

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
1.11 Work with data from request.....	8
1.12 Express error handler.....	9
1.13 Using express-async-handler.....	10

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*

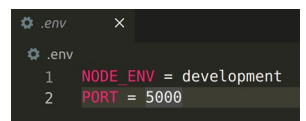
1.4 Create a simple express app

```
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}`));
```

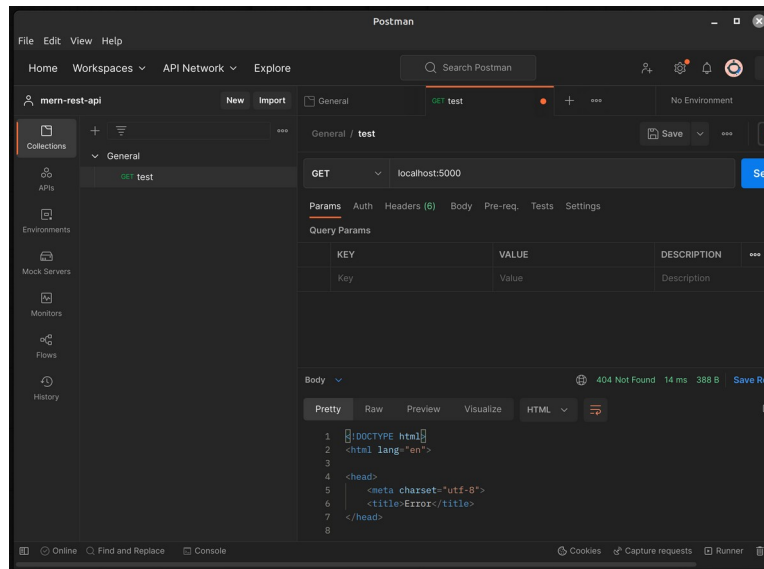


The screenshot shows a .env file with the following content:

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

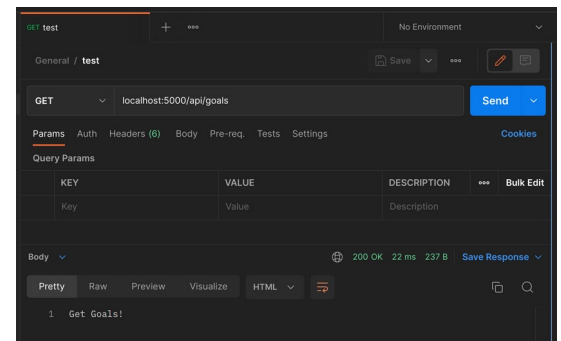
1.5 Postman

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



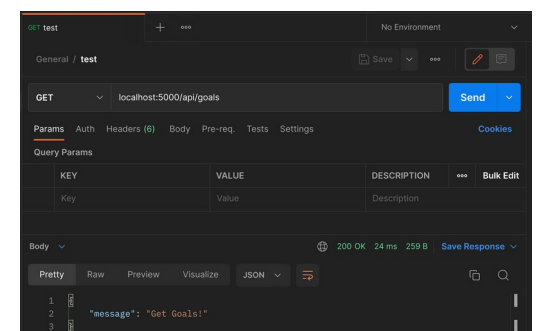
1.6 Create a simple test route with text response

```
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

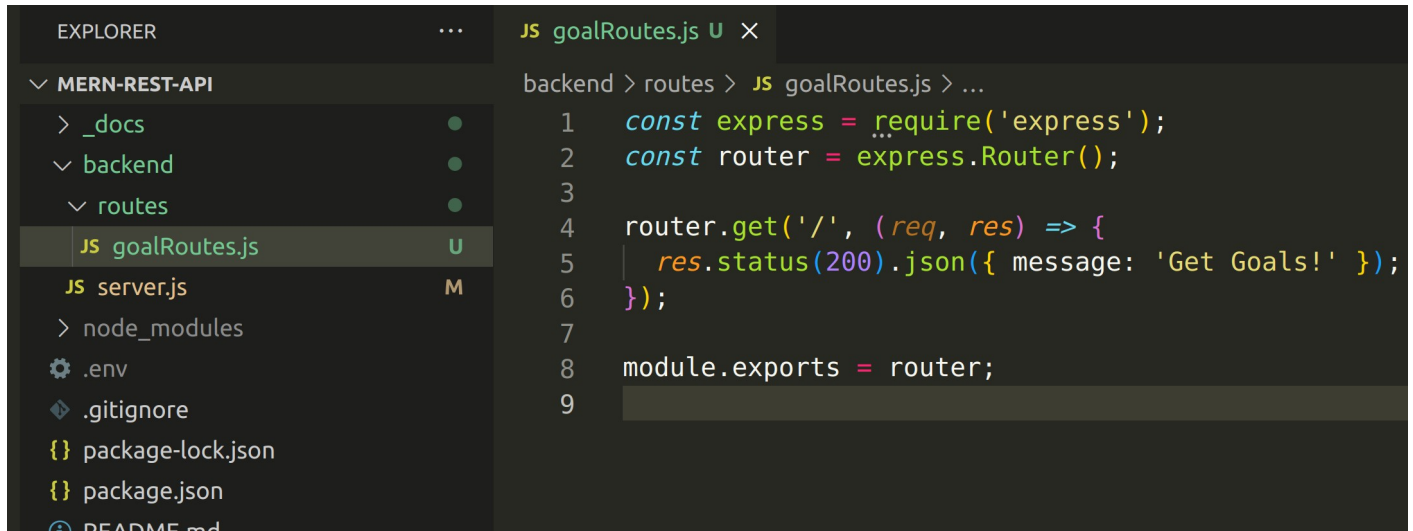
```
JS server.js M X
backend > JS server.js > ...
1  const express = require('express');
2  const dotenv = require('dotenv').config();
3  const port = process.env.PORT || 5000;
4
5  const app = express();
6
7  app.get('/api/goals', (req, res) => {
8    res.status(200).json({ message: 'Get Goals!' });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

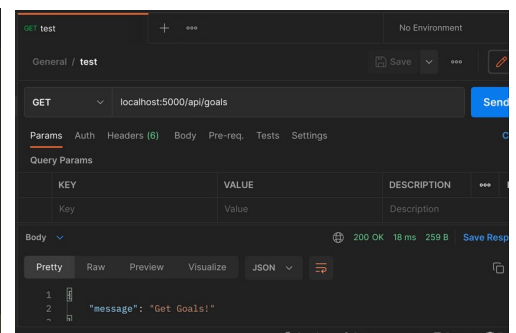


```
EXPLORER
MERN-REST-API
  _docs
  backend
    routes
      JS goalRoutes.js
      JS server.js
  node_modules
  .env
  .gitignore
  package-lock.json
  package.json
  README.md

JS 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
```



```
JS server.js
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
```



```
test
GET localhost:5000/api/goals
200 OK 18 ms 259 B
{"message": "Get Goals!"}
```

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 };
```

1.11 Work with data from request

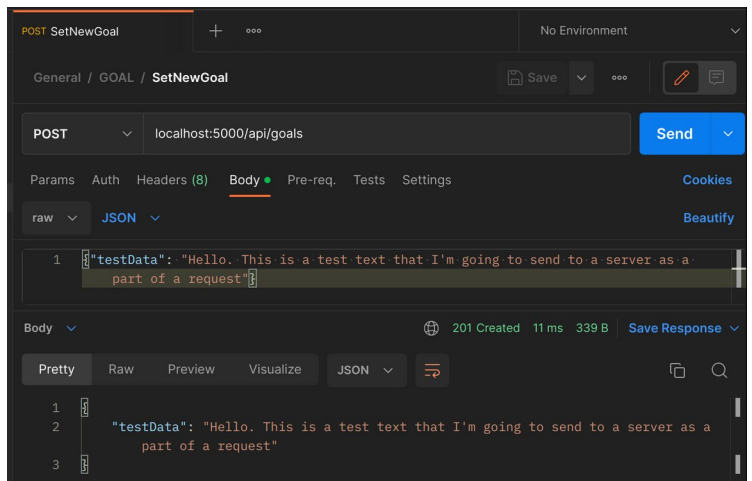
Add these 2 lines to be able to work with data in requests

```
app.use(express.json());
```

```
app.use(express.urlencoded({ extended: false }));
```

```
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(express.json());
8  app.use(express.urlencoded({ extended: false }));
9
10 app.use('/api/goals', require('./routes/goalRoutes'));
11
12 app.listen(port, () => console.log(`Server was started on port ${port}`));
13
```

```
// @desc    Create a new Goal
// @route    POST to /api/goals
// @access   Private
const setGoal = (req, res) => {
  ⚡ res.status(201).json(req.body);
};
```



1.12 Express error handler

Create a new folder backend/middleware to store all the middlewares. Create a new file backend/middleware/errorMiddleware.js

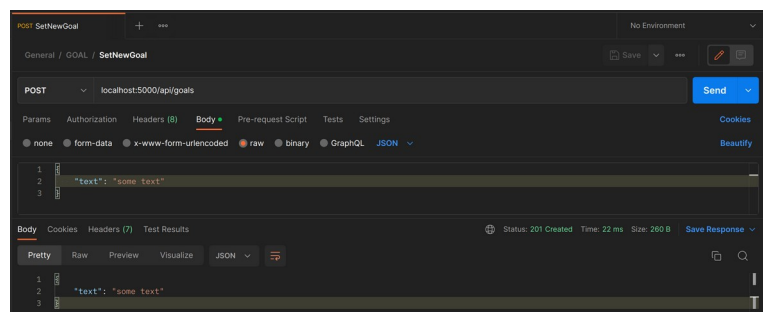
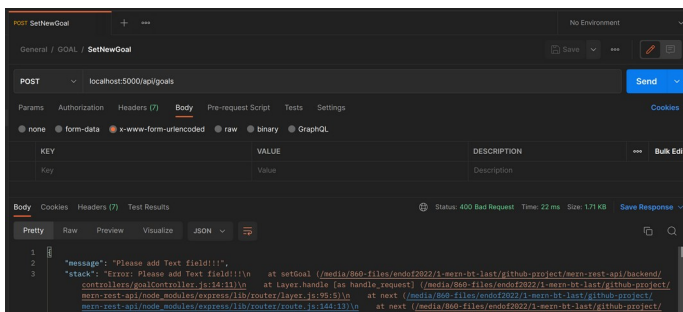
```
JS errorMiddleware.js U X
backend > middleware > JS errorMiddleware.js > ...
1  const errorHandler = (err, req, res, next) => {
2    const statusCode = res.statusCode ? res.statusCode : 500;
3    res.status(statusCode);
4    res.json({
5      message: err.message,
6      stack: process.env.NODE_ENV === 'production' ? null : err.stack,
7    });
8  };
9  module.exports = {
10   errorHandler,
11 };

```

Import your first middleware into server.js and apply it

```
JS server.js M X
backend > JS server.js > ...
1  const express = require('express');
2  const { errorHandler } = require('./middleware/errorMiddleware');
3  const dotenv = require('dotenv').config();
4  const port = process.env.PORT || 5000;
5
6  const app = express();
7
8  app.use(express.json());
9  app.use(express.urlencoded({ extended: false }));
10
11 app.use('/api/goals', require('./routes/goalRoutes'));
12
13 app.use(errorHandler);
14
15 app.listen(port, () => console.log(`Server was started on port ${port}`));

```



1.13 Using express-async-handler

Install express-async-handler to be able to work with async/await requests (for mongoose/mongoDB) and try/catch blocks

CLI=> *npm i express-async-handler*

Import in into controller and change each function

```
JS goalController.js M X
backend > controllers > JS goalController.js > ...
1 | const asyncHandler = require('express-async-handler');
2 |
3 > // @desc    Get all the Goals ...
6 | const getGoals = asyncHandler(async (req, res) => {
7 |   res.status(200).json({ message: 'Get all the Goals!' });
8 | });
9 |
10 > // @desc    Create a new Goal ...
13 | const setGoal = asyncHandler(async (req, res) => {
14 |   if (!req.body.text) {
15 |     res.status(400);
16 |     throw new Error('Please add Text field!!!');
17 |   }
18 |   res.status(201).json(req.body);
19 | });
20 |
21 > // @desc    Update a Goal by ID ...
24 | const updateGoal = asyncHandler(async (req, res) => {
25 |   res
26 |     .status(200)
27 |     .json({ message: `Update a Goal with ID = '${req.params.id}'` });
28 | });
29 |
30 > // @desc    Delete a Goal by ID ...
33 | const deleteGoal = asyncHandler(async (req, res) => {
34 |   res
35 |     .status(200)
36 |     .json({ message: `Delete a Goal with ID = '${req.params.id}'` });
37 | });
38 |
39 | module.exports = { getGoals, setGoal, updateGoal, deleteGoal };
40 |
```

