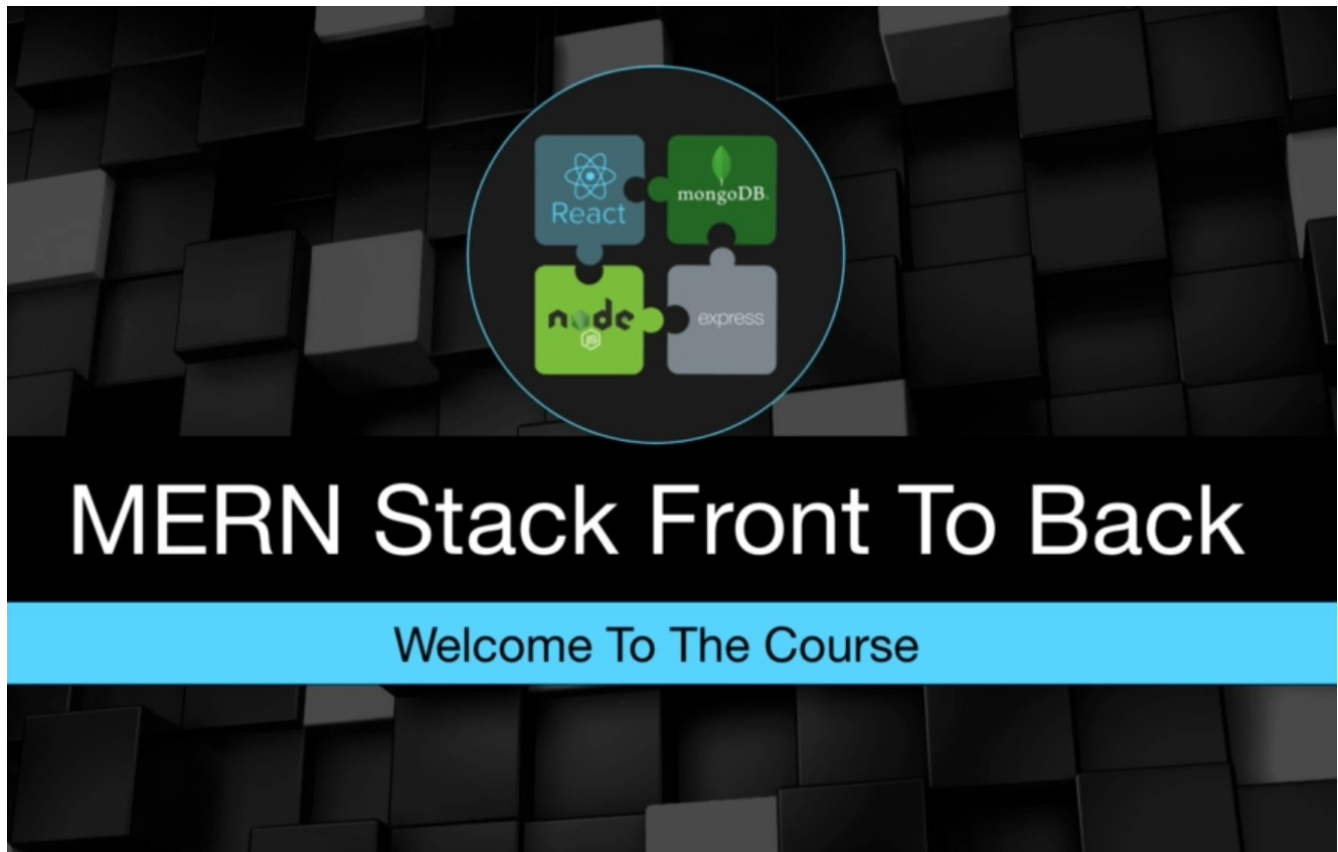


MERN Stack Front To Back: Full Stack React, Redux & Node.js (2020)



<https://www.udemy.com/course/mern-stack-front-to-back/>

by [Brad Traversy](#)

Table of Contents

0. Introduction.....	3
0.1 Environment & Setup.....	3
0.1.1 Node.js.....	3
0.1.2 Visual Studio Code (VSC).....	3
0.1.3 GIT.....	3
0.1.4 Postman.....	4
0.1.5 React Developer Tools chrome extension.....	4
0.1.6 Redux DevTools chrome extension.....	4
0.1.7 VSC extensions.....	4
1. Express & MongoDB Setup.....	5
1.1 MongoDB (create a new cluster for project).....	5
1.1 Express (install a package and setup a server).....	6
1.2 Create a connection with mongoDB.....	9
1.3 Create routes.....	11
1.4 User API Routes & JWT Authentication.....	13
1.4.1 Create an User schema for MongoDB.....	13
1.4.2 Edit test route to <i>api</i> /user endpoint.....	13
1.4.3 Add data validation with express-validate npm module.....	14
1.4.4 Register a new user.....	14
1.4.5 JWT.....	15
1.4.6 Build a custom middleware to work with protected routes.....	16
1.4.7 User authentication.....	16
1.5 Profile API Routes.....	17
1.5.1 Create a Profile Schema.....	17
What was used in project?.....	19

0. Introduction



Modern Technologies Used

- VSCode Editor
- ES6+ Syntax
- Async / Await
- React Hooks
- Redux With DevTools
- JWT (JSON Web Tokens)
- Postman HTTP Client
- Mongoose / MongoDB / Atlas
- Bcrypt Password Hashing
- Heroku & Git Deployment

0.1 Environment & Setup

0.1.1 Node.js

Windows: <https://nodejs.org/en/> - just download and install with GUI.

Linux:

```
CLI=> sudo apt-get update
```

```
CLI=> sudo apt install curl build-essential
```

```
CLI=> curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
```

```
CLI=> sudo apt install -y nodejs
```

```
CLI=> node -v [=> v14.16.1]
```

0.1.2 Visual Studio Code (VSC)

Windows: <https://code.visualstudio.com/> - just download and install with GUI.

Linux:

```
CLI=> sudo apt-get update
```

```
CLI=> sudo apt install code
```

0.1.3 GIT

Windows: <https://git-scm.com/> just download and install with GUI.

Linux: **GIT is a basic component**

0.1.4 Postman

Windows: <https://www.postman.com/> - just download and install with GUI.

Linux: <https://www.postman.com/downloads/> - download archive *tar.gz*
<https://dl.pstmn.io/download/latest/linux64>

go to download folder and unpack that file with command *tar xvf [PACKAGENAME].tar.gz*

CLI=> *tar xvf Postman-linux-x64-8.6.1.tar.gz*

go to folder and use shortcut

0.1.5 React Developer Tools chrome extension

0.1.6 Redux DevTools chrome extension

0.1.7 VSC extensions

- Bracket Pair Colorizer - <https://marketplace.visualstudio.com/items?itemName=CoenraadS.bracket-pair-colorizer>
- ES7 React/Redux/GraphQL/React-Native snippets - <https://marketplace.visualstudio.com/items?itemName=dsznajder.es7-react-js-snippets>
- Prettier - Code formatter - <https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode>

VSC=>Manage=>Settings=> “format on save” should be enable

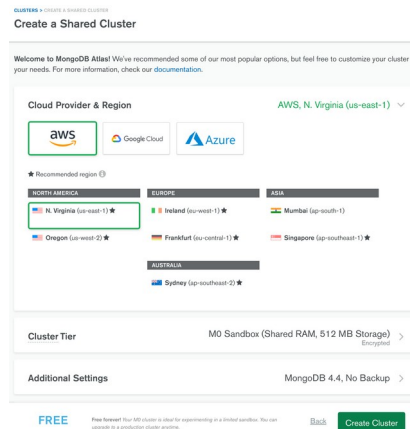
1. Express & MongoDB Setup

1.1 MongoDB (create a new cluster for project)

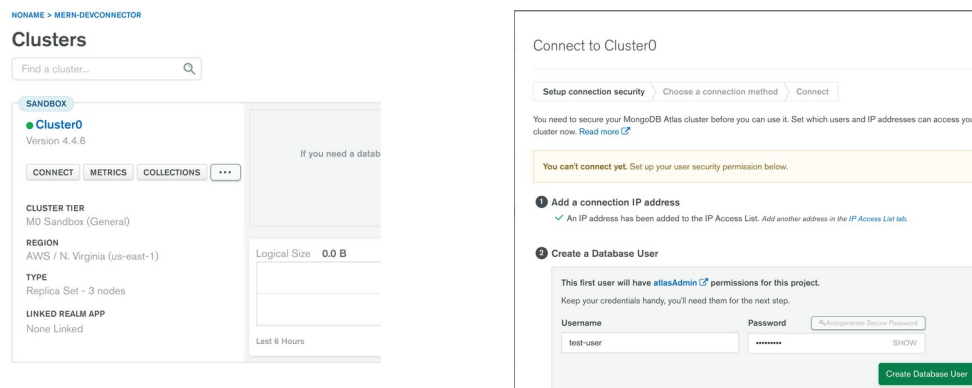
Create an account at <https://www.mongodb.com/>. Sign in to an account.

Create a new project – **MERN-devconnector**

Create a new cluster - **Cluster0**

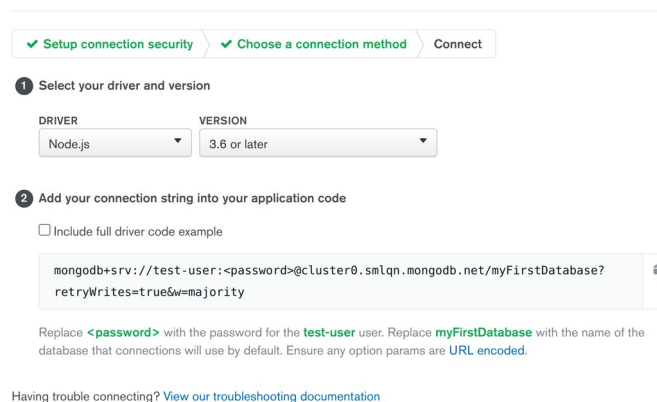


Push button “CONNECT” and add your IP adress and create a new database user to connect to a DB.



Push button “Choose a connection method” and “connect your application” and save that link

Connect to Cluster0



1.1 Express (install a package and setup a server)

1.1.1 Create a .gitignore file

It is a file with list of files/folders which will be ignored by GIT. Add **node_modules** folder

1.1.2 Initialize an NPM project and setup entry point

CLI=> *npm init -y*

```
stslon@stslon-System-Product-Name:/media/stslon/860_2/june2021/git-projects/mern2$ npm init -y
Wrote to /media/stslon/860_2/june2021/git-projects/mern2/package.json:

{
  "name": "mern2",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/ILopatenko/mern2.git"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/ILopatenko/mern2/issues"
  },
  "homepage": "https://github.com/ILopatenko/mern2#readme"
}
```

This command creates a new file package.json with basic information about project and all the dependencies. I'm going to change entry point ("main": "server.js") for this project to **server.js**

1.1.3 Install all the packages as regular dependencies

I'm going to use in this project next packages:

- **express** – backend server;
- **express-validator** – for validation data;
- **bcryptjs** – for making hash for passwords;
- **config** – for making global variables;
- **gravatar** – for working with user's avatars;
- **jsonwebtoken** – for working with JWT;
- **mongoose** – for working with MongoDB database;
- **request** – for working with another API based services;

CLI=> *npm install express express-validator bcryptjs config gravatar jsonwebtoken mongoose request*

1.1.4 Install all the packages as DEV dependencies

DEV dependencies will be installed without direct reference to a project (npm will add them to package.json file as a devDependencies)

- **nodemon** – will track all the changes and make a restart server automatically;
- **concurrently** – allows me to run few scripts at the same time with a single command;

CLI=> `npm install -D nodemon concurrently`

```
{ } package.json U X
{ } package.json > ...
1  {
2    "name": "mern2",
3    "version": "1.0.0",
4    "description": "",
5    "main": "server.js",
6    "scripts": {
7      "test": "echo \\\"Error: no test specified\\\" && exit 1"
8    },
9    "repository": {
10     "type": "git",
11     "url": "git+https://github.com/ILopatenko/mern2.git"
12   },
13   "keywords": [],
14   "author": "",
15   "license": "ISC",
16   "bugs": {
17     "url": "https://github.com/ILopatenko/mern2/issues"
18   },
19   "homepage": "https://github.com/ILopatenko/mern2#readme",
20   "dependencies": {
21     "bcryptjs": "^2.4.3",
22     "config": "^3.3.6",
23     "express": "^4.17.1",
24     "gravatar": "^1.8.1",
25     "jsonwebtoken": "^8.5.1",
26     "mongoose": "^5.12.13",
27     "request": "^2.88.2"
28   },
29   "devDependencies": {
30     "concurrently": "^6.2.0",
31     "nodemon": "^2.0.7"
32   }
33 }
```

1.1.5 Create an express server

```
JS server.js U X
_docs > JS server.js > ...
1  const express = require('express');
2
3  const app = express();
4
5  app.get('/', (req, res) => res.send('API is running'));
6
7  const PORT = process.env.PORT || 5000;
8
9  app.listen(PORT, () => console.log(`Server is started on port ${PORT}`));
10
```

1.1.6 Change run script at package.json

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

Now command “npm run start” will run server.js with node.js and command “npm run server” will run server.js with nodemon

1.1.7 The first run

The image displays three screenshots illustrating the initial setup and execution of a web server.

Terminal Screenshot: Shows the command prompt for a user named 'stslon' in a directory. The command `npm run server` is executed. The output shows that nodemon version 2.0.7 is installed and is watching for file changes. The server is started on port 5000.

Browser Screenshot: A web browser window showing the address `localhost:5000`. The page content displays the message "API is running".

Postman Screenshot: A screenshot of the Postman application. A GET request is configured to `localhost:5000`. The response is shown in the 'Body' tab, displaying the message "API is running". The status is 200 OK, and the response time is 12 ms.

1.2 Create a connection with mongoDB.

Create a new folder **config**. Inside this folder create a new file – **default.json** – with information to connect with a database

```
{ default.json U X
config > {} default.json > ...
1  {}
2  "mongoURI": "mongodbsrv://test-user:test-user@cluster0.smlqn.mongodb.net/myFirstDatabase?retryWrites=true&w=majority"
3  }
```

Create a new file – **db.js** – with connection logic

```
JS db.js U
config > JS db.js > ...
1  const mongoose = require('mongoose');
2
3  const config = require('config');
4
5  const db = config.get('mongoURI');
6
7  const connectDB = async () => {
8    try {
9      await mongoose.connect(db);
10     console.log('MongoDB is connected!');
11   } catch (error) {
12     console.error(err.message);
13     process.exit(1);
14   }
15 };
16
17 module.exports = connectDB;
```

Add **db.js** to a **server.js**

```
JS server.js M X
JS server.js > ...
1  const express = require('express');
2  const connectToDB = require('./config/db');
3
4  const app = express();
5  connectToDB();
6
7  app.get('/', (req, res) => res.send('API is running'));
8
9  const PORT = process.env.PORT || 5000;
10
11 app.listen(PORT, () => console.log(`Server is started on port ${PORT}`));
12
```

```
stslon@stslon-System-Product-Name:/media/stslon/860_2/june2021/git-projects/mern2$ npm run server
> mern2@1.0.0 server /media/stslon/860_2/june2021/git-projects/mern2
> nodemon server.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
(node:49585) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version.
(Use `node --trace-deprecation ...` to show where the warning was created)
Server is started on port 5000
(node:49585) [MONGODB DRIVER] Warning: Top-level use of w, wtimeout, j, and fsync is deprecated. Use writeConcern instead.
(node:49585) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new MongoClient, pass option { useUnifiedTopology: true } to the MongoClient constructor.
MongoDB is connected!
```

Fix all the warnings and run a server again:

```
JS db.js U X
config > JS db.js > ...
1  const mongoose = require('mongoose');
2
3  const config = require('config');
4
5  const db = config.get('mongoURI');
6
7  const connectDB = async () => {
8    try {
9      await mongoose.connect(db, {
10        useNewUrlParser: true,
11        useUnifiedTopology: true,
12      });
13      console.log('MongoDB is connected!');
14    } catch (error) {
15      console.error(err.message);
16      process.exit(1);
17    }
18  };
19
20  module.exports = connectDB;
21
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
stslon@stslon-System-Product-Name:/media/stslon/860_2/june2021/git-projects/mern2$ npm run server
> mern2@1.0.0 server /media/stslon/860_2/june2021/git-projects/mern2
> nodemon server.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
Server is started on port 5000
MongoDB is connected!
```

1.3 Create routes.

Create a new folder – **routes/api** – to store and work with all the routes.

Inside this folder create new files – **user.js**, **profile.js**, **post.js** and **auth.js**

```
JS user.js U X
routes > api > JS user.js > ...
1  const express = require('express');
2  const router = express.Router();
3
4  //@route      GET api/user
5  //@desc      Test route
6  //@access     Public
7  router.get('/', (req, res) => res.send('User route'));
8
9  module.exports = router;
```

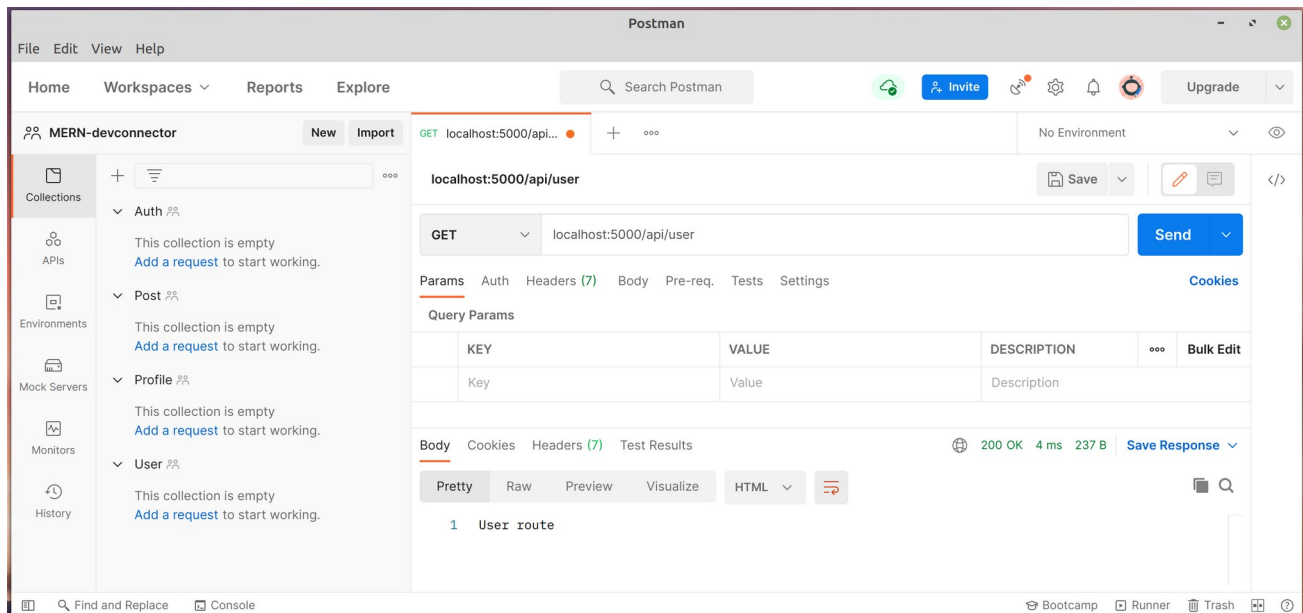
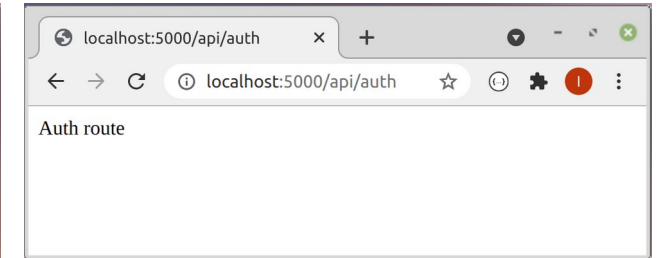
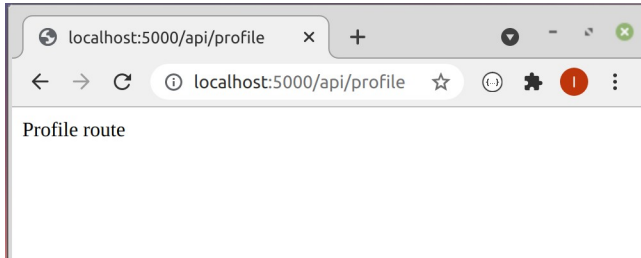
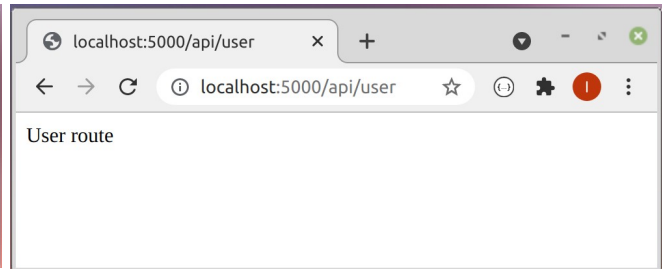
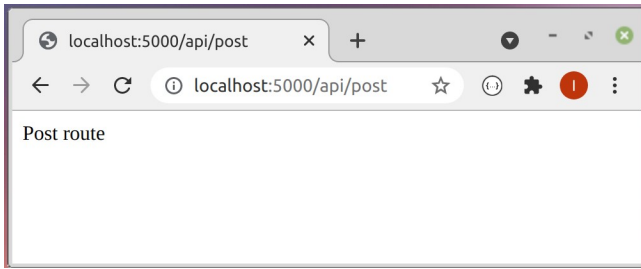
```
JS profile.js U X
routes > api > JS profile.js > router.get('/') callback
1  const express = require('express');
2  const router = express.Router();
3
4  //@route      GET api/profile
5  //@desc      Test route
6  //@access     Public
7  router.get('/', [req, res] => res.send('Profile route'));
8
9  module.exports = router;
```

```
JS post.js U X
routes > api > JS post.js > ...
1  const express = require('express');
2  const router = express.Router();
3
4  //@route      GET api/post
5  //@desc      Test route
6  //@access     Public
7  router.get('/', (req, res) => res.send('Post route'));
8
9  module.exports = router;
```

```
JS auth.js U X
routes > api > JS auth.js > ...
1  const express = require('express');
2  const router = express.Router();
3
4  //@route      GET api/auth
5  //@desc      Test route
6  //@access     Public
7  router.get('/', (req, res) => res.send('Auth route'));
8
9  module.exports = router;
```

Add all these routes to **server.js**

```
JS server.js M X
JS server.js > ...
1  const express = require('express');
2
3  const connectToDB = require('./config/db');
4
5  const userRoute = require('./routes/api/user');
6  const profileRoute = require('./routes/api/profile');
7  const postRoute = require('./routes/api/post');
8  const authRoute = require('./routes/api/auth');
9
10 const app = express();
11 connectToDB();
12
13 app.get('/', (req, res) => res.send('API is running'));
14
15 //define all the routes
16 app.use('/api/user', userRoute);
17 app.use('/api/profile', profileRoute);
18 app.use('/api/post', postRoute);
19 app.use('/api/auth', authRoute);
20
21 const PORT = process.env.PORT || 5000;
22
23 app.listen(PORT, () => console.log(`Server is started on port ${PORT}`));
24
```



1.4 User API Routes & JWT Authentication

1.4.1 Create an User schema for MongoDB

Create a new folder – **models** – to store all the schemas.

Create a new file – **User.js** – inside this folder.

```
JS User.js U X
models > JS User.js > UserSchema > password
1 const mongoose = require('mongoose');
2
3 const UserSchema = new mongoose.Schema({
4   name: {
5     type: String,
6     required: true,
7   },
8   email: {
9     type: String,
10    required: true,
11    unique: true,
12  },
13  password: {
14    type: String,
15    required: true,
16  },
17  avatar: {
18    type: String,
19  },
20  date: {
21    type: Date,
22    default: Date.now,
23  },
24 });
25
26 module.exports = User = mongoose.model('user', UserSchema);
27
```

1.4.2 Edit test route to *api/user* endpoint

Add Middleware (ex. bodyParser) to work with request object. Change user.js route. Change request in Postman

```
JS server.js M X
JS server.js > ...
1 const express = require('express');
2
3 const connectToDB = require('./config/db');
4
5 const userRoute = require('./routes/api/user');
6 const profileRoute = require('./routes/api/profile');
7 const postRoute = require('./routes/api/post');
8 const authRoute = require('./routes/api/auth');
9
10 const app = express();
11
12 //Init a built in Middleware (ex. bodyParser)
13 app.use(express.json({ extended: false }));
14
15 connectToDB();
16
17 app.get('/', (req, res) => res.send('API is running'));
18
19 //define all the routes
20 app.use('/api/user', userRoute);
21 app.use('/api/profile', profileRoute);
22 app.use('/api/post', postRoute);
23 app.use('/api/auth', authRoute);
24
25 const PORT = process.env.PORT || 5000;
26
27 app.listen(PORT, () => console.log('Server is started on port ${PORT}'));
28
```

```
JS user.js M X
routes > api > JS user.js > ...
1 const express = require('express');
2 const router = express.Router();
3
4 /* @route GET api/user
5 //@desc Test route
6 //@access Public
7 router.get('/', (req, res) => res.send('User route'));
8
9 //@route POST api/user
10 //@desc Register a new user
11 //@access Public
12 router.post('/', (req, res) => {
13   console.log(req.body);
14   res.send('User route');
15 });
16
17 module.exports = router;
18
```

localhost:5000/api/user

POST localhost:5000/api/user

Params Authorization Headers (9) Body Pre-request Script Te

Headers 8 hidden

KEY	VALUE
Content-Type	application/json
Key	Value

POST localhost:5000/api/user

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "name": "Iurii Lopatenko"
3 }
```

```
nodemon] 2.0.7
nodemon] to restart at any time, enter `rs`
nodemon] watching path(s): *.*
nodemon] watching extensions: js,mjs,json
nodemon] starting `node server.js`
Server is started on port 5000
MongoDB is connected!
{ name: 'Iurii Lopatenko' }
```

Now Postman can send a request (with some data in body) and server can receive this request and work with data (in this example server just sent an object from req.body to console.log)

1.4.3 Add data validation with express-validate npm module

```
1 const express = require('express');
2 const router = express.Router();
3 const { body, validationResult } = require('express-validator');
4
5 /* @@route GET api/user
6 //@desc Test route
7 //@access Public
8 router.get('/', (req, res) => res.send('User route')); */
9
10 @@route POST api/user
11 //@desc Register a new user
12 //@access Public
13 router.post(
14   '/',
15   body('name')
16     .isString()
17     .withMessage('Name should be a text')
18     .not()
19     .isEmpty()
20     .withMessage('Name can not be an empty'),
21   body('email')
22     .isEmail()
23     .withMessage('Email should an email')
24     .not()
25     .isEmpty()
26     .withMessage('Email can not be an empty'),
27   body('password')
28     .isLength({ min: 6 })
29     .withMessage('Password should be at least 6 characters length'),
30   (req, res) => {
31     const errors = validationResult(req);
32     if (!errors.isEmpty()) {
33       return res.status(400).json({ errors: errors.array() });
34     }
35     console.log(req.body);
36     res.send('User route');
37   }
38 );
39
40 module.exports = router;
```

POST localhost:5000/api/user

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "name": null,
3   "email": 34,
4   "password": "hello"
5 }
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "errors": [
3     {
4       "value": null,
5       "msg": "Name should be a text",
6       "param": "name",
7       "location": "body"
8     },
9     {
10      "value": null,
11      "msg": "Name can not be an empty",
12      "param": "name",
13      "location": "body"
14    },
15     {
16      "value": 34,
17      "msg": "Email should an email",
18      "param": "email",
19      "location": "body"
20    },
21     {
22      "value": "hello",
23      "msg": "Password should be at least 6 characters length",
24      "param": "password",
25      "location": "body"
26    }
27   ]
28 }
```

User / Register

POST localhost:5000/api/user

Params Auth Headers (10) Body Pre-req Tests Settings

raw JSON

```
1 {
2   "name": "Iurii Lopatenko",
3   "email": "YALopatenko@gmail.com",
4   "password": "hello3"
5 }
```

Body

Pretty Raw Preview Visualize HTML

1 User route

```
Server is started on port 5000
MongoDB is connected!

{
  name: 'Iurii Lopatenko',
  email: 'YALopatenko@gmail.com',
  password: 'hello3'
}
```

1.4.4 Register a new user.

```
1 const bcrypt = require('bcryptjs');
2 //@@route POST api/user
3 //@desc Register a new user
4 //@access Public
5 router.post('/',
6   body('name').isString().withMessage('Name should be a text').not().isEmpty().withMessage('Name can not be empty'),
7   body('email').isEmail().withMessage('Email should be an email').not().isEmpty().withMessage('Email can not be empty'),
8   body('password').isLength({ min: 6 }).withMessage('Password should be at least 6 characters length'),
9   async (req, res) => {
10     const errors = validationResult(req);
11     if (!errors.isEmpty()) {
12       //action in case when validation has detected issue
13       return res.status(400).json({ errors: errors.array() });
14     }
15     //Destructuring all the data from req.body to local variables
16     const { name, email, password } = req.body;
17     try {
18       //action in case when validation has not detected any issues:
19       //1 - See if user with current email exists in database
20       let user = await User.findOne({ email });
21       if (user) {
22         //if user with this email was found in database - throw an error
23         return res.status(400).json({ errors: [{ msg: "User already exists! Try to use another email!" }] });
24       }
25       //2 - Get a gravatar for this email
26       const avatar = gravatar.url(email, {s: '200',r: 'pg',d: 'mm'});
27       //3 Create a new user
28       user = new User({name, email, avatar, password});
29       //4 - Encrypt user's password
30       //4.1 - Create a salt
31       const salt = await bcrypt.genSalt(10);
32       //4.2 - Hash the password
33       user.password = await bcrypt.hash(password, salt);
34       //5 - Save a new user model to a database
35       await user.save();
36       //TODO !!!!!
37       //6 - return a JWT
38       //TODO !!!!!
39       res.send('User registered!');
40     } catch (error) {
41       console.error(err.message);
42       res.status(500).send('Server error!');
43     }
44   }
45 );
46
47 module.exports = router;
```

POST localhost:5000/api/user

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "name": "Iurii Lopatenko",
3   "email": "YALopatenko@gmail.com",
4   "password": "qwerty"
5 }
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize HTML

1 User registered!

myFirstDatabase.users

COLLECTION SIZE: 266B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 8KB

Find Indexes Schema Anti-Patterns Aggregation

FILTER {"filter": "example"}

QUERY RESULTS 1-1 OF 1

```
1 {
2   "_id": ObjectId("60c3a530609f6f665b1de62b"),
3   "name": "Iurii Lopatenko",
4   "email": "YALopatenko@gmail.com",
5   "avatar": "http://www.gravatar.com/avatar/29b33c4fbaf493f1e63469325fa5900?s=200&r=pg&d=mm",
6   "password": "$2a$10$KGLiCmH2tKbONG.kW494e98f02f4IP01stB2CD20Tw4EKHjCbway",
7   "date": "2021-06-11T18:02:24.981+00:00",
8   "__v": 0
9 }
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "errors": [
3     {
4       "msg": "User already exists! Try to use another email!"
5     }
6   ]
7 }
```

1.4.5 JWT

Encoded

PASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c

Decoded

EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
)
```

☐ secret base64 encoded

```
//6 - return a JWT
//6.1 - create a payload
const payload = {
  user: {
    id: user.id,
  },
};
//6.2 - create a JWT with payload and secret and a callback function
jwt.sign(
  payload,
  config.get('jwtSecret'),
  { expiresIn: 360000 },
  (err, token) => {
    if (err) throw err;
    res.json({ token });
  });
```

() default.json M X

config > {} default.json > ...

1 {

2 {

3 "mongoURI": "mongodb+srv://test-user:test-user@cluster0.smlqn.mongodb.net/myFirstDatabase?retryWrites=true&w=majority",

4 "jwtSecret": "superSecret"

}

User / Register

Save

POST

localhost:5000/api/user

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

Body

Cookies

Headers (7)

Test Results

Status: 200 OK

Time: 245 ms

Size: 431 B

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

4

1.4.6 Build a custom middleware to work with protected routes

```
JS authjs U X
middleware > JS authjs > ...
1 const jwt = require('jsonwebtoken');
2 const config = require('config');
3
4 module.exports = function (req, res, next) {
5   //Get a JWT
6   const token = req.header('x-auth-token');
7   //IF JWT is not exists - throw an error
8   if (!token) {
9     return res.status(401).json({ msg: 'No token, auth denied!' });
10  }
11  //Check a JWT
12  try {
13    //Create a decoded variable with payload (userId, and information about JWT life time) from decoded JWT
14    const decoded = jwt.verify(token, config.get('jwtSecret'));
15    //Create a new parameter in request with userID - { id: '60c3bdba5debe09dde3855a7' }
16    req.user = decoded.user;
17    next();
18  } catch (error) {
19    res.status(401).json({ msg: 'Token is not valid' });
20  }
21 };
22
```

```
JS authjs M X
routes > api > JS authjs > ...
1 const express = require('express');
2 const router = express.Router();
3 const auth = require('../middleware/auth');
4 const User = require('../models/User');
5
6 //@route GET api/auth
7 //@desc Test route
8 //@access Public
9 router.get('/', auth, async (req, res) => {
10   try {
11     const user = await User.findById(req.user.id).select('-password');
12     res.json(user);
13   } catch (error) {
14     console.error(err.message);
15     res.status(500).send('Server error');
16   }
17 });
18
19 module.exports = router;
20
```

GET localhost:5000/api/auth

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Headers < 7 hidden

KEY	VALUE	DESCRIPTION
x-auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2Vybm9keS5kaW50f54d8fc69f53c98c4fd6e647995?s=2006r=pg5d=mm	

Body Cookies Headers (7) Test Results Status: 200 OK Time: 93 ms

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": "60c3b7a68c9b3891ceccbe55",
3   "name": "Test2 User",
4   "email": "test2@test.test",
5   "avatar": "https://www.gravatar.com/avatar/8a50f54d8fc69f53c98c4fd6e647995?s=2006r=pg5d=mm",
6   "date": "2021-06-11T19:21:10.466Z",
7   "_v": 0
8 }
```

GET localhost:5000/api/auth

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Headers < 7 hidden

KEY	VALUE	DESCRIPTION
x-auth-token	wrongToken	

Body Cookies Headers (7) Test Results Status: 401 Unauthorized Time: 12 ms

Pretty Raw Preview Visualize JSON

```
1 {
2   "msg": "Token is not valid"
3 }
```

1.4.7 User authentication

```
JS authjs M
routes > api > JS authjs > router.post('/') callback
21
22 //@route POST api/auth
23 //@desc Authenticate user & get a JWT
24 //@access Public
25 router.post('/', [
26   body('email').isEmail().withMessage('Email should be an email'),
27   body('password').exists().withMessage('Password is required')
28 ], async (req, res) => {
29   const errors = validationResult(req);
30   if (!errors.isEmpty()) {
31     //action in case when validation has detected issue
32     return res.status(400).json({ errors: errors.array() });
33   }
34   //Destructuring all the data from req.body to local variables
35   const { email, password } = req.body;
36   try {
37     //action in case when validation has not found a user:
38     //1 - See if user with current email exists in database
39     let user = await User.findOne({ email });
40     if (!user) {
41       //if user with this email was not found in database - throw an error
42       return res.status(400).json({
43         errors: [{ msg: 'Invalid credentials' }],
44       });
45     }
46     //2 - checking the password
47     const isMatch = await bcrypt.compare(password, user.password);
48     if (!isMatch) {
49       //if password is wrong - throw an error
50       return res.status(400).json({
51         errors: [{ msg: 'Invalid credentials' }],
52       });
53     }
54     //3 - return a JWT
55     //3.1 - create a payload
56     const payload = {
57       user: {
58         id: user.id,
59       },
60     };
61     //3.2 - create a JWT with payload and secret and a callback function
62     jwt.sign(
63       payload,
64       config.get('jwtSecret'),
65       { expiresIn: 360000 },
66       (err, token) => {
67         if (err) throw err;
68         res.json({ token });
69       }
70     );
71   } catch (error) {
72     console.error(err.message);
73     res.status(500).send('Server error');
74   }
75 };
76
77 module.exports = router;
78
```

POST localhost:5000/api/auth

Params Authorization Headers (11) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "email": "test@test.test",
3   "password": "qerty"
4 }
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 152 ms Size: 431 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2Vybm9keS5kaW50f54d8fc69f53c98c4fd6e647995?s=2006r=pg5d=mm"
3 }
```


1.5 Profile API Routes

1.5.1 Create a Profile Schema

```
const mongoose = require('mongoose');
const ProfileSchema = new mongoose.Schema({
  user: {type: mongoose.Schema.Types.ObjectId, ref: 'user'},
  company: {type: String},
  website: {type: String},
  location: {type: String},
  status: {type: String, required: true},
  skills: {type: [String], required: true},
  bio: {type: String},

  githubusername: {type: String},

  experience: [{
    title: {type: String, required: true},
    company: {type: String, required: true},
    location: {type: String},
    from: {type: Date, required: true},
    to: {type: Date},
    current: {type: Boolean, required: false},
    description: {type: String},
  }],

  education: [{
    school: {type: String, required: true},
    degree: {type: String, required: true},
    fieldofstudy: {type: String, required: true},
    from: {type: Date, required: true},
    to: {type: Date},
    current: {type: Boolean, required: false},
    description: {type: String},
  }],

  social: {
    facebook: {type: String},
    linkedin: {type: String},
    instagramm: {type: String}
  },

  date: {type: Date, default: Date.now}
});

module.exports = Profile = mongoose.model('profile', ProfileSchema);
```


What was used in project?

Operating Systems: Linux Mint 20.1 and Windows 10

Node.js – javascript runtime enviroment.

NPM

GIT -

express

express-validator

MongoDB

Mongoose

jwt

nodemon

postman

VSCode

Chrome