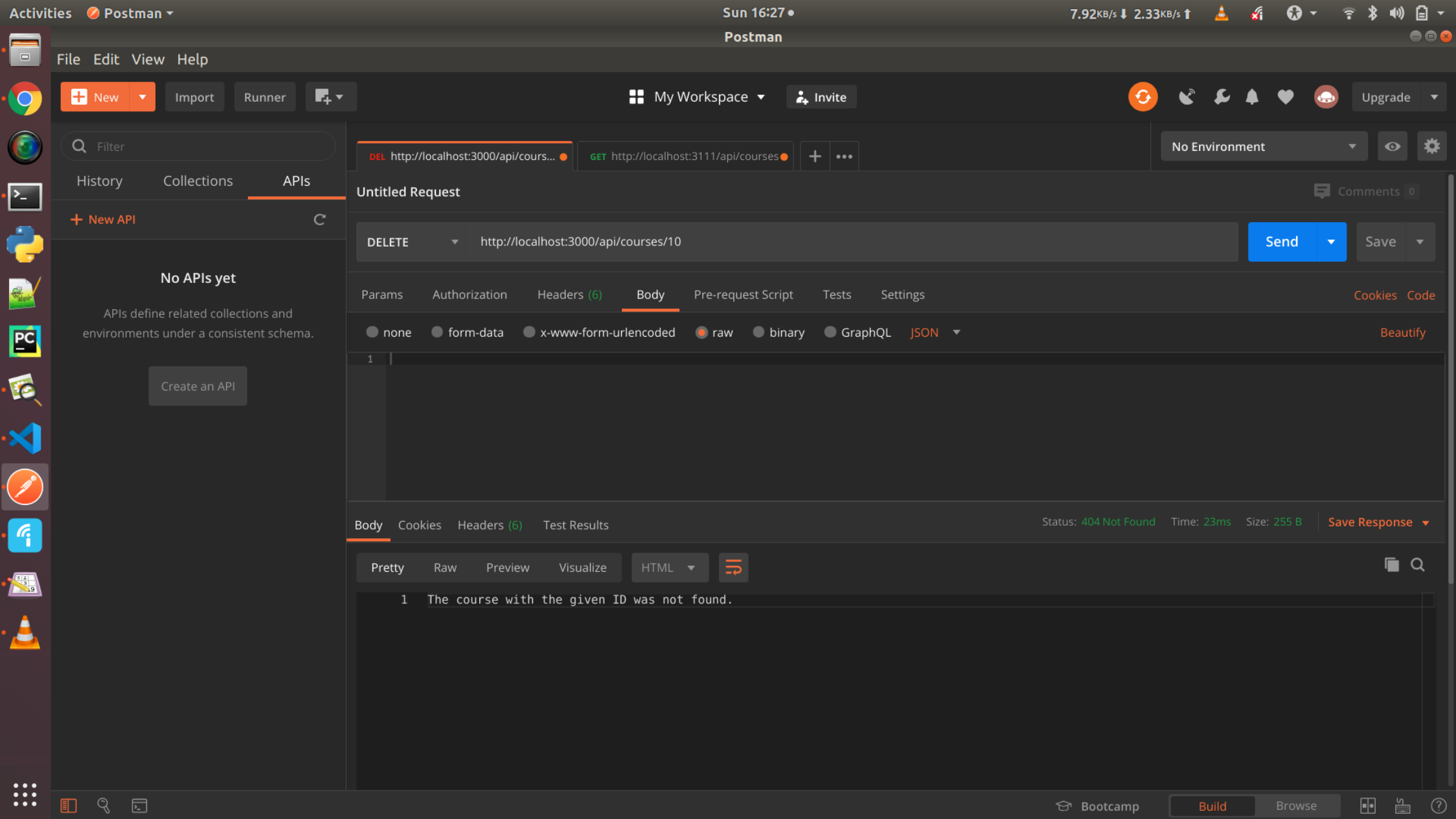
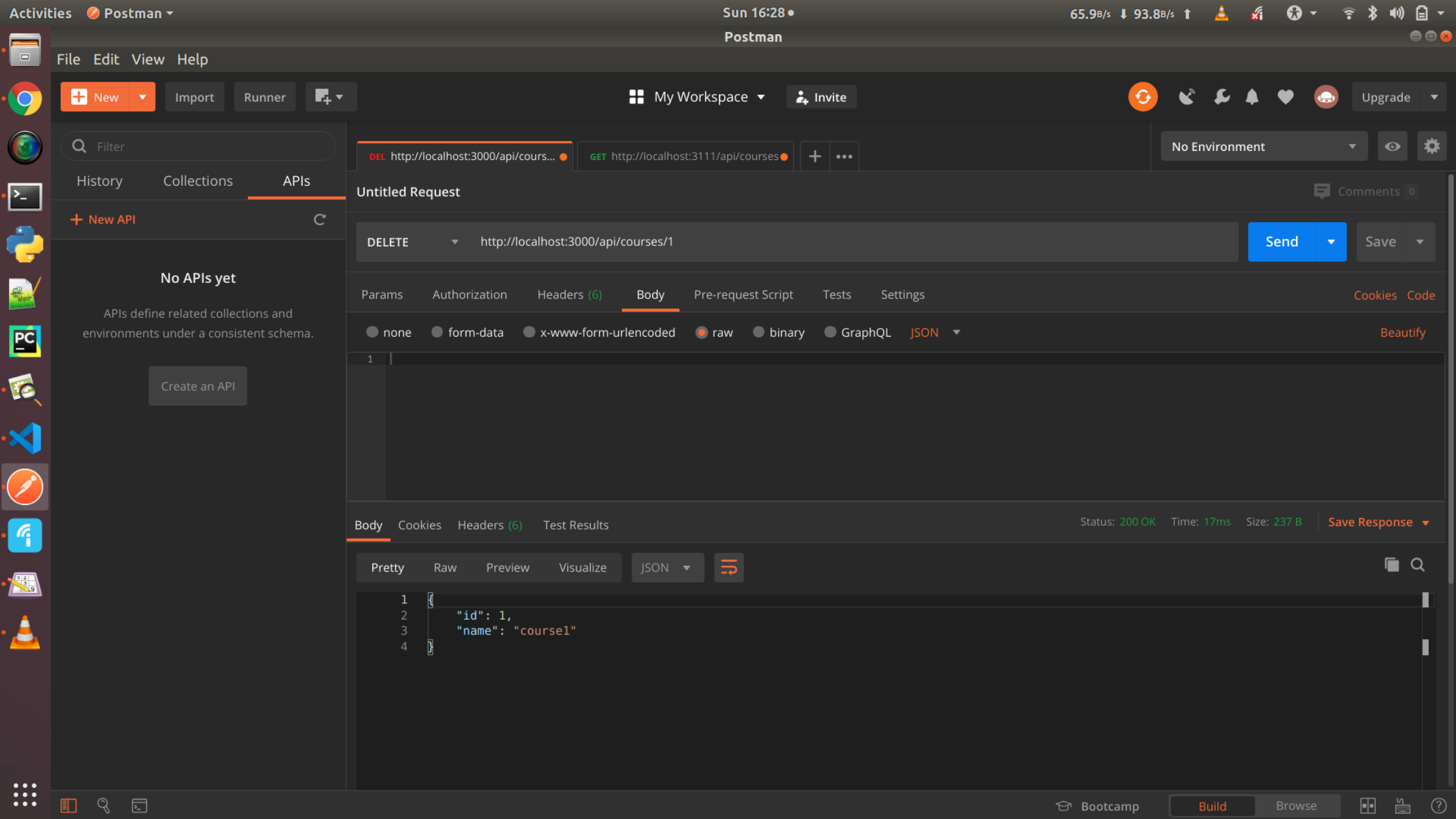


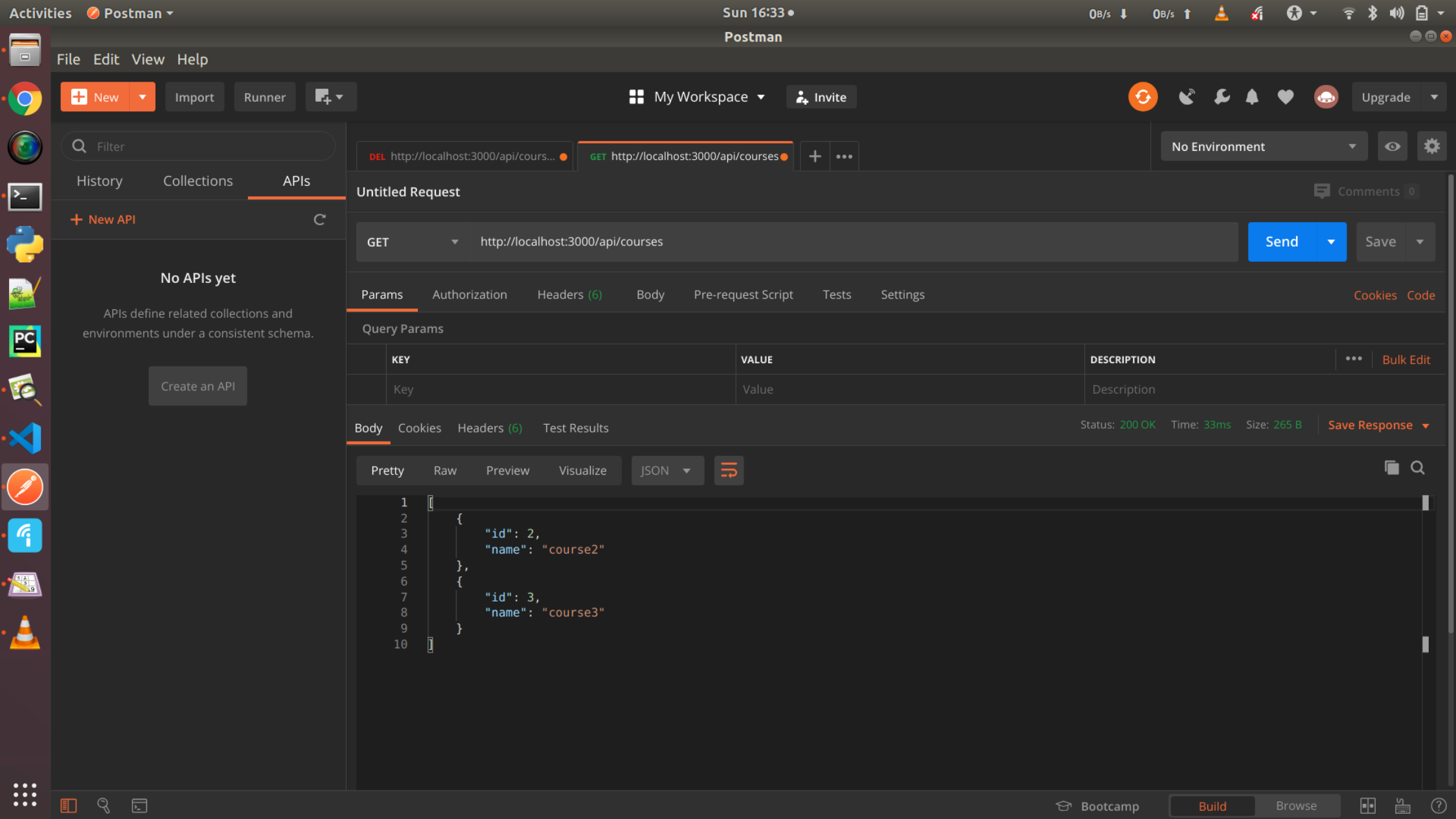
Handling DELETE Requests



Mosh Hamedani
programmingwithmosh.com







No APIs yet

APIs define related collections and environments under a consistent schema.

Create an API

Untitled Request

Comments 0

GET http://localhost:3000/api/courses

Send Save

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Cookies Code

Query Params

| KEY | VALUE | DESCRIPTION | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
| Key | Value | Description | | |

Body Cookies Headers (6) Test Results

Status: 200 OK Time: 33ms Size: 265 B Save Response

Pretty Raw Preview Visualize

JSON




```
1 [{
2   {
3     "id": 2,
4     "name": "course2"
5   },
6   {
7     "id": 3,
8     "name": "course3"
9   }
10 ]
```

Fixing a Bug



JS index.js X

13. Handling HTTP Delete Requests > JS index.js >  app.delete('/api/courses/:id') callback

```

48 app.put('/api/courses/:id', (req, res) => {
49   // Lookup up the course
50   // If not existing, return 404 - Not found
51   const course = courses.find(c => c.id === parseInt(req.params.id));
52   if (!course) return res.status(404).send('The course with the given ID was not found.');
```

```

53
54   // Validate
55   // If invalid, return 400 - Bad request
56   // const schema = {
57   //   name: Joi.string().min(3).required()
58   // };
59   // const result = Joi.validate(req.body, schema);
60   //const result = validateCourse(req.body);
61   const {error} = validateCourse(req.body); // It is equivalent to result.error
62   // This is known as object destructuring {error}
63   // if (result.error) {
64   //   res.status(400).send(result.error.details[0].message);
65   //   return;
66   // }
67   if (error) return res.status(400).send(error.details[0].message);
68
69   // Update course
70   // Return the updated course
71   course.name = req.body.name;
72   res.send(course);
73 });
```

```
74 app.delete('/api/courses/:id',(req, res)=>{
75     // Look up the course
76     // Not existing, return 404 - Not Found
77     const course = courses.find(c => c.id === parseInt(req.params.id));
78     if (!course) return res.status(404).send('The course with the given ID was not found.');
```

Ln 78, Col 24 Spaces: 4 UTF-8 LF Babel JavaScript JavaScript Standard Style

ActivitiesVisual Studio Code

Sun 16:37

0B/s0B/s

index.js - Applications - Visual Studio Code

FileEditSelectionViewGoRunTerminalHelp

EXPLORER

OPEN EDITORS

APPLICATIONS

OUTLINE

JS index.js 13. Handling HTTP Delete Requests

01. Introduction

02. RESTful Services

03. Introducing Express

04. Building Your First Web Server

05. Nodemon

06. Environment Variables

07. Route Parameters

08. Handling HTTP GET Requests

09. Handling HTTP POST Requests

10. Calling Endpoints Using Postman

11. Input Validation

12. Handling HTTP PUT Requests

JS index.js

13. Handling HTTP Delete Requests

JS index.js

14. Project- Build the Genres API

15. Building RESTful APIs with Express Recap

JS index.js

13. Handling HTTP Delete Requests > JS index.js > app.delete('/api/courses/:id') callback

Complexity is 4 Everything is cool!

app.delete('/api/courses/:id',(req, res)=>{
 // Look up the course
 // Not existing, return 404 - Not Found
 const course = courses.find(c => c.id === parseInt(req.params.id));
 if (!course) return res.status(404).send('The course with the given ID was not found.');

// Delete
 const index = courses.indexOf(course);
 courses.splice(index,1);

 // Return the same course
 res.send(course);
});

Complexity is 3 Everything is cool!

app.get('/api/courses/:id', (req, res) => {
 const course = courses.find(c => c.id === parseInt(req.params.id));
 if (!course) res.status(404).send('The course with the given ID was not found.');

res.send(course);
});

//PORT
const port = process.env.PORT || 3000;//Initialize PORT in the terminal : export PORT=3333
app.listen(port, () => console.log(`Listening to port \${port} ...`));
// To add environment variable use :
// Ubuntu : export PORT=3333
// Windows : SET PORT=3333
// To Kill the activity on the port 3000 :sudo kill -9 \$(sudo lsof -t -i:3000)
// To check the active ports : sudo lsof -i -P -n | grep LISTEN

function validateCourse(course){
 const schema = {
 name: Joi.string().min(3).required()
 };
 return Joi.validate(course, schema);
}

master*0002✓ javascript | ✓ index.js

Ln 78, Col 24Spaces: 4UTF-8LFBabel JavaScriptJavaScript Standard Style

EXPLORER

JS index.js X

> 01. Introduction

> 02. RESTful Services

> 04. Building Your First Web Server

> 06. Environment Variables

> 08. Handling HTTP GET Requests

> 10. Calling Endpoints Using Postman

12. Handling HTTP PUT Requests

13. Handling HTTP Delete Requests

14. Project- Build the Genres API

> 15. Building RESTful APIs with Express Recap

13. Handling HTTP Delete Requests > JS index.js > app.get('/api/courses/:id') callback

Complexity is 4 Everything is cool!

```
87 app.get('/api/courses/:id', (req, res) => {
88     const course = courses.find(c => c.id === parseInt(req.params.id));
89     if (!course) return res.status(404).send('The course with the given ID was not found.');
90     res.send(course);
91 })
```

```
91 });
92 //PORT
93 const port = process.env.PORT || 3000; //Initialize PORT in the terminal : export PORT=3333
94 app.listen(port, () => console.log(`Listening to port ${port} ....`));
95 // To add environment variable use :
96 // Ubuntu : export PORT=3333
97 // Windows : SET PORT=3333
98 // To Kill the activity on the port 3000 : sudo kill -9 $(sudo lsof -t -i:3000)
99 // To check the active ports : sudo lsof -i -P -n | grep LISTEN
```

```
101 function validateCourse(course){
102     const schema = {
103         name: Joi.string().min(3).required()
104     };
105     return Joi.validate(course, schema);
106 }
```


ActivitiesVisual Studio Code

Sun 16:3986.0B/s585B/s

index.js - Applications - Visual Studio Code

FileEditSelectionViewGoRunTerminalHelp

EXPLORER

OPEN EDITORS

APPLICATIONS

OUTLINE

JS index.js

13. Handling HTTP Delete Requests

01. Introduction

02. RESTful Services

03. Introducing Express

04. Building Your First Web Server

05. Nodemon

06. Environment Variables

07. Route Parameters

08. Handling HTTP GET Requests

09. Handling HTTP POST Requests

10. Calling Endpoints Using Postman

11. Input Validation

12. Handling HTTP PUT Requests

JS index.js

13. Handling HTTP Delete Requests

JS index.js

14. Project- Build the Genres API

15. Building RESTful APIs with Express Recap

JS index.js

13. Handling HTTP Delete Requests > JS index.js > app.post('/api/courses') callback

Complexity is 3 Everything is cool!

app.post('/api/courses', (req, res) => {
 const {error} = validateCourse(req.body); // It is equivalent to result.error
 // This is known as object destructuring {error}
 if (error) return res.status(400).send(error.details[0].message);
 // const schema = {
 // name: Joi.string().min(3).required()
 // };
 // const result = Joi.validate(req.body, schema);
 // //console.log(result);
 // if (result.error) {
 // //res.status(400).send(result.error);
 // res.status(400).send(result.error.details[0].message);
 // return;
 // }
 // // if(!req.body.name || req.body.name.length < 3){
 // // // 400 Bad Request
 // // res.status(400).send('Name is required and should be minimum 3 characters.');

Complexity is 6 It's time to do something...

app.put('/api/courses/:id', (req, res) => {
 // Lookup up the course
 // If not existing, return 404 - Not found
 const course = courses.find(c => c.id === parseInt(req.params.id));
 if (!course) return res.status(404).send('The course with the given ID was not found.');

Ln 22, Col 70Spaces: 4UTF-8LFBabel JavaScriptJavaScript Standard Style