Learn The MERN Stack

youtube playlist

Table of Contents

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)

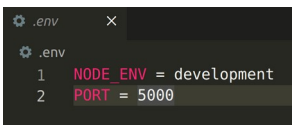
```
"scripts": {
  "start": "node backend/server.js",
  "server": "nodemon backend/server.js"
 },
```

Now you can run your server using next command:

CLI=> ***npm run server***

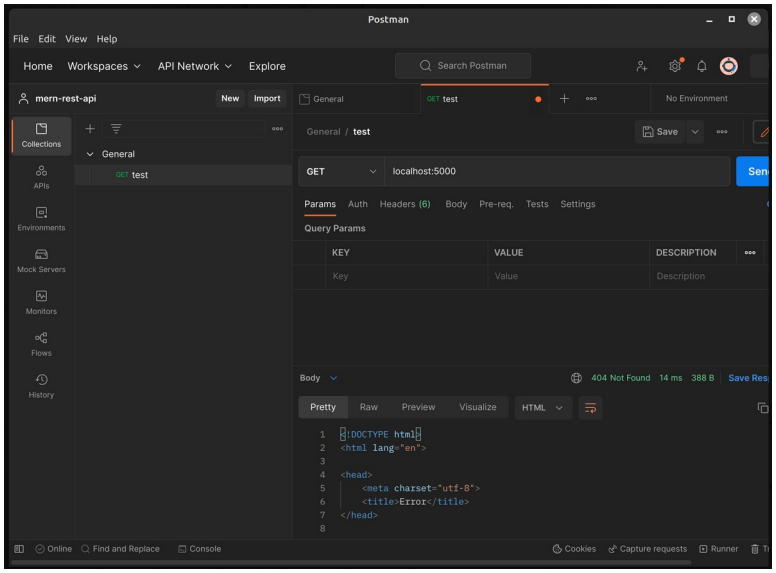
1.4 Create a simple express app

```
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 starterd on port ${port}`));
```



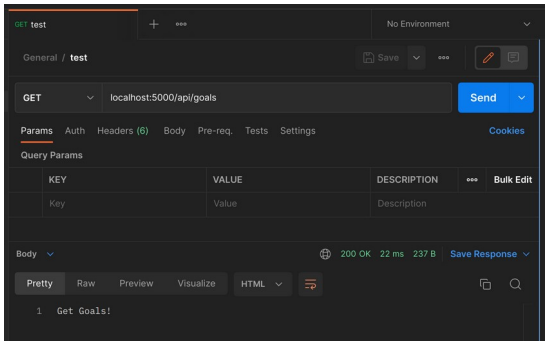
1.5 Postman

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



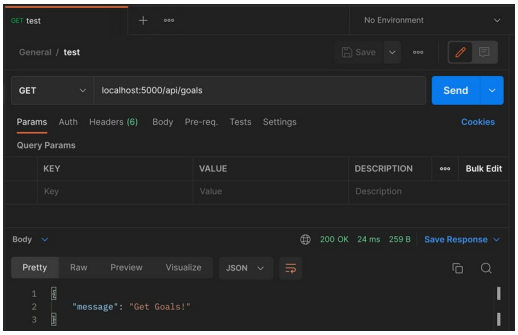
1.6 Create a simple test route with text response

```
JS server.js M
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 starterd on port ${port}`));
```



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

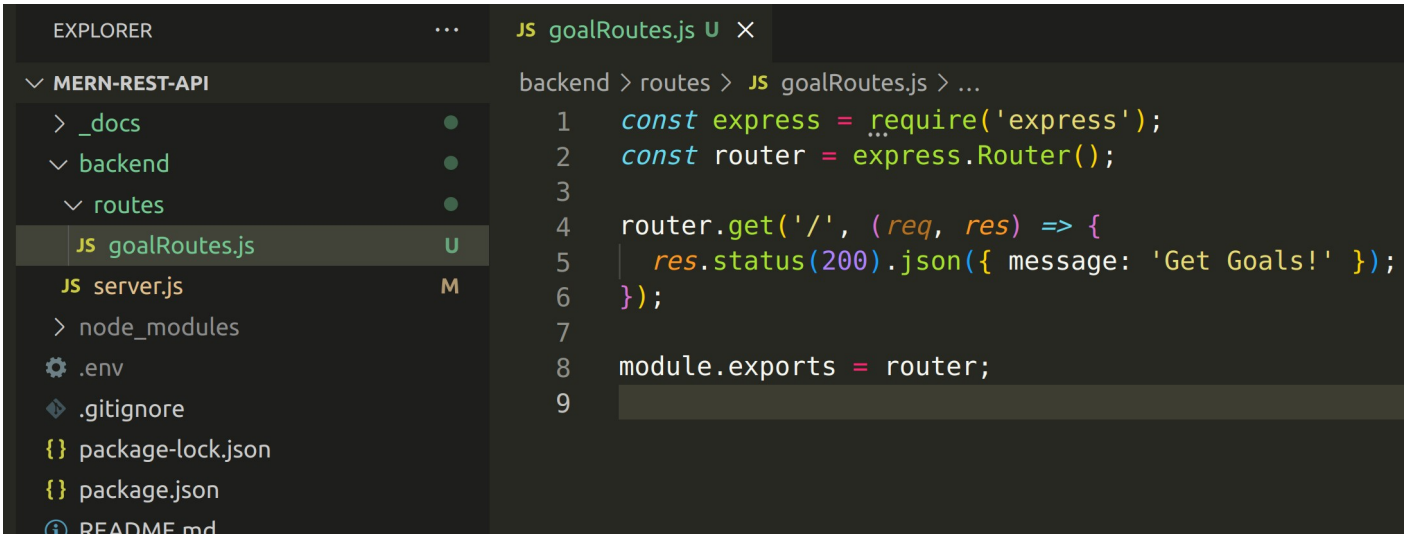
```
JS server.js M
backend > JS server.js > ...
1  const express = require('express');
2  const dotenv = require('dotenv').config();
3  const port = process.env.PORT || 5000;
4
5  const app = express();
6
7  app.get('/api/goals', (req, res) => {
8    res.status(200).json({ message: 'Get Goals!' });
9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

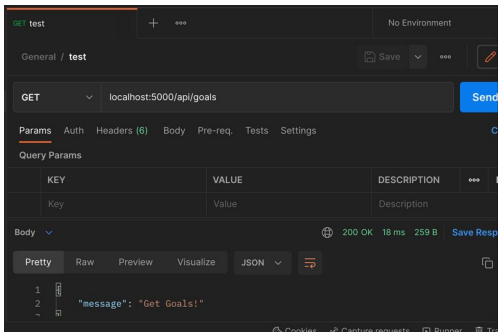


1.8 Split out all the routes into different files (entities)

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.





1.9 Add different routes for Goal entity

JS goalRoutes.js U

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

server.js M

backend > server.js > ...

```
const express = require('express');
const dotenv = require('dotenv').config();
const port = process.env.PORT || 5000;

const app = express();

app.use('/api/goals', require('./routes/goalRoutes'));

app.listen(port, () => console.log(`Server was starterd on port ${port}`));
```

goalRoutes.js U

backend > routes > goalRoutes.js > ...

```
const express = require('express');
const router = express.Router();
const {
  getGoals,
  setGoal,
  updateGoal,
  deleteGoal,
} = require('../controllers/goalController');

router.route('/').get(getGoals).post(setGoal);
router.route('/:id').put(updateGoal).delete(deleteGoal);

module.exports = router;
```

goalController.js U

backend > controllers > goalController.js > ...

```
// @desc    Get all the Goals
// @route   GET to /api/goals
// @access  Private
const getGoals = (req, res) => {
  res.status(200).json({ message: 'Get all the Goals!' });
};

// @desc    Create a new Goal
// @route   POST to /api/goals
// @access  Private
const setGoal = (req, res) => {
  res.status(201).json({ message: 'Set a new Goal!' });
};

// @desc    Update a Goal by ID
// @route   PUT to /api/goals/:id
// @access  Private
const updateGoal = (req, res) => {
  res
    .status(200)
    .json({ message: `Update a Goal with ID = '${req.params.id}'` });
};

// @desc    Delete a Goal by ID
// @route   DELETE to /api/goals/:id
// @access  Private
const deleteGoal = (req, res) => {
  res
    .status(200)
    .json({ message: `Delete a Goal with ID = '${req.params.id}'` });
};

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

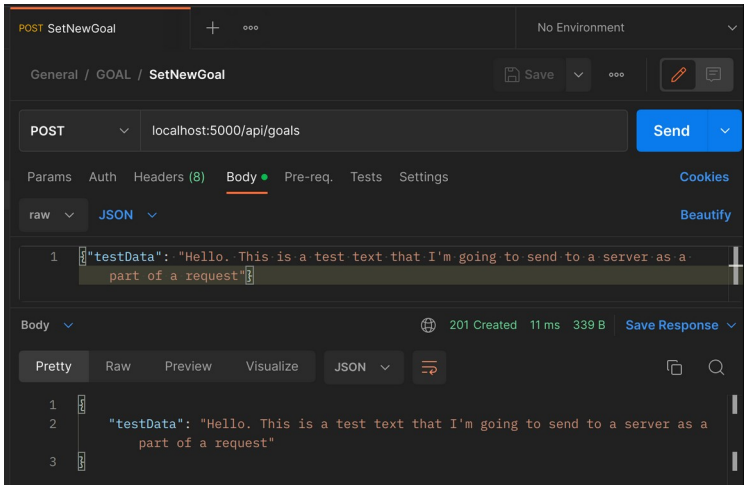
```
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
```

server.js M

backend > 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 starterd 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);
};
```



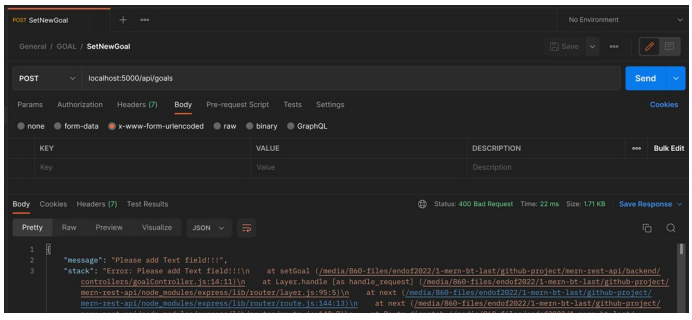
1.12 Express error handler

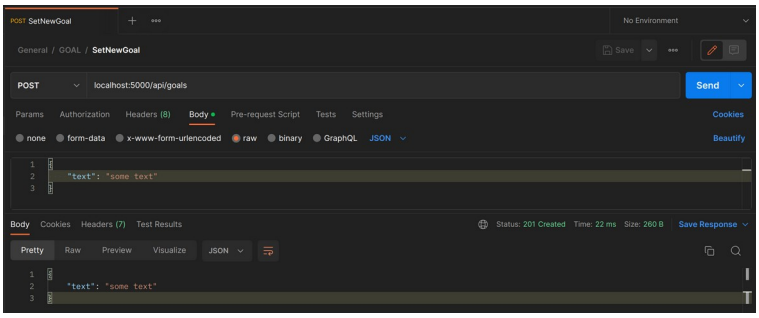
Create a new folder backend/middleware to store all the middlewares. Create a new file backend/middleware/errorMiddleware.js

```
JS errorMiddleware.js U
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
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 starterd on port ${port}`));
```





1.13 Using express-async-handler

Install express-async-handler to be able to work witn 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

backend > controllers > JS goalController.js > ...

```
const asynkHandler = require('express-async-handler');

// @desc    Get all the Goals ...
const getGoals = asynkHandler(async (req, res) => {
  res.status(200).json({ message: 'Get all the Goals!' });
});

// @desc    Create a new Goal ...
const setGoal = asynkHandler(async (req, res) => {
  if (!req.body.text) {
    res.status(400);
    throw new Error('Please add Text field!!!');
  }
  res.status(201).json(req.body);
});

// @desc    Update a Goal by ID ...
const updateGoal = asynkHandler(async (req, res) => {
  res
    .status(200)
    .json({ message: `Update a Goal with ID = '${req.params.id}'` });
});

// @desc    Delete a Goal by ID ...
const deleteGoal = asynkHandler(async (req, res) => {
  res
    .status(200)
    .json({ message: `Delete a Goal with ID = '${req.params.id}'` });
});

module.exports = { getGoals, setGoal, updateGoal, deleteGoal };
```