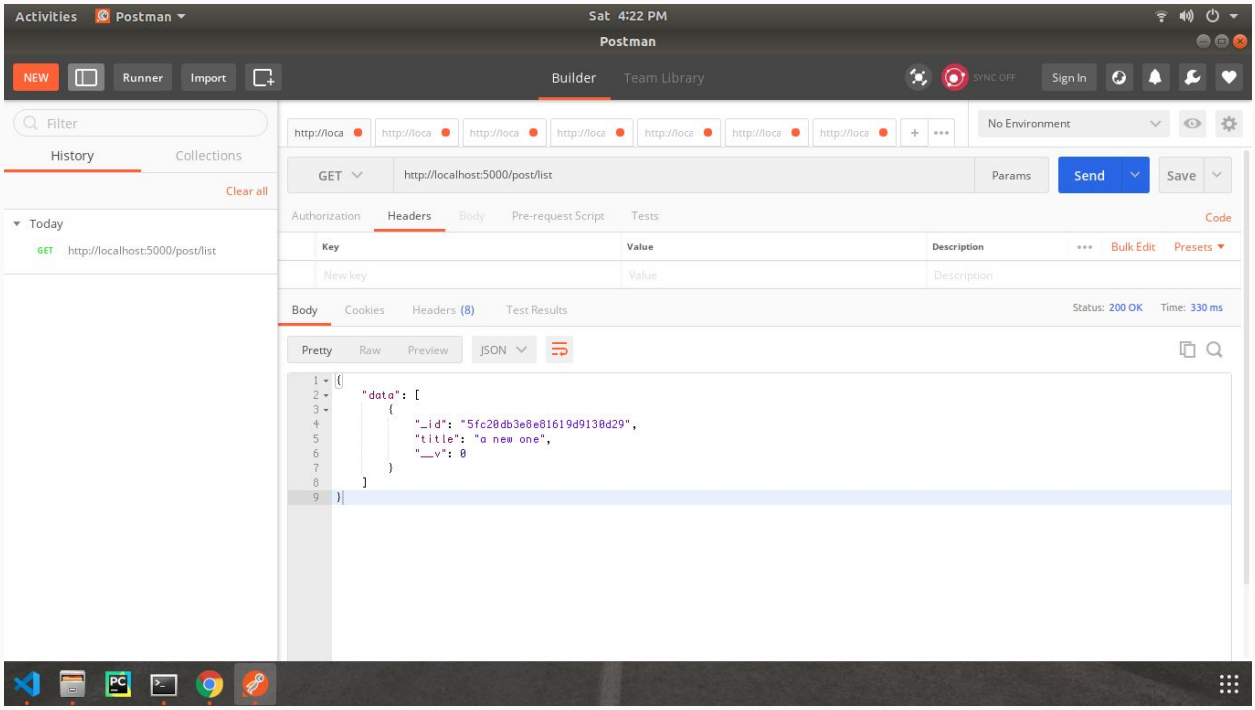
  "title": "a new one",

  "__v": 0

  }

  ]

  }

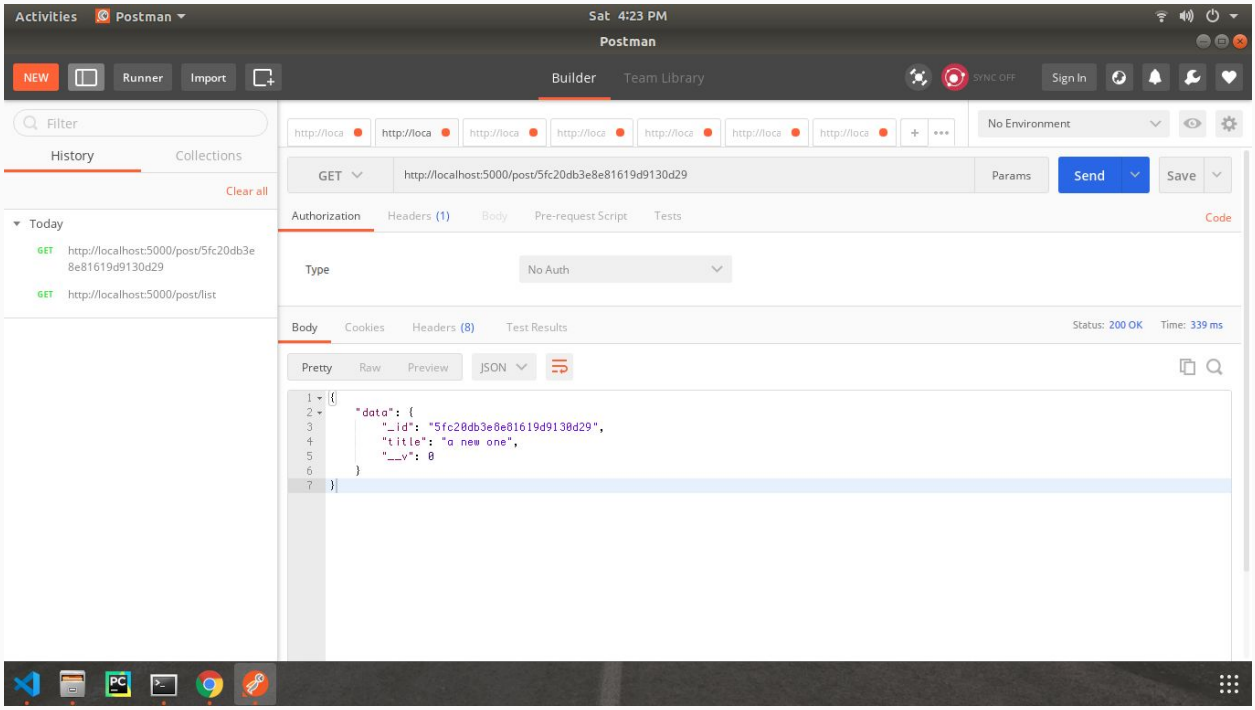


- To get a specific post :

  In http://localhost:5000/post/5fc20868a976dd1538ab4fb2 (ie. http://localhost:5000/post/id) make a GET request

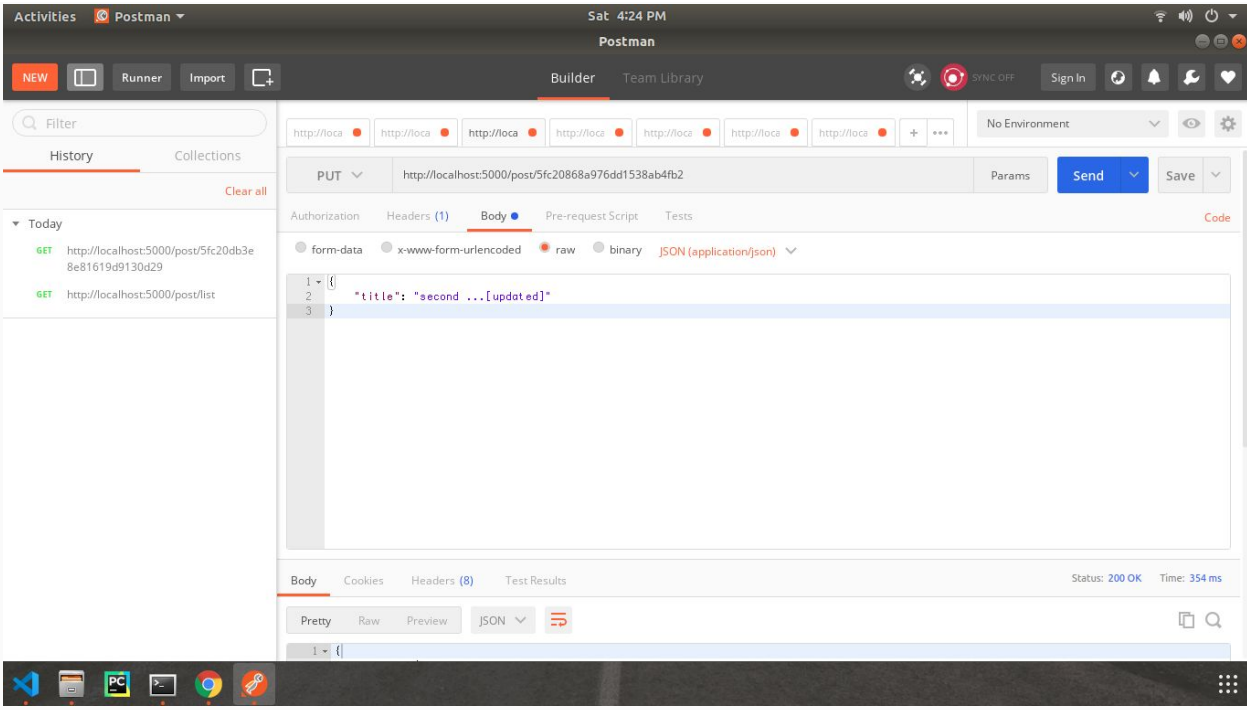
  You will get back and object :

```
{
   "data": {
       "_id": "5fc20db3e8e81619d9130d29",
       "title": "a new one",
       "__v": 0
   }
}
```



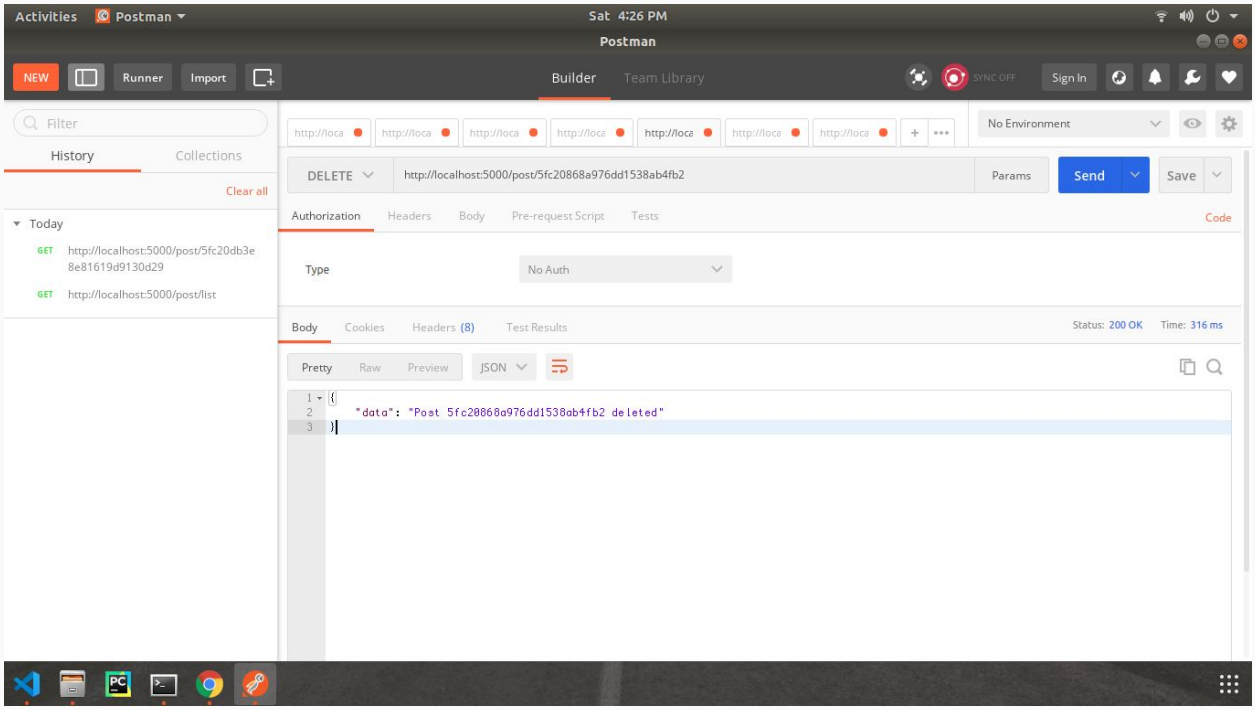
- To update a post:

  In http://localhost:5000/post/5fc20868a976dd1538ab4fb2 make a PUT request.



- To delete a post:

  In http://localhost:5000/post/5fc20868a976dd1538ab4fb2 make a DELETE request



PS. make sure to use the correct post id.

In this way, we can create a basic CRUD application in Bluemonk.
The controller module can be integrated with the Permission layer. We will learn that in the next tutorial.

This tutorial helped you understand the project structure, routes, and built-in controllers. This is just a basic overview of working with the framework. We will learn about other features in another tutorial.