

This is a placeholder for a title page

Table of Contents

- This is a placeholder for a title page.....1
- 1. Introduction.....3
 - 1.x VSCode.....3
 - 1.x.1 Prettier set up.....3
- 2. Express & MongoDB Setup.....4
 - 2.1. MongoDB Atlas Setup.....4
 - 2.1.1 Create an account at MongoDB.com.....4
 - 2.1.2 Log in at MongoDB.com.....4
 - 2.1.3 Create a new project at mongoDB.com.....4
 - 2.1.4 Create a new cluster.....4
 - 2.1.5 Create a new user.....4
 - 2.1.6 Change whitelist of IP addresses.....4
 - 2.2 Install Dependencies & Basic Express Setup.....4
 - 2.2.1 Create a file .gitignore. Add to that file folder node_modules/.....4
 - 2.2.2 Create GIT repository with [CLI=> git init].....4
 - 2.2.3 Create NPM with [CLI=> npm init].....4
 - 2.2.4 Install all the regular dependencies.....5
 - 2.2.5 Install all the developer dependencies.....5
 - 2.2.6 Create a main entry file – server.js.....5
 - 2.2.7 Change start scripts at package.json.....5
 - 2.2.8 Create a SERVER with simple test route.....5
 - 2.3 Connecting To MongoDB With Mongoose.....6
 - 2.3.1 Create a connection to database.....6
 - 2.3.2 Create a connection logic on server.....6
 - 2.4 Create route files with Express Router.....7
- 3. User API Routes & JWT Authentication.....8

1. Introduction.

There will be a general information about project, MERN stack and all side technologies.

1.x VSCode

1.x.1 Prettier set up

<https://glebbahmutov.com/blog/configure-prettier-in-vscode/#settings>

VSCode setup

To use the Prettier we have just installed from VSCode we need to install the [Prettier VSCode extension](#):

1. Launch VS Code Quick Open (Ctrl+P)
2. Run the following command

```
1 ext install esbenp.prettier-vscode
```

Because you might have global settings related to code formatting, I prefer having in each repository a file with local workspace VSCode settings. I commit this file `.vscode/settings.json` to source control to make sure everyone uses the same extension to format the code.

```
.vscode/settings.json
1 {
2   "editor.defaultFormatter": "esbenp.prettier-vscode",
3   "editor.formatOnSave": true
4 }
```

Now every time we save a JavaScript file, it will be formatted using Prettier automatically. Here is me formatting `projectA/index.js`

2. Express & MongoDB Setup

2.1. MongoDB Atlas Setup

2.1.1 Create an account at MongoDB.com

2.1.2 Log in at MongoDB.com

2.1.3 Create a new project at mongoDB.com

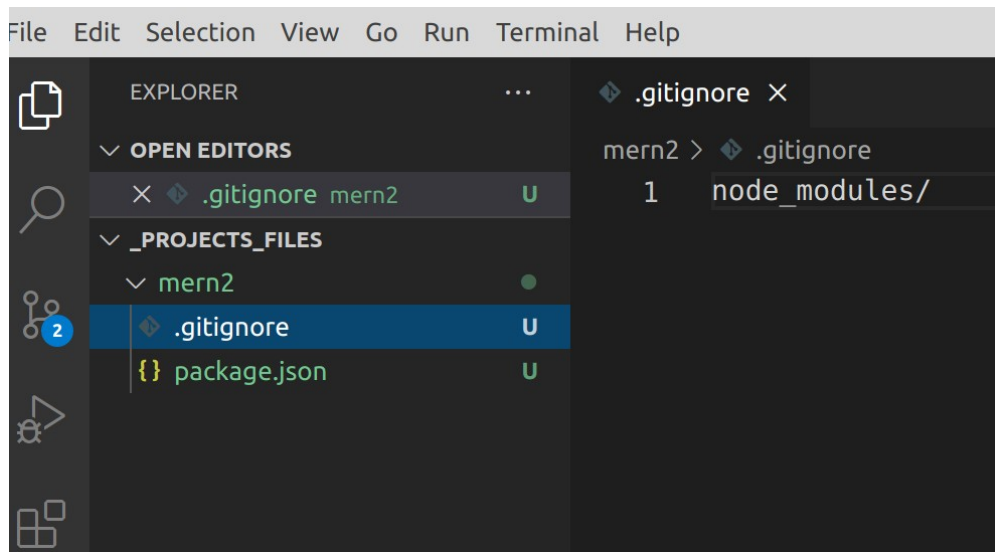
2.1.4 Create a new cluster

2.1.5 Create a new user

2.1.6 Change whitelist of IP addresses

2.2 Install Dependencies & Basic Express Setup

2.2.1 Create a file .gitignore. Add to that file folder node_modules/



2.2.2 Create GIT repository with [CLI=> git init]

2.2.3 Create NPM with [CLI=> npm init]

2.2.4 Install all the regular dependencies

[CLI=> npm i express express-validator bcryptjs config gravatar jsonwebtoken mongoose request]

2.2.5 Install all the developer dependencies

[CLI=> npm i -D nodemon concurrently]

2.2.6 Create a main entry file – server.js

[CLI=> touch server.js]

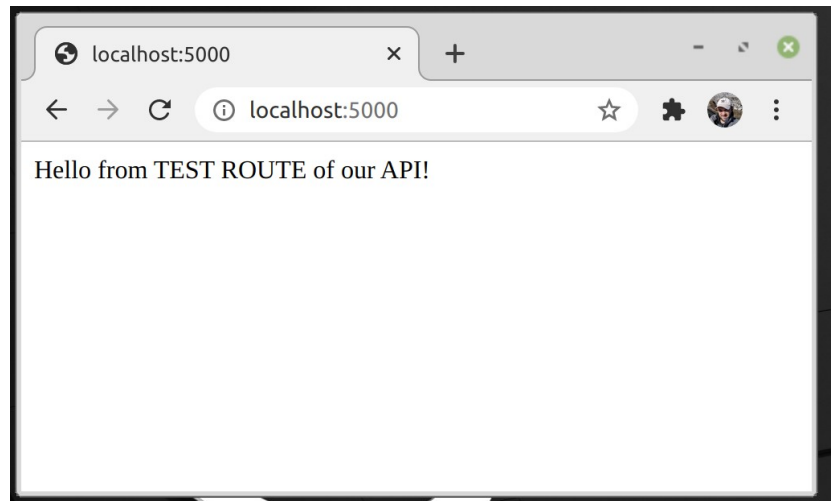
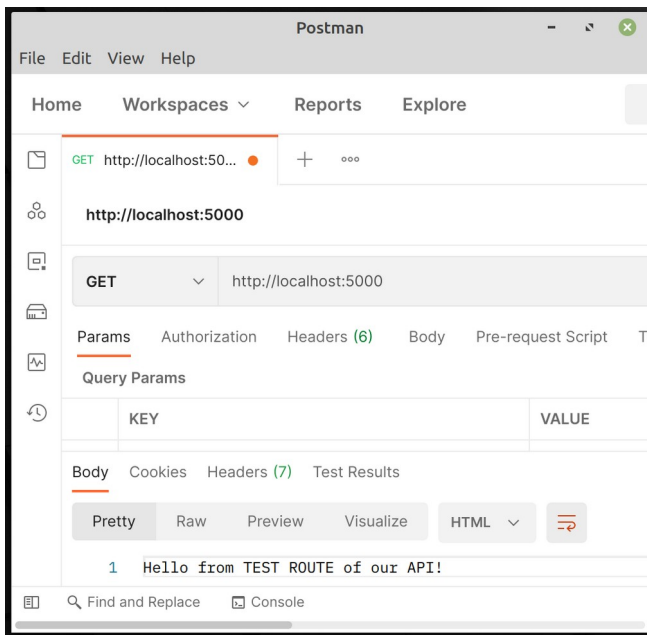
2.2.7 Change start scripts at package.json

```
{ } package.json X
{ } package.json > ...
1  {
2    "name": "mern-devconnector",
3    "version": "1.0.0",
4    "description": "social network MERN stack",
5    "main": "server.js",
   ▶ Debug
6    "scripts": {
7      "start": "node server",
8      "server": "nodemon server.js"
9    },
```

Now to start server: [CLI=> npm run server]. It will start SERVER.JS file with NODEMON

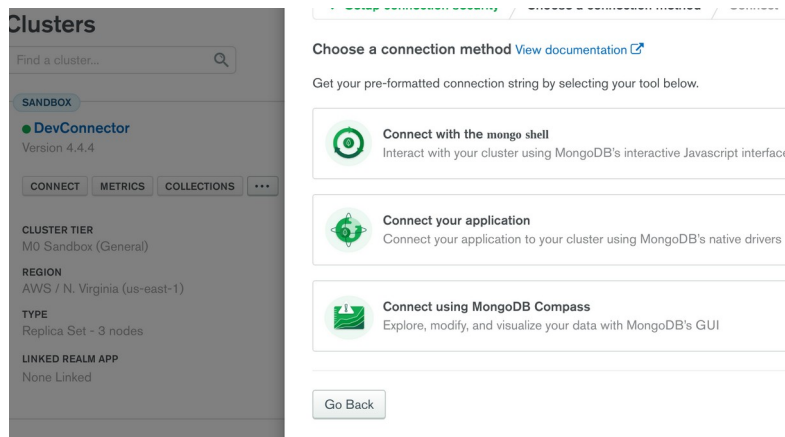
2.2.8 Create a SERVER with simple test route

```
JS server.js X
JS server.js > ...
1  //Import EXPRESS
2  const express = require("express");
3
4  //Initialize a main APP variable
5  const app = express();
6
7  //Create a TEST ROUTE (for GET request to root folder '/')
8  app.get("/", (req, res) => res.send("Hello from TEST ROUTE of our API!"));
9
10 //Create a variable to store port for server
11 const PORT = process.env.PORT || 5000;
12
13 //Create an APP LISTENER.
14 app.listen(PORT, () => {
15   console.log(`Server was started on port ${PORT}`);
16 });
```



2.3 Connecting To MongoDB With Mongoose

2.3.1 Create a connection to database



2.3.2 Create a connection logic on server

```

1  const mongoose = require("mongoose");
2
3  //Import CONFIG
4  const config = require("config");
5
6  //Import mongoURI from default.json in folder config
7  const db = config.get("mongoURI");
8
9  //Create a connection
10 const connectDB = async () => {
11   try {
12     //TRY to connect
13     await mongoose.connect(db, {
14       useNewUrlParser: true,
15       useUnifiedTopology: true,
16     });
17     // If DB was connected - log a message
18     console.log("MongoDB was connected ...");
19   } catch (error) {
20     //If ERROR log error and exit process with a failure
21     console.error(error.message);
22     process.exit(1);
23   }
24 };
25
26 module.exports = connectDB;

```

```

JS server.js X
JS server.js > ...
1 //Import EXPRESS
2 const express = require("express");
3
4 //Import logic to CONNECT to a DATABASE
5 const connectDB = require("./config/db");
6
7 //Initialize a main APP variable
8 const app = express();
9
10 //Create a CONNECTION to DATABASE
11 connectDB();
12
13 //Create a TEST ROUTE (for GET request to root folder '/')
14 app.get("/", (req, res) => res.send("Hello from TEST ROUTE of our API!"));
15
16 //Create a variable to store port for server
17 const PORT = process.env.PORT || 5000;
18
19 //Create an APP LISTENER.
20 app.listen(PORT, () => {
21 | console.log(`Server was started on port ${PORT}`);
22 | });
23

```

```

stslon@stslon-System-Product-Name:/media/stslon/8CB04F45B04F354C/
> mern-devconnector@1.0.0 server
> nodemon server

[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:24960) Warning: Accessing non-existent property 'MongoError'
(Use `node --trace-warnings ...` to show where the warning was created)
Server was started on port 5000
(node:24960) DeprecationWarning: Listening to events on the Db class
MongoDB was connected ...

```

2.4 Create route files with Express Router

```

JS server.js X
JS server.js > ...
1 //Import EXPRESS
2 const express = require("express");
3
4 //Import logic to CONNECT to a DATABASE
5 const connectDB = require("./config/db");
6
7 //Initialize a main APP variable
8 const app = express();
9
10 //Create a CONNECTION to DATABASE
11 connectDB();
12
13 //Create MAIN routes
14 app.use("/api/user", require("./routes/api/user"));
15 app.use("/api/profile", require("./routes/api/profile"));
16 app.use("/api/post", require("./routes/api/post"));
17 app.use("/api/auth", require("./routes/api/auth"));
18
19 //Create a variable to store port for server
20 const PORT = process.env.PORT || 5000;
21
22 //Create an APP LISTENER.
23 app.listen(PORT, () => {
24 | console.log(`Server was started on port ${PORT}`);
25 | });
26

```

```

EXPLORER
... JS user.js X
routes > api > JS user.js > ...
1 //Import EXPRESS
2 const express = require("express");
3
4 //Import EXPRESS ROUTER
5 const router = express.Router();
6
7 router.get("/login", (req, res) => {
8 | res.send("hello from api/user/login");
9 | });
10
11 router.get("/", (req, res) => {
12 | res.send("hello from api/user");
13 | });
14
15 module.exports = router;
16

```

3. User API Routes & JWT Authentication