

Learn The MERN Stack

[youtube playlist](#)

Table of Contents

1 Express & MongoDB Rest API.....	3
1.1 Preconditions.....	3
1.2 Installing 3 rd party libraries.....	3
1.3 Change default run scripts.....	3
1.4 Create a simple express app.....	3
1.5 Postman.....	4
1.6 Create a simple test route with text response.....	4
1.7 Setup a response (as a JSON object) and a status code.....	4
1.8 Split out all the routes into different files (entities).....	5
.....	5
.....	5
1.9 Add different routes for Goal entity.....	6
1.10 Add Goal controller.....	7

1 Express & MongoDB Rest API

1.1 Preconditions

Create a new repository in gitHub. Clone your new empty repo to your local computer. Start a new npm project:

CLI=> *npm init -y*

Create backend folder to store all the files that related to backend logic.

Create a main file server.js

1.2 Installing 3rd party libraries

CLI=> *npm i express dotenv mongoose colors*

CLI=> *npm i -D nodemon*

1.3 Change default run scripts

Open package.json file (it should be in root folder of your project)

```
"scripts": {  
  "start": "node backend/server.js",  
  "server": "nodemon backend/server.js"  
},
```

Now you can run your server using next command:

CLI=> *npm run server*

1.4 Create a simple express app

```
const express = require('express');  
const dotenv = require('dotenv').config();  
const port = process.env.PORT || 5000;  
const app = express();  
app.listen(port, () => console.log(`Server was started on port ${port}`));
```



The screenshot shows a code editor with a file named 'server.js' open. The code is as follows:

```
JS server.js M X  
backend > JS server.js > ...  
1  const express = require('express');  
2  const dotenv = require('dotenv').config();  
3  const port = process.env.PORT || 5000;  
4  
5  const app = express();  
6  
7  app.listen(port, () => console.log(`Server was started on port ${port}`));
```

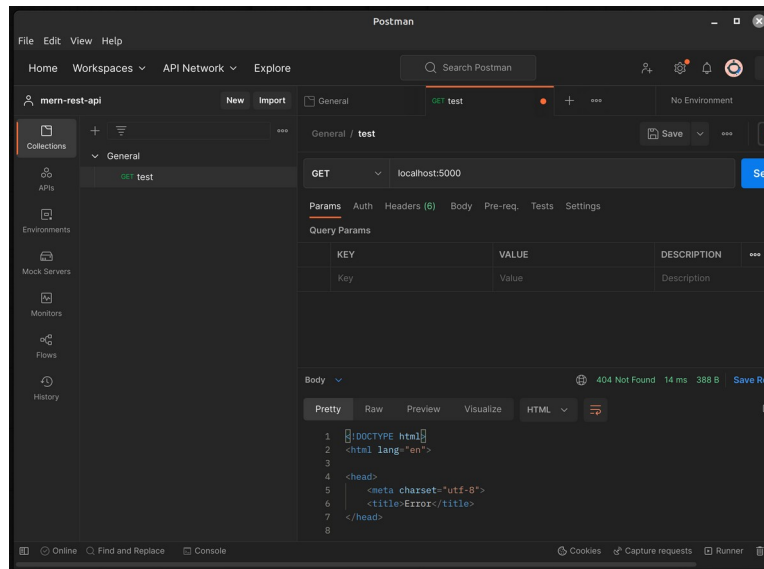


The screenshot shows a .env file with the following content:

```
.env  
1  NODE_ENV = development  
2  PORT = 5000
```

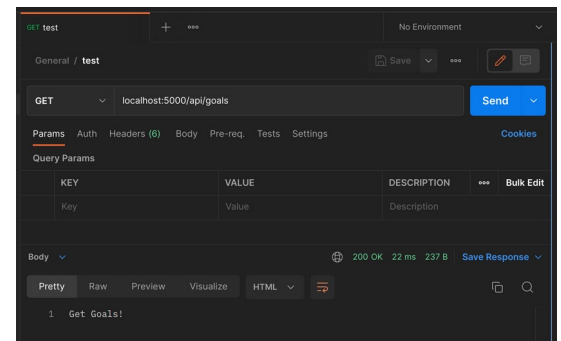
1.5 Postman

Download Postman. Create a new collection and a first test request.



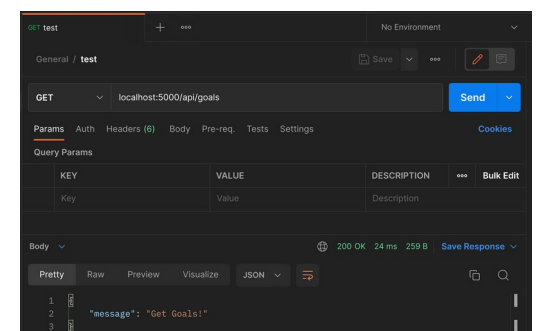
1.6 Create a simple test route with text response

```
JS server.js M X
backend > JS server.js > ...
1  const express = require('express');
2  const dotenv = require('dotenv').config();
3  const port = process.env.PORT || 5000;
4
5  const app = express();
6
7  app.get('/api/goals', (req, res) => {
8    res.send('Get Goals!');
9  });
10
11 app.listen(port, () => console.log(`Server was started on port ${port}`));
```



1.7 Setup a response (as a JSON object) and a status code

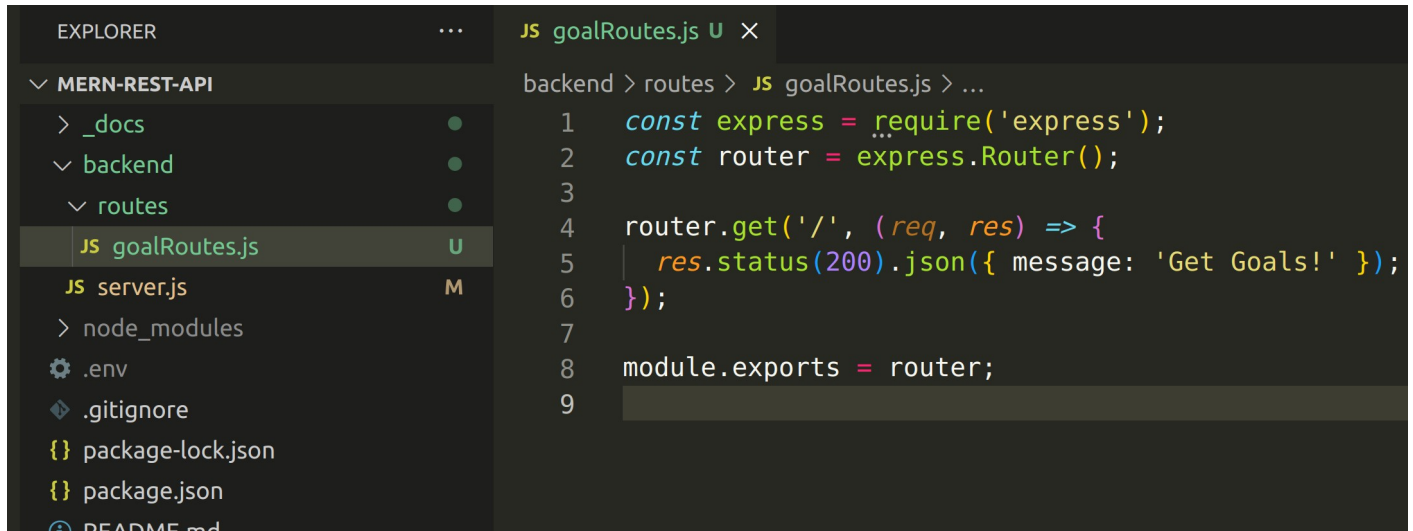
```
JS server.js M X
backend > JS server.js > ...
1  const express = require('express');
2  const dotenv = require('dotenv').config();
3  const port = process.env.PORT || 5000;
4
5  const app = express();
6
7  app.get('/api/goals', (req, res) => {
8    res.status(200).json({ message: 'Get Goals!' });
9  });
10
11 app.listen(port, () => console.log(`Server was started on port ${port}`));
```



1.8 Split out all the routes into different files (entities)

Create a new folder backend/routes to store a few main routes (for each of main entities).

Create a first route file for Goal entity.



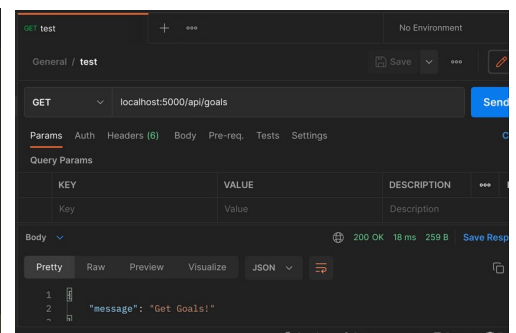
The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays the project structure: MERN-REST-API, _docs, backend, routes, JS goalRoutes.js (selected), JS server.js, node_modules, .env, .gitignore, package-lock.json, package.json, and README.md. The main editor area shows the content of goalRoutes.js:

```
backend > routes > JS goalRoutes.js > ...
1  const express = require('express');
2  const router = express.Router();
3
4  router.get('/', (req, res) => {
5    res.status(200).json({ message: 'Get Goals!' });
6  });
7
8  module.exports = router;
9
```



The screenshot shows the VS Code editor with the server.js file open. The code is as follows:

```
JS server.js M X
backend > JS server.js > ...
1  const express = require('express');
2  const dotenv = require('dotenv').config();
3  const port = process.env.PORT || 5000;
4
5  const app = express();
6
7  app.use('/api/goals', require('./routes/goalRoutes'));
8
9  app.listen(port, () => console.log(`Server was started on port ${port}`));
10
```



The screenshot shows a REST client interface. A GET request to localhost:5000/api/goals has been sent. The response is shown in the 'Body' tab, which is formatted as JSON:

```
1  {
2    "message": "Get Goals!"
3  }
```

The status bar at the bottom indicates a 200 OK response with a response time of 18 ms and a body size of 259 B.

1.9 Add different routes for Goal entity

```
JS goalRoutes.js U X
backend > routes > JS goalRoutes.js > ...
1  const express = require('express');
2  const router = express.Router();
3
4  router.get('/', (req, res) => {
5    res.status(200).json({ message: 'Get all the Goals!' });
6  });
7
8  router.post('/', (req, res) => {
9    res.status(201).json({ message: 'Set a new Goal!' });
10 });
11
12 router.put('/:id', (req, res) => {
13   res
14     .status(200)
15     .json({ message: `Update a Goal with ID = '${req.params.id}'` });
16 });
17
18 router.delete('/:id', (req, res) => {
19   res
20     .status(200)
21     .json({ message: `Delete a Goal with ID = '${req.params.id}'` });
22 });
23 module.exports = router;
```

1.10 Add Goal controller

Create a new folder backend/controllers to store all the controllers. Create a new file backend/controllers/goalController.js

```
JS server.js M X
backend > JS server.js > ...
1  const express = require('express');
2  const dotenv = require('dotenv').config();
3  const port = process.env.PORT || 5000;
4
5  const app = express();
6
7  app.use('/api/goals', require('./routes/goalRoutes'));
8
9  app.listen(port, () => console.log(`Server was started on port ${port}`));
```

```
JS goalRoutes.js U X
backend > routes > JS goalRoutes.js > ...
1  const express = require('express');
2  const router = express.Router();
3  const {
4    getGoals,
5    setGoal,
6    updateGoal,
7    deleteGoal,
8  } = require('../controllers/goalController');
9
10 router.route('/').get(getGoals).post(setGoal);
11 router.route('/:id').put(updateGoal).delete(deleteGoal);
12
13 module.exports = router;
```

```
JS goalController.js U X
backend > controllers > JS goalController.js > ...
1  // @desc   Get all the Goals
2  // @route   GET to /api/goals
3  // @access  Private
4  const getGoals = (req, res) => {
5    res.status(200).json({ message: 'Get all the Goals!' });
6  };
7
8  // @desc   Create a new Goal
9  // @route   POST to /api/goals
10 // @access  Private
11 const setGoal = (req, res) => {
12   res.status(201).json({ message: 'Set a new Goal!' });
13 };
14
15 // @desc   Update a Goal by ID
16 // @route   PUT to /api/goals/:id
17 // @access  Private
18 const updateGoal = (req, res) => {
19   res
20     .status(200)
21     .json({ message: `Update a Goal with ID = '${req.params.id}'` });
22 };
23
24 // @desc   Delete a Goal by ID
25 // @route   DELETE to /api/goals/:id
26 // @access  Private
27 const deleteGoal = (req, res) => {
28   res
29     .status(200)
30     .json({ message: `Delete a Goal with ID = '${req.params.id}'` });
31 };
32
33 module.exports = { getGoals, setGoal, updateGoal, deleteGoal };
```

