Steps to Building an API

A Camariana – MDI'a National Diploma in Computer Science

Once you identify your project name and have your schema in place, follow the followings steps to get up and running with your project.

Step 1

Go to github.com and create a new repository.

- The name of the repository should be the name of your project.
- add a description if you want
- add a README
- add a .gitignore but choose **node**

For me, my project name is **bintaface**

Once, you create the repository clone it to your local computer.

Next, open the project with VS Code

Step 2

Once you open the project, if all is good, you should see the following two files

- .gitignore
- README.md

Nice progress, let's move on.

Within VS Code go to Terminal on the menu on top and click on New Terminal

Within the terminal type the following to initialise the project

```
npm init -y
```

Note: this will automatically create a package.json file

Step 3

Now lets install the packages we need to for our API. Here's is a list of packages we need for now

- express
- mongoose
- nodemon

```
1  npm install express mongoose
```

and we install nodemon as dev dependency

```
1  npm install --save-dev nodemon
```

Note: You need to be connected to the internet for this work

You also need to make sure you have installed mongoDB locally in your system.

Step 4

Project struture,

Within the root of the project, create a folder called **src**
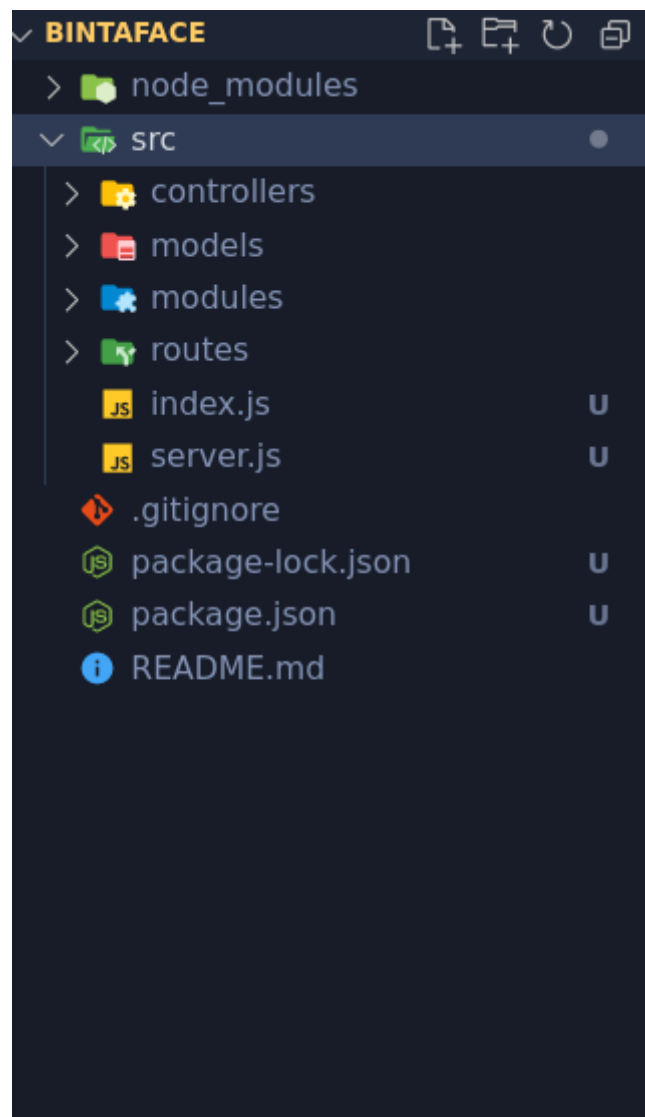
Inside the /src folder create the following folders:

- routes
- controllers
- models
- modules or utilities (the same idea, stick to one)

Inside the /src folder create the following files:

- index.js
- server.js

Here's is an example of how you project should look like for now.

Step 5

The initial server code for our express API

Within the **server.js** do the following code

```
1  const express = require('express');
2  const app = express();
3
4
5
6  app.use(express.json());
7  app.use(express.urlencoded({ extended: true }));
8
9
10 app.get('/', (req, res) => {
11   res.status(200).json({ message: 'Peace' });
12 });
13
14
15 module.exports = app;
```

And within the **index.js** do the following code

```
1  const app = require('./server');
2
3  const PORT = 3001;
4
5  app.listen(PORT, () => {
6      console.log(`Rest API listening on port ${PORT}`);
7  });
```

So alot is happening here with few lines of code, thanks to the power of express.

Everything will be explain soon

Let's update our **package.json** to include the following start script in the "scripts" section:

The script section looks like this

```
1  "scripts": {
2      "test": "echo \"Error: no test specified\" && exit 1"
3  },
```

Now change it to this

```
1  "scripts": {
2      "start": "node src/index.js",
3      "dev": "nodemon src/index.js"
4  },
```

We can check to see if things are working as expected by running our newly created start script.

Within the terminal in VS Code, run the following command

```
1  npm run dev
```

You should see nodemon doing its thing and

a message saying Rest API listening on port 3001

In your browser, if you go to the address, http://localhost:3001/ you should see

```
1  {
2     "message": "Peace"
3  }
```

Congrats, you have build a simple API.

Step 6

Connecting to our database

Before we actually connect to our database, let's create a connection module to use to connect to our database.

Within the **/modules** folder, create a file called **connect.js** and do the following code

```
1  const mongoose = require('mongoose');
2
3  const URI = 'mongodb://localhost:27017/bintaface';
4
5  const connect = () => {
6      return mongoose.connect(URI, {
7          useNewUrlParser: true,
8          useUnifiedTopology: true,
9          autoIndex: false,
10     })
11 };
12
13 module.exports = connect;
```

Note: Don't forget to update the database name at the end of the URI to your own database name

this line

```
1  const URI = 'mongodb://localhost:27017/bintaface';
```

Next up let's require our connect module in the **server.js** file and connect to our database

Here's the code to that,

First we require the connect module like this:

```
1  const connect = require('./modules/connect');
```

And here is the connection function to include in the **server.js** after requiring it

```
1  // Connect to Database
2  void (async () => {
3    try {
4      await connect();
5      console.log('connected to database');
6    } catch (error) {
7      console.log('error connecting to database:', error.message);
8    }
9  })();
```

This will automatically create your database in mongoDB, you can use compass to make sure the database is created by simply looking at the list of databases.

Here's the rest of the updated **server.js** file

```
1  const express = require('express');
2  const app = express();
3
4  const connect = require('./modules/connect');
5
6
7  app.use(express.json());
8  app.use(express.urlencoded({ extended: true }));
9
10
11 // Connect to Database
12 void (async () => {
13   try {
14     await connect();
15     console.log('connected to database');
16   } catch (error) {
17     console.log('error connecting to database:', error.message);
18   }
19 })();
20
21
22 app.get('/', (req, res) => {
23   res.status(200).json({ message: 'Peace' });
24 });
25
26
27
28 module.exports = app;
```

Nice progress, if you get here.

Step 7

Translating our schemas into models

This process is the same for all your models, the difference will be the name of the model/entity and it's attributes

1. **The model**

Let's start with the user model

Inside the **/models** folder, create a file called **user.js** and do the followig code

```
1  const mongoose = require('mongoose')
2
3  const userSchema = new mongoose.Schema(
4    {
5      email: {
6        type: String,
7        required: true,
8        unique: true,
9        trim: true
10      },
11      password: {
12        type: String,
13        required: true
14      },
15      firstname: {
16        type: String,
17        trim: true,
18        maxlength: 25
19      },
20      minit: {
21        type: String,
22        trim: true,
23        maxlength: 25
24      },
25      lastname: {
26        type: String,
27        trim: true,
28        maxlength: 25
29      },
30      role: {
31        type: String,
32        enum: ['Admin', 'User'],
33        default: 'User'
34      }
35    },
36    { timestamps: true }
37  )
```

```
38   }
39
40   userSchema.set('toJSON', {
41     transform: (document, returnedObject) => {
42       returnedObject.id = returnedObject._id.toString()
43       delete returnedObject._id
44       delete returnedObject.__v
45       // the password should not be revealed
46       delete returnedObject.password
47     }
48   })
49
50
51   const User   = mongoose.model('user', userSchema)
52
53   module.exports = User
```

2. The controller

Once we create the user model and export the model, we can happily require the model in our controller and create a new user.

Inside the **/controllers** folder, create a file called **user.js** and do the followig code

```
1    const User = require('../models/user');
2
3    const createUser = async (req, res) => {
4      const content = req.body;
5
6      console.log(content);
7      try {
8        const user = await User.create({ ...content });
9
10       return res.status(201).json({ data: user });
11     } catch (error) {
12       console.log(error);
13       return res.status(500).end();
14     }
15   };
16
17
18
19   module.exports = {
20     createUser
21   };
```

3. The router

Then we can create a new file called **router.js** inside the **/routes** to handle all of routes of the entire application. But for now will do only the user router.

So, within the **router.js** file

let's do the following code

```
1   const express = require('express');
2   const router = express.Router();
3
4   const { createUser } = require('../controllers/user');
5
6
7   // User route
8   router.post('/user', createUser);
9
10
11  module.exports = router;
```

Finally and finally, we can add our routes to the **server.js** for routing our users to the various endpoints.

Let's update our **server.js** to include our route and put it to use.

First we are going to require the routes like this:

```
1   const routes = require('./routes/router');
```

then we can put it to use like this:

```
1   app.use('/api', routes);
```

Here is the updated **server.js** file with the above code included

```
1   const express = require('express');
2   const app = express();
3
4   const connect = require('./modules/connect');
5   const routes = require('./routes/router');
6
7
8   app.use(express.json());
9   app.use(express.urlencoded({ extended: true }));
10
11
12  // Connect to Database
13  void (async () => {
14    try {
15      await connect();
```

```
16        console.log('connected to database');
17    } catch (error) {
18        console.log('error connecting to database:', error.message);
19    }
20 })();
21
22
23 app.get('/', (req, res) => {
24    res.status(200).json({ message: 'Peace' });
25 });
26
27
28 app.use('/api', routes);
29
30
31 module.exports = app;
```

Step 8

Let's test to see, if wee would be able to create a new user using the VS CODE rest client

Within the **root folder** of the project creates a new folder called **requests**

Inside requests create a file called **create-one.rest**

In that file, let's send a POST request to create a new user, see the code below for that.

```
1  POST http://localhost:3001/api/user HTTP/1.1
2  content-type: application/json
3
4  {
5     "email": "ac@camariana.gm",
6     "password": "secret",
7     "firstname": "A",
8     "minit": "",
9     "lastname": "Camariana"
10 }
```

Take note of line 1

```
1  POST http://localhost:3001/api/user HTTP/1.1
```

if you lcarefuly study the HTTP protocol, you will notice the following

- 3001 – that's the port
- /api/user – that our user enpoint

Within the VS CODE rest client, you will see on top of line 1 **Send Request**, click it.

If all goes well you should see a response like this:

```
1   HTTP/1.1 201 Created
2   X-Powered-By: Express
3   Content-Type: application/json; charset=utf-8
4   Content-Length: 210
5   ETag: W/"d2-/I0yK/fE80rXZdmXASBEqN4fWZs"
6   Date: Sat, 15 Oct 2022 07:06:03 GMT
7   Connection: close
8
9   {
10    "data": {
11      "email": "ac@camariana.gm",
12      "firstname": "A",
13      "minit": "",
14      "lastname": "Camariana",
15      "role": "User",
16      "createdAt": "2022-10-15T07:06:03.201Z",
17      "updatedAt": "2022-10-15T07:06:03.201Z",
18      "id": "634a5bdb441c0826a1b5e678"
19    }
20  }
```

This is good, this means, we have successfully created one user.

Please add another different 2 – 5 users by updating the content of the POST request in the create-one.rest file and send a request.

Step 9

In this step we will learn how to retrieve data for **all users**. This method should work for getting data from any table.

So let's go to the **user controller**, in the **/controllers/user.js** let's do the following code to get all the users from the database

```
1   const getAllUsers = async (req, res) => {
2     try {
3       const users = await User.find({});
4
5       return res.status(201).json({ data: users });
6     } catch (error) {
7       console.log(error);
8       return res.status(500).end();
9     }
10  };
```

Once you add that handle, let's export it so that our router can see it. To do that update the module.exports to add `getAllUsers` to the export list.

```
1  module.exports = {
2    createUser,
3    getAllUsers
4  };
```

So far our **user.js** controller should like this

```
1  const User = require('../models/user');
2
3  const createUser = async (req, res) => {
4    const content = req.body;
5
6    console.log(content);
7    try {
8      const user = await User.create({ ...content });
9
10     return res.status(201).json({ data: user });
11   } catch (error) {
12     console.log(error);
13     return res.status(500).end();
14   }
15 };
16
17 const getAllUsers = async (req, res) => {
18   try {
19     const users = await User.find({});
20
21     return res.status(201).json({ data: users });
22   } catch (error) {
23     console.log(error);
24     return res.status(500).end();
25   }
26 };
27
28
29
30 module.exports = {
31   createUser,
32   getAllUsers
33 };
```

Once you do that let's require it in the **/routers/route.js** file.

currently our require for the user controller looks like this

```
1  const { createUser } = require('../controllers/user');
```

let's add the new `getAllUsers` handler to that line. it should now look like this:

```
1  const { createUser, getAllUsers } = require('../controllers/user');
```

Finally, let's add the route to the user route

```
1  router.get('/user', getAllUsers);
```

The final code for the /route.js should like this

```
1   const express = require('express');
2   const router = express.Router();
3
4   const { createUser, getAllUsers } = require('../controllers/user');
5
6
7   // User route
8   router.post('/user', createUser);
9   router.get('/user', getAllUsers);
10
11
12  module.exports = router;
```

Note the new changes in the router.js file, especially the verb is now **get**, not **post** like when we were creating.

Let's test and see of we will get all the users using out rest client

Inside **requests** create a file called **get-all.rest**

In that file, let's send a get request to get all the users, see the code below for that.

```
1  GET http://localhost:3001/api/user
```

You see a list of all the users in the you db like the one below. I only have one user

```
1   {
2     "data": [
3       {
4         "email": "ac@camariana.gm",
5         "firstname": "A",
6         "minit": "",
7         "lastname": "Camariana",
8         "role": "User",
9         "createdAt": "2022-10-15T07:06:03.201Z",
10        "updatedAt": "2022-10-15T07:06:03.201Z",
11        "id": "634a5bdb441c0826a1b5e678"
```

```
12        }
13      ]
14  }
```

Copy and paste your results to slack, let me see your users.

Step 10

We wil now create a handler to get only **one user**.

So let's go back to the **user controller**. in the **/controllers/user.js** let's do the following code to get one user from the database

```
 1  const getOneUser = async (req, res) => {
 2    const id = req.params.id
 3    try {
 4      const user = await User.findOne({ _id: id });
 5
 6      if (!user) {
 7        return res.status(400).json({ message: 'user not found' });
 8      }
 9      return res.status(201).json({ data: user });
10    } catch (error) {
11      console.log(error);
12      return res.status(500).end();
13    }
14  };
```

Once you add that handle, let's export it so that our router can see it. like this below

```
 1
 2  module.exports = {
 3    createUser,
 4    getAllUsers,
 5    getOneUser
 6  };
```

Next, require it in the **/routers/route.js** file. and add the route

```
 1  const { createUser, getAllUsers, getOneUser } =
      require('../controllers/user');
```

and add it to the route

```
 1  // User route
 2  router.post('/user', createUser);
 3  router.get('/user', getAllUsers);
 4  router.get('/user/:id', getOneUser);
```

Note how we add the parameter to the path `'/user/:id'`

Let's test and see of we will get one user using out rest client

Inside **requests** create a file called **get-one.rest**

In that file, let's send a get request to get one user, see the code below for that.

```
1  GET http://localhost:3001/api/user/6306ae79da5f62731c434830
```

Note: look at what is after the forward slash after the /user/... <-- this is the id of the user we are looking for.

If you use this id it will say not found. so copy one of the id from your users response and paste it after the /user/here and send a request, you should get that single user data like mine here

```
1  {
2    "data": {
3      "email": "ac@camariana.gm",
4      "firstname": "A",
5      "minit": "",
6      "lastname": "Camariana",
7      "role": "User",
8      "createdAt": "2022-10-15T07:06:03.201Z",
9      "updatedAt": "2022-10-15T07:06:03.201Z",
10     "id": "634a5bdb441c0826a1b5e678"
11   }
12 }
```

Step 11

We wil now create a handler to update only **one user**.

So let's go back to the **user controller**. in the **/controllers/user.js** let's do the following code to update one user from the database

```
1  const updateOne = async (req, res) => {
2    const id = req.params.id
3    const content = req.body
4
5    try {
6      const user = await User.findOneAndUpdate(
7        { _id: id },
8        content,
9        { new: true }
10     );
11
12     if (!user) {
```

```
13        return res.status(400).json({ message: 'user not found' });
14      }
15      return res.status(201).json({ data: user });
16    } catch (error) {
17      console.log(error);
18      return res.status(500).end();
19    }
20  };
```

Once you add that handler, let's export it, so that our router can see it. like this below

```
1  module.exports = {
2    createUser,
3    getAllUsers,
4    getOneUser,
5    updateOne
6  };
```

Next, require it in the **/routers/route.js** file. and add the route

```
1  const { createUser, getAllUsers, getOneUser, updateOne } =
     require('../controllers/user');
2
```

and add it to the route

```
1  // User route
2  router.post('/user', createUser);
3  router.get('/user', getAllUsers);
4  router.get('/user/:id', getOneUser);
5  router.put('/user/:id', updateOne);
```

Note how we add the parameter to the path `'/user/:id'` and note the verb here is **put** instead **get**

Let's test and see if we will update one user using out rest client

Inside **requests** create a file called **update-one.rest**

In that file, let's send a get request to update one user, see the code below for that.

```
1  PUT http://localhost:3001/api/user/634a5bdb441c0826a1b5e678
2  content-type: application/json
3
4  {
5     "email": "ac@camariana.gm",
6     "firstname": "A",
7     "minit": "",
8     "lastname": "Camara",
9     "role": "Admin"
10  }
```

Note: look at what is after the forward slash after the /user/... <-- this is the id of the user we are looking for to update.

If you use this id it will say not found. so copy one of the id from your users response and paste it after the /user/here and send a request, you should get that single user data like mine here

```
1  {
2     "data": {
3        "email": "ac@camariana.gm",
4        "firstname": "A",
5        "minit": "",
6        "lastname": "Camara",
7        "role": "Admin",
8        "createdAt": "2022-10-15T07:06:03.201Z",
9        "updatedAt": "2022-10-21T12:26:10.448Z",
10       "id": "634a5bdb441c0826a1b5e678"
11    }
12  }
```

So what changes?

Before the changes the user data was like this:

```
1  {
2     "data": {
3        "email": "ac@camariana.gm",
4        "firstname": "A",
5        "minit": "",
6        "lastname": "Camariana",
7        "role": "User",
8        "createdAt": "2022-10-15T07:06:03.201Z",
9        "updatedAt": "2022-10-15T07:06:03.201Z",
10       "id": "634a5bdb441c0826a1b5e678"
11    }
12  }
```

So what changes?

I updated the **lastname** and the **role**. Compare the above file and the updated one above it.