

# Build a REST API

NODE | EXPRESS | MONGO

# What is HTTP...

HyperText Transfer Protocol

- Application Layer Protocol
- Built on top of TCP/IP protocol
- Rules for transferring resources
- Every HTTP Request is executed independently without the knowledge of requests that came before it

# ...What is HTTP

HyperText Transfer Protocol

- HTTP is stateless
- TCP/IP is not stateless
- Payload (body/data) can be anything as long as it is defined in header

# ...What is HTTP

HyperText Transfer Protocol

- HTTP is stateless
- TCP/IP is not stateless
- Payload (body/data) can be anything as long as it is defined in header

# What is REST...

Representational State Transfer

- Architectural Pattern with design guidelines
- HTTP is usually the underlying protocol
- Use HTTP methods explicitly
- Every RESTful resource has a unique ID

# ...What is REST

Representational State Transfer

- Client state is not persisted between requests
- Caching Policy for responses
- Separation of concerns between clients and servers
- Layered system

# What is an API...

Application Programming Interface

- Defines server-side functions that are supported
- Where requests should be made
- Format of the request and response

# ...What is an API

Application Programming  
Interface

- No standard way of writing APIs
- REST provides guidelines
- CRUD Operations
- Create | Read | Update | Delete



# IDE

Integrated Development Environment

- VS Code
- Atom
- Webstorm (Paid / Trial)

# Node.js

Javascript Runtime

- Node.js 8.9.3+
- NPM (Node Package Manager)
- Using natively supported ES6/7
- <https://node.green>

# MongoDB

NoSQL Database

- Local Installation
- mLab Cloud
- Docker

# REST Client

API Testing

- Insomnia
- Postman

# REST Client

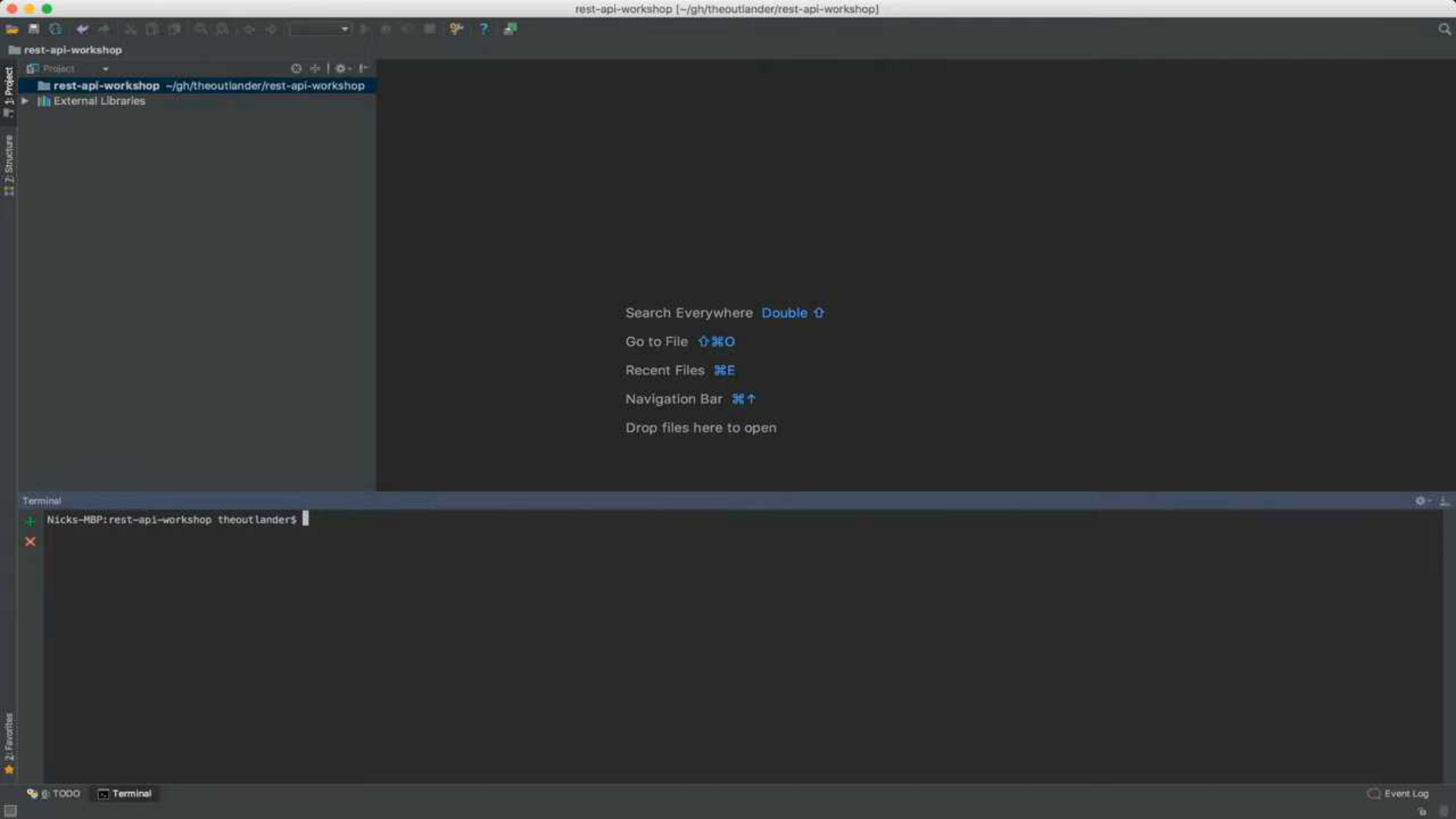
API Testing

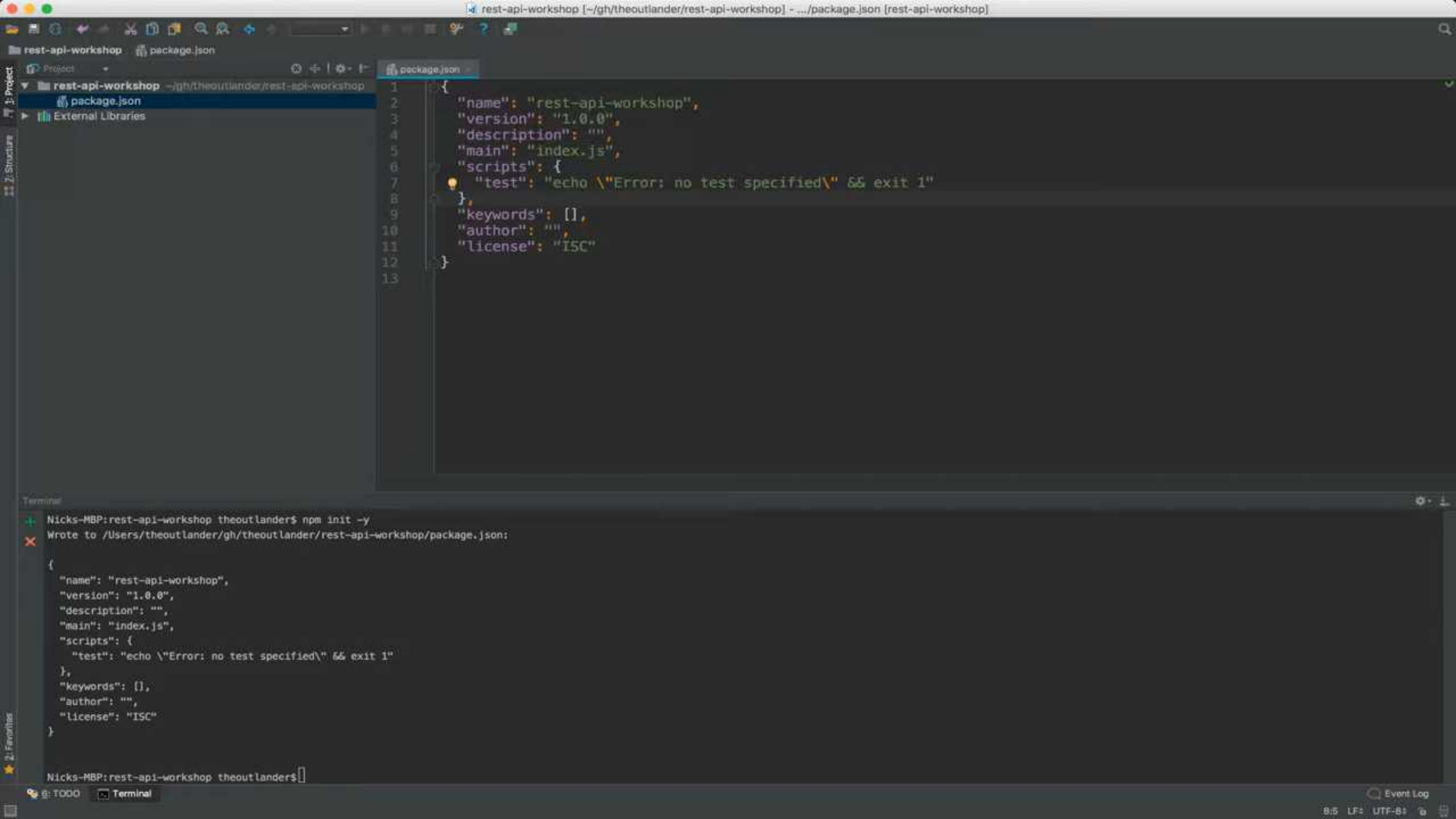
- Insomnia
- Postman

# Application Setup

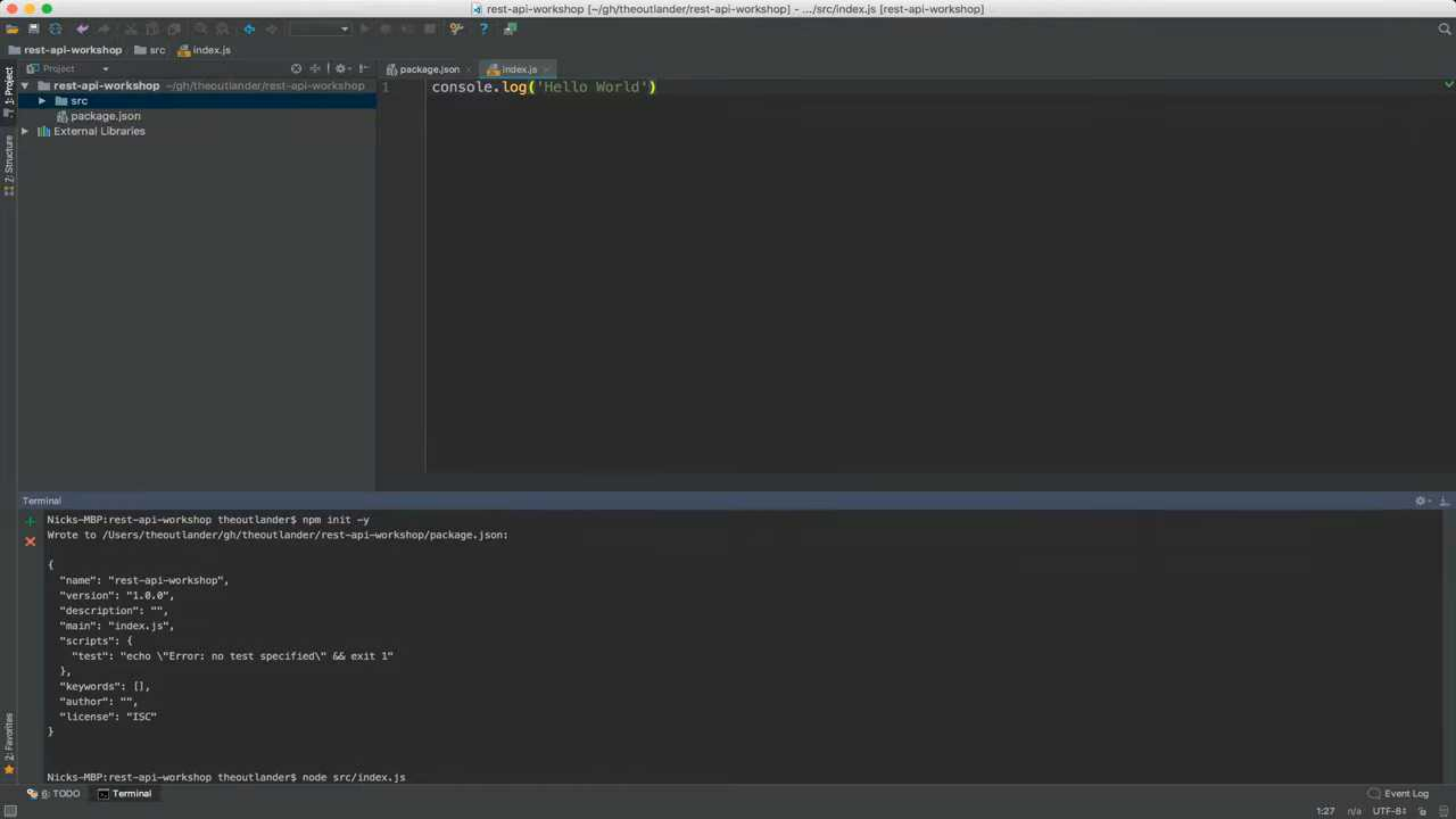
Getting Started

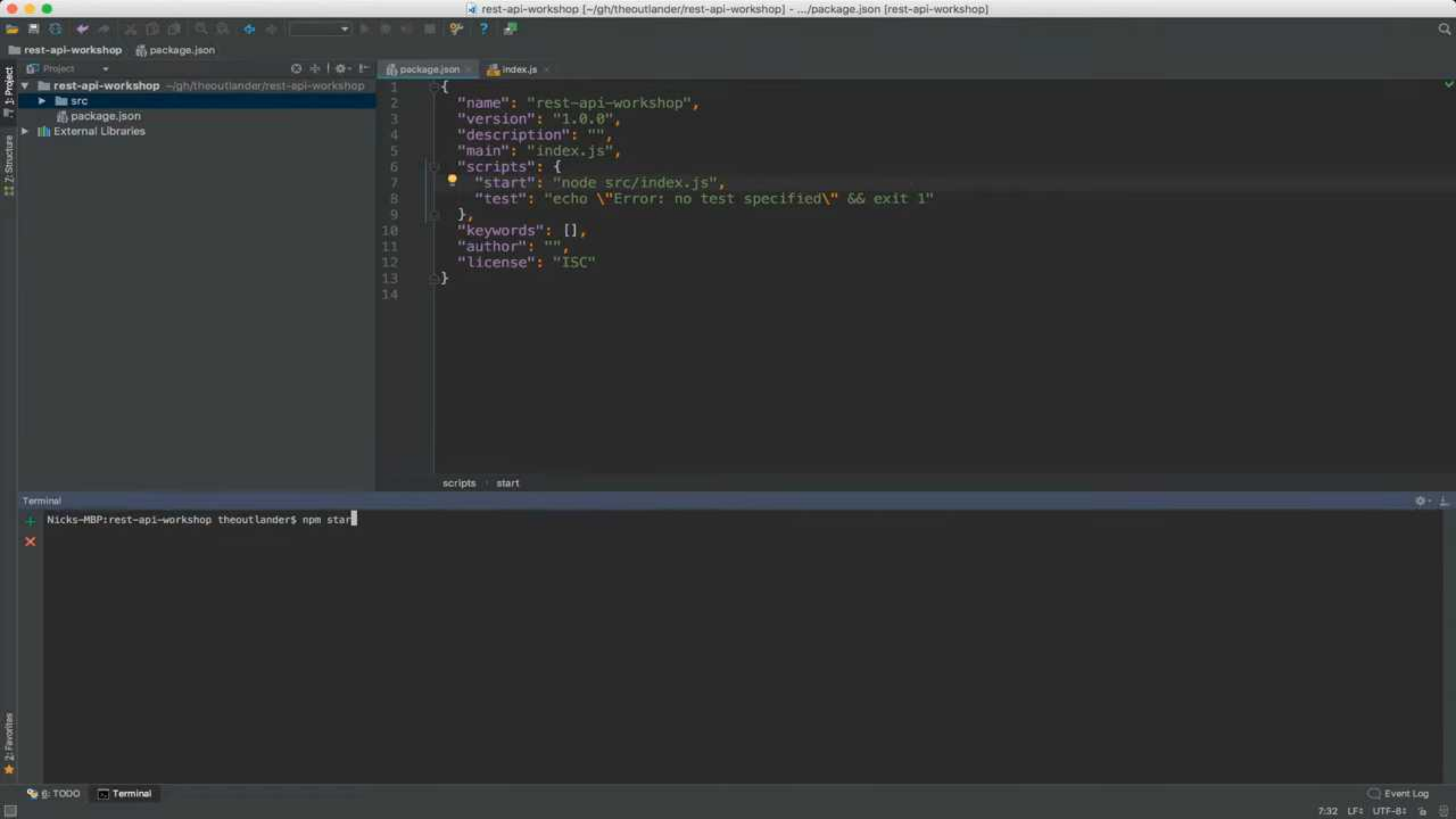
- Project Folder
- Initialize Node.js Project
- Hello World
- Startup Scripts







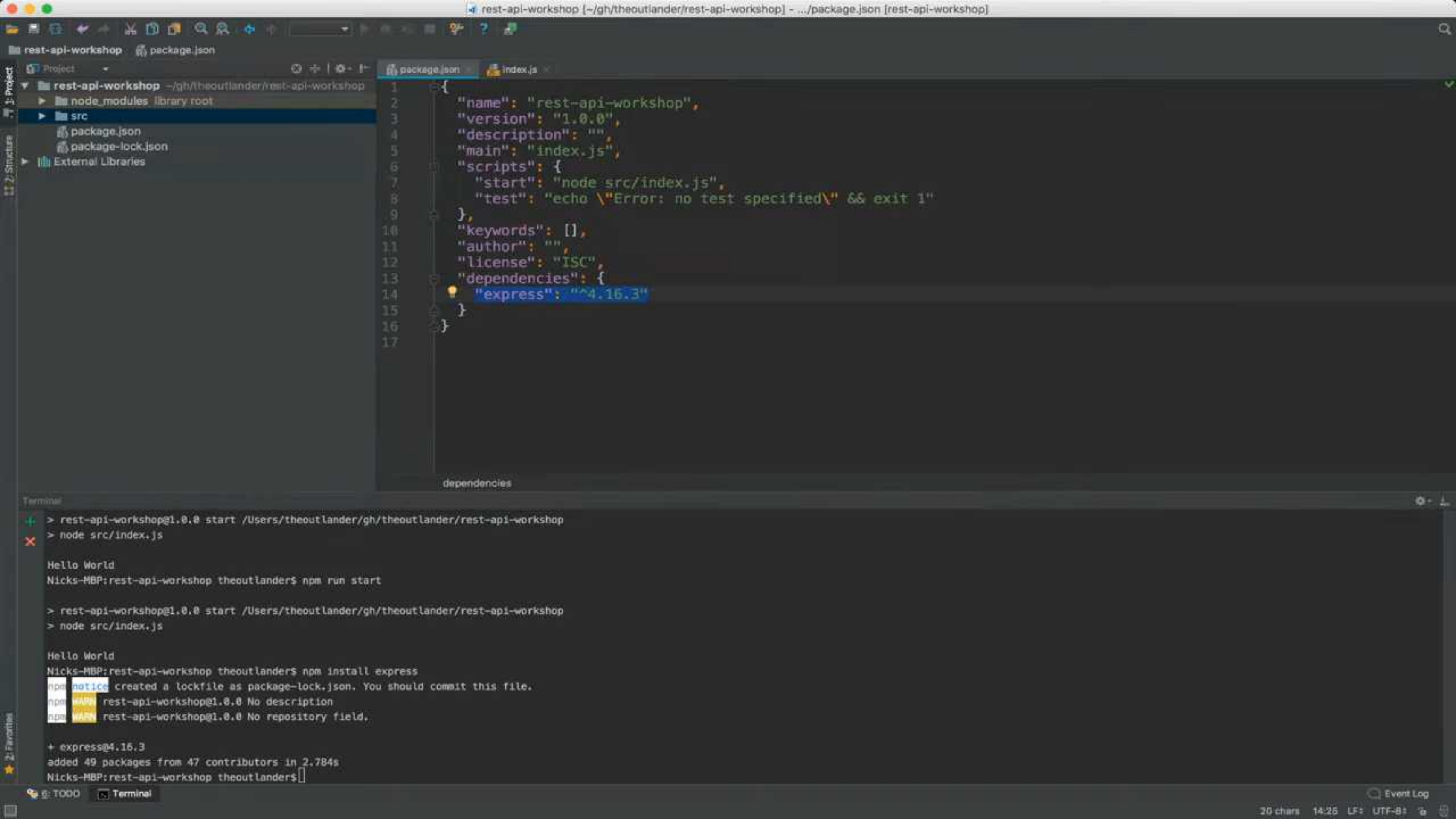


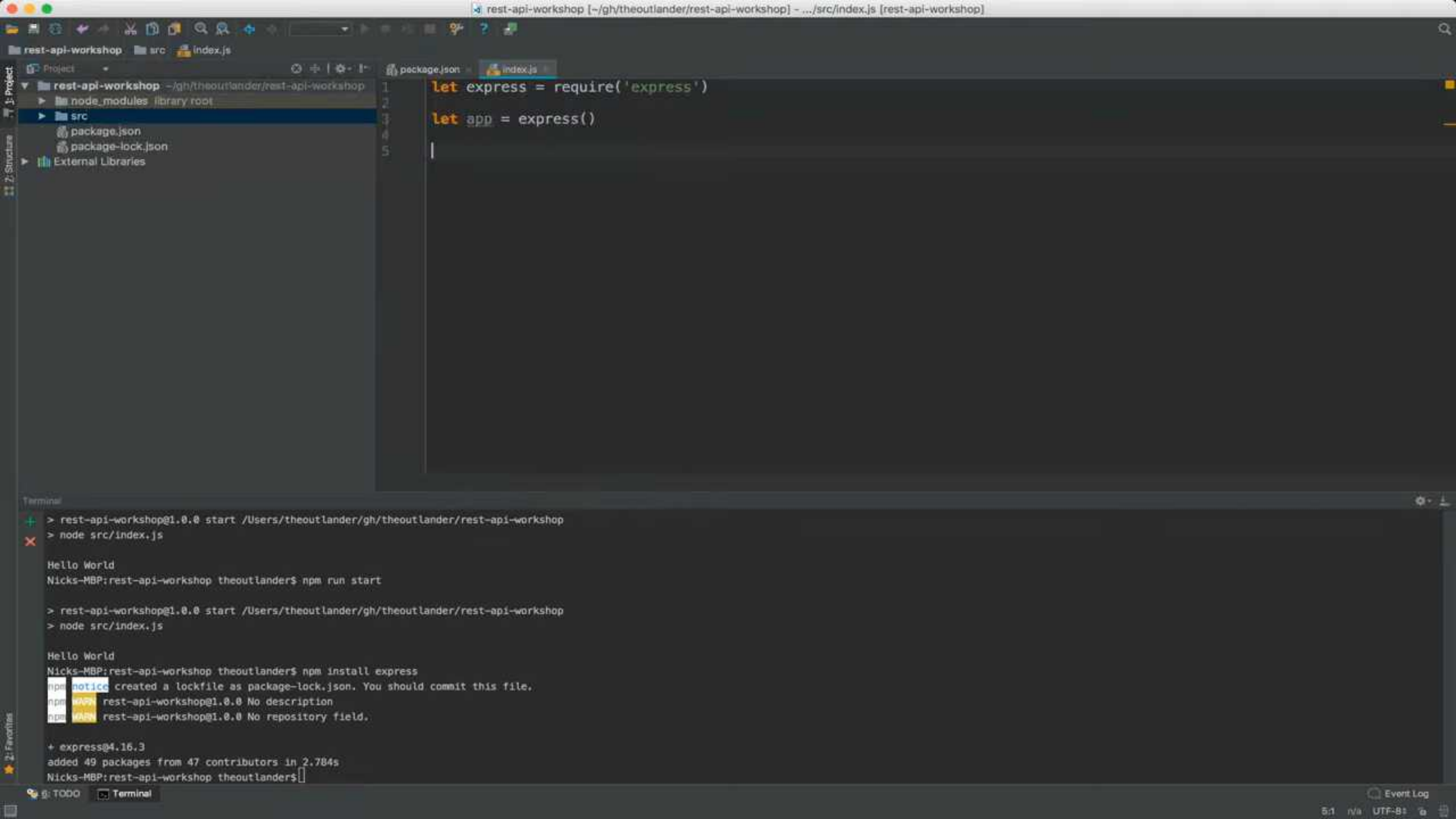


# Express

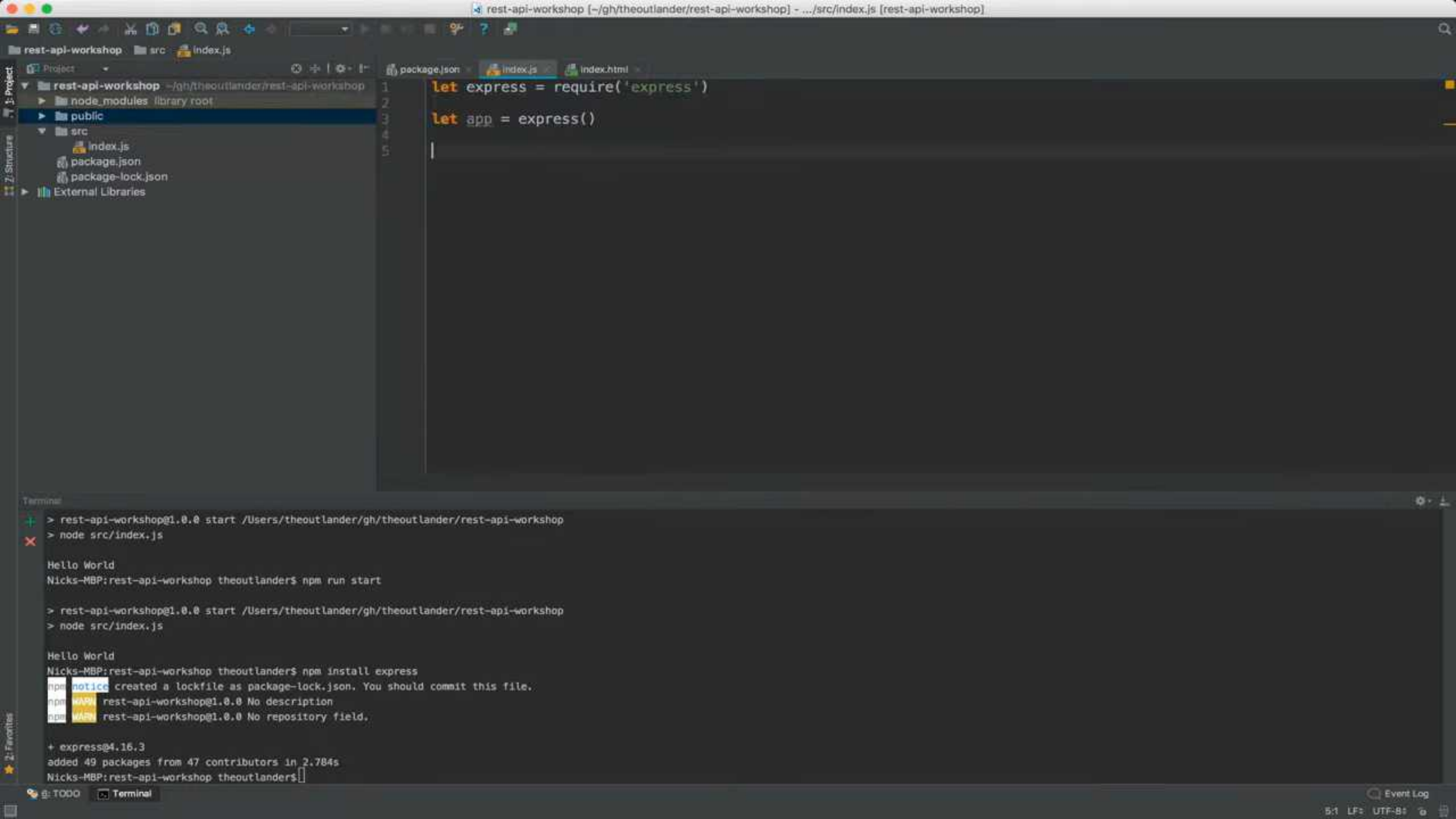
Web Server

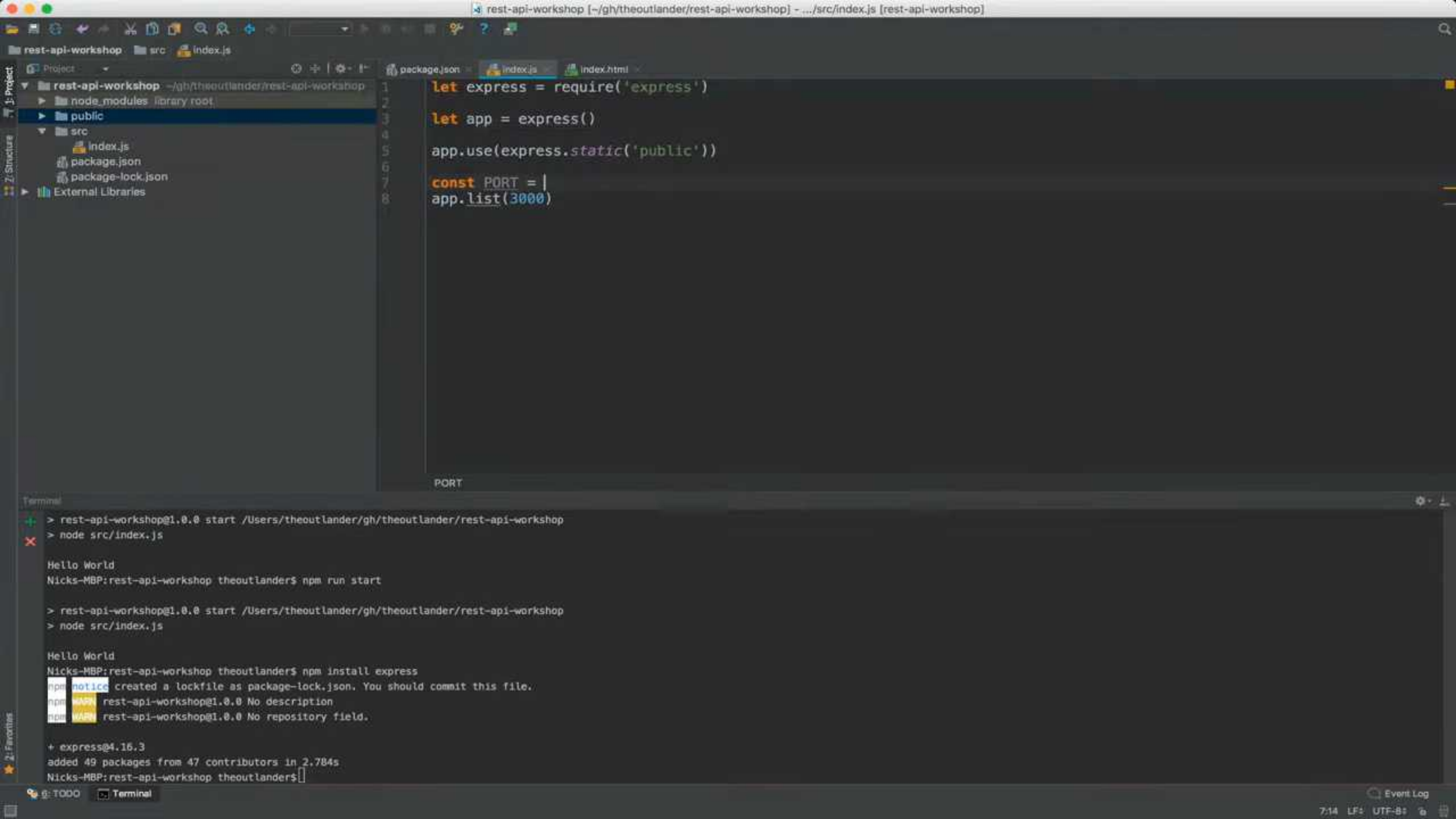
- Install Dependency
- Reference the express library
- Create express application instance

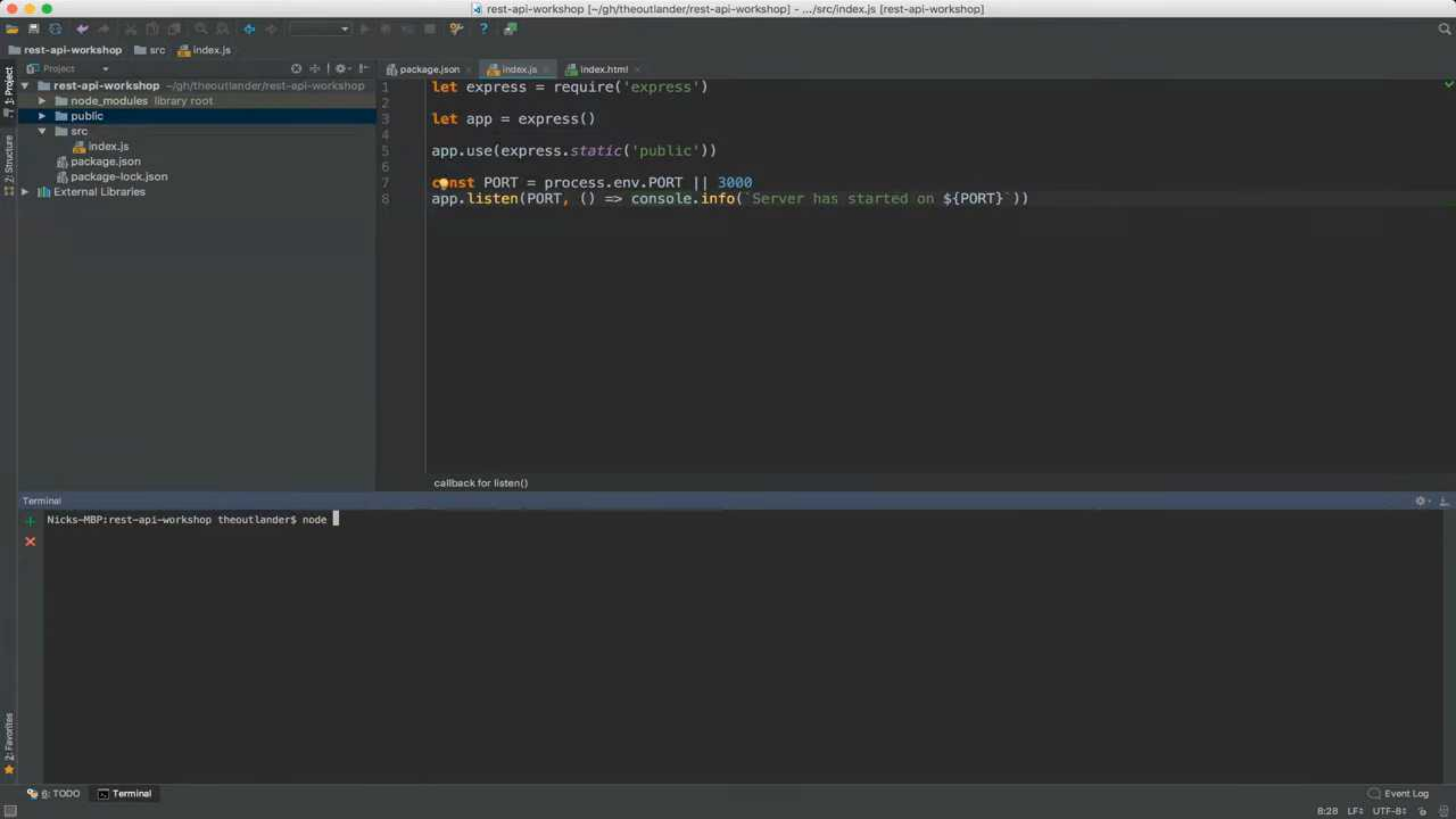




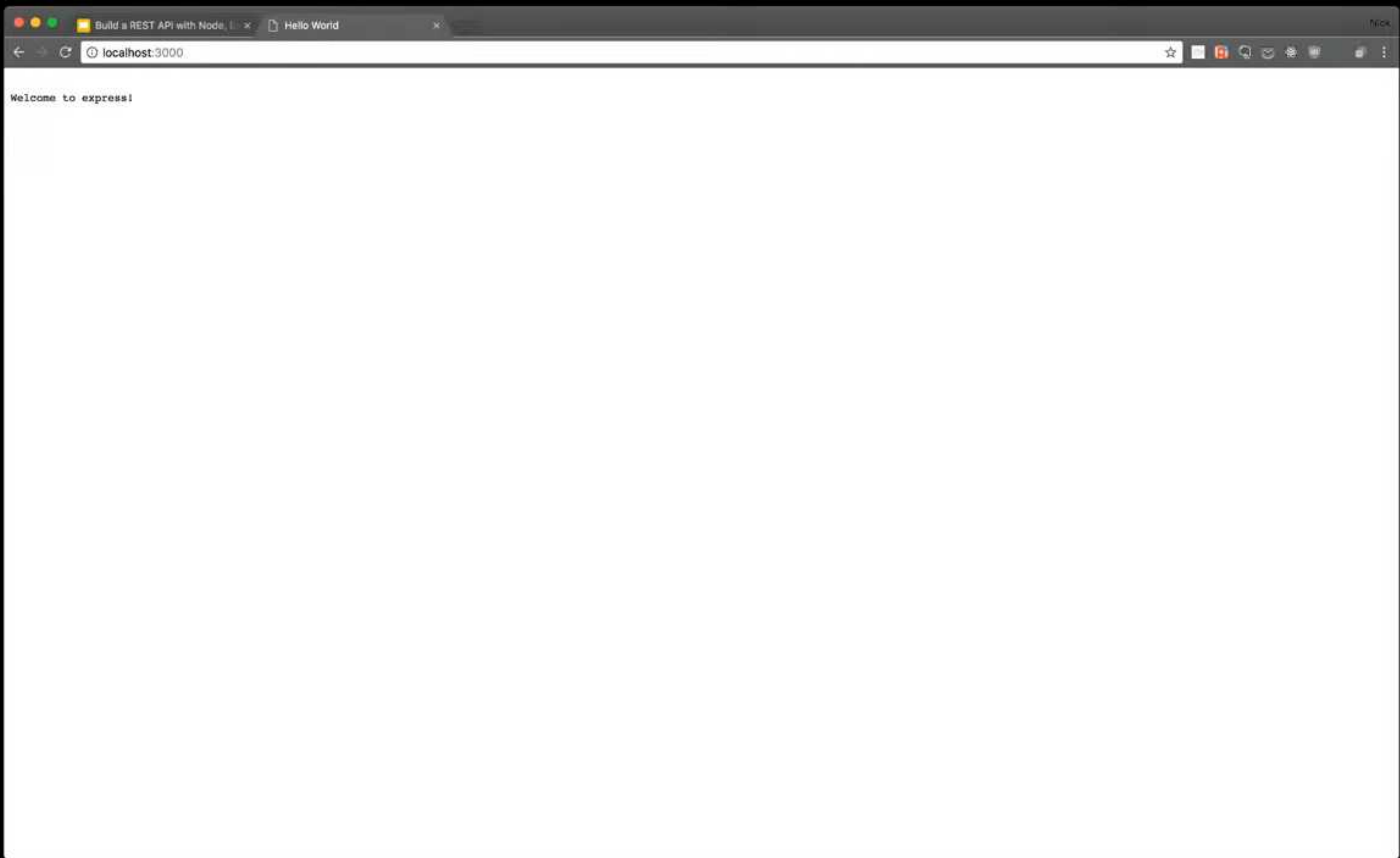








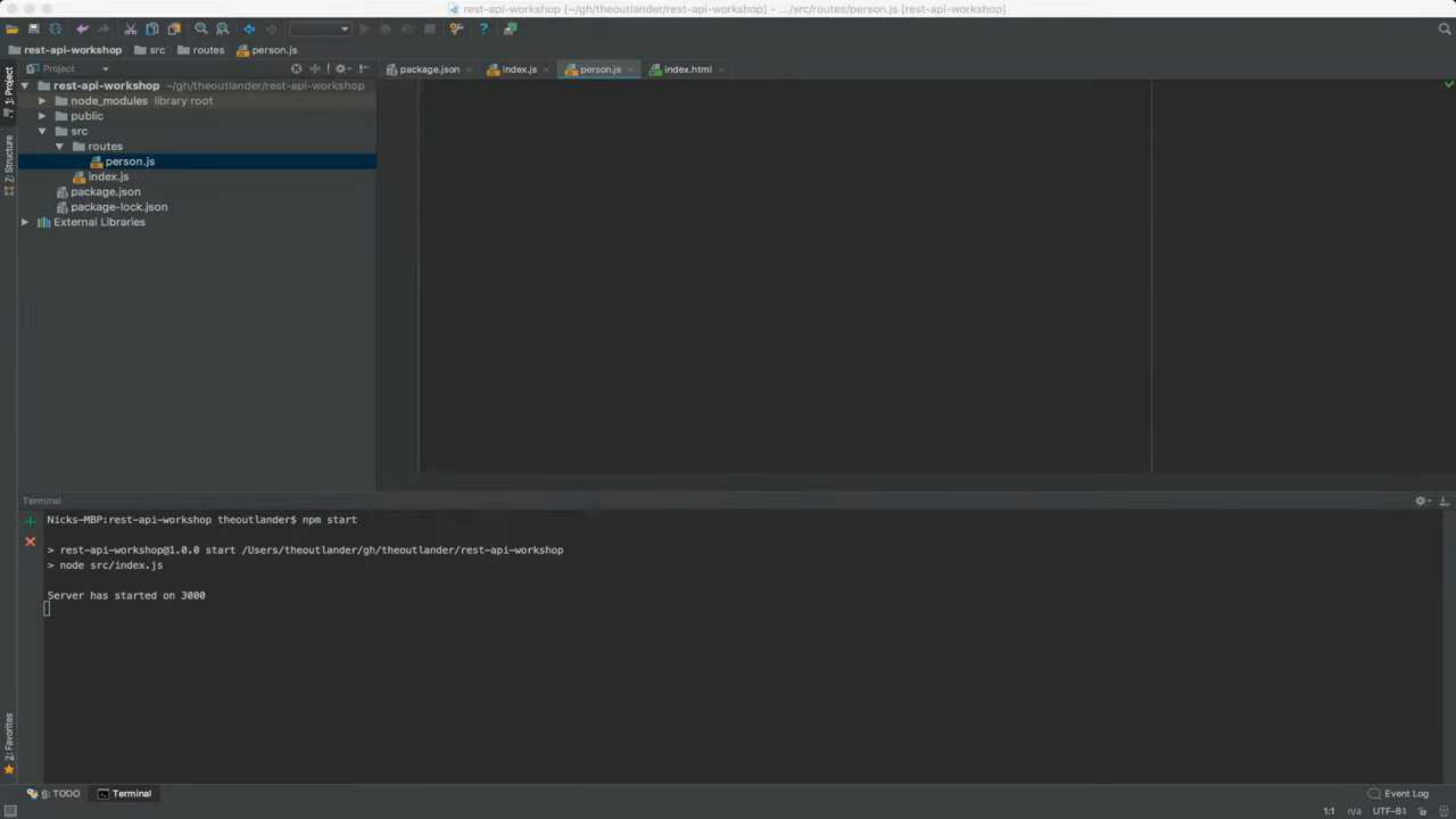


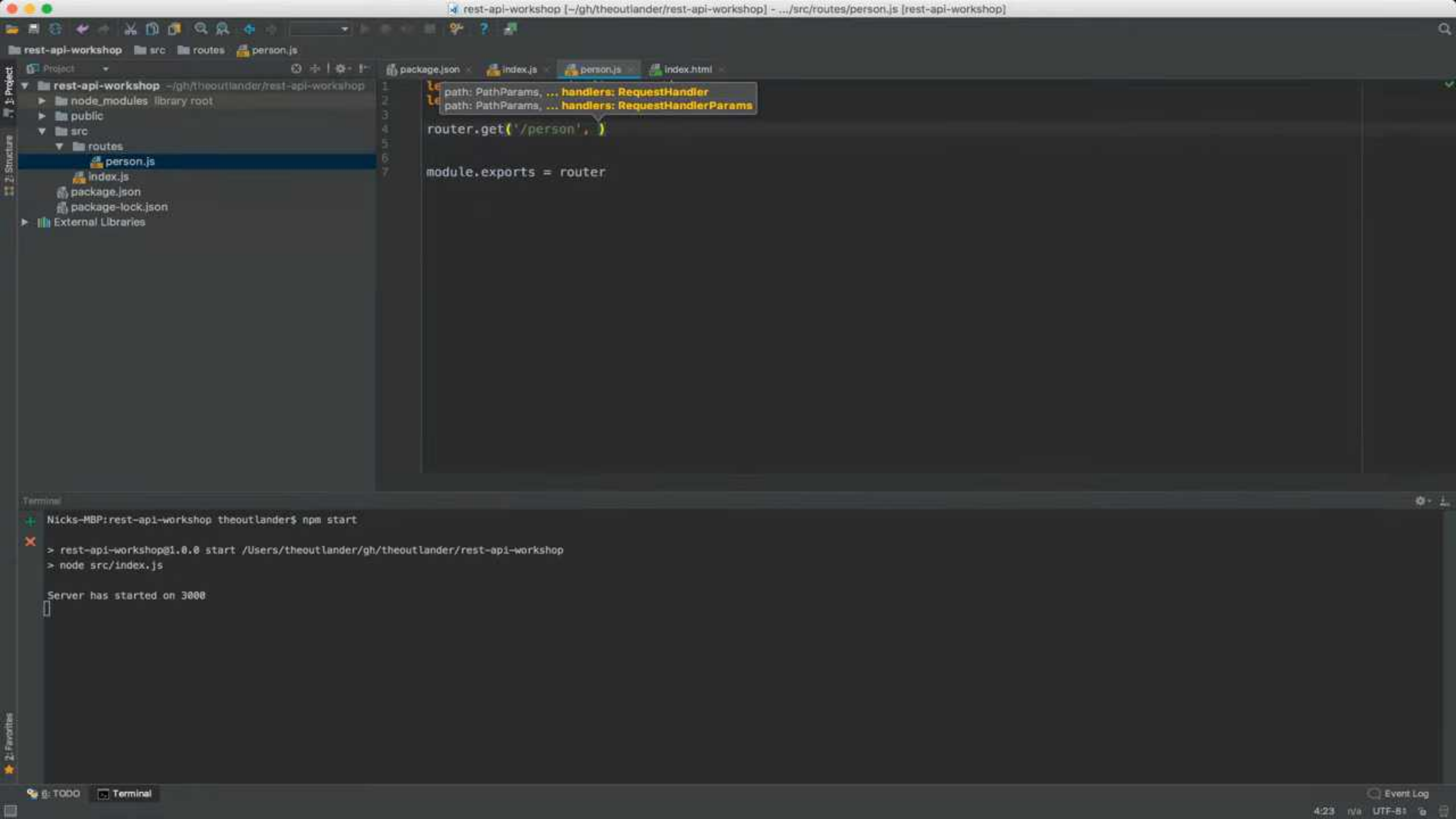


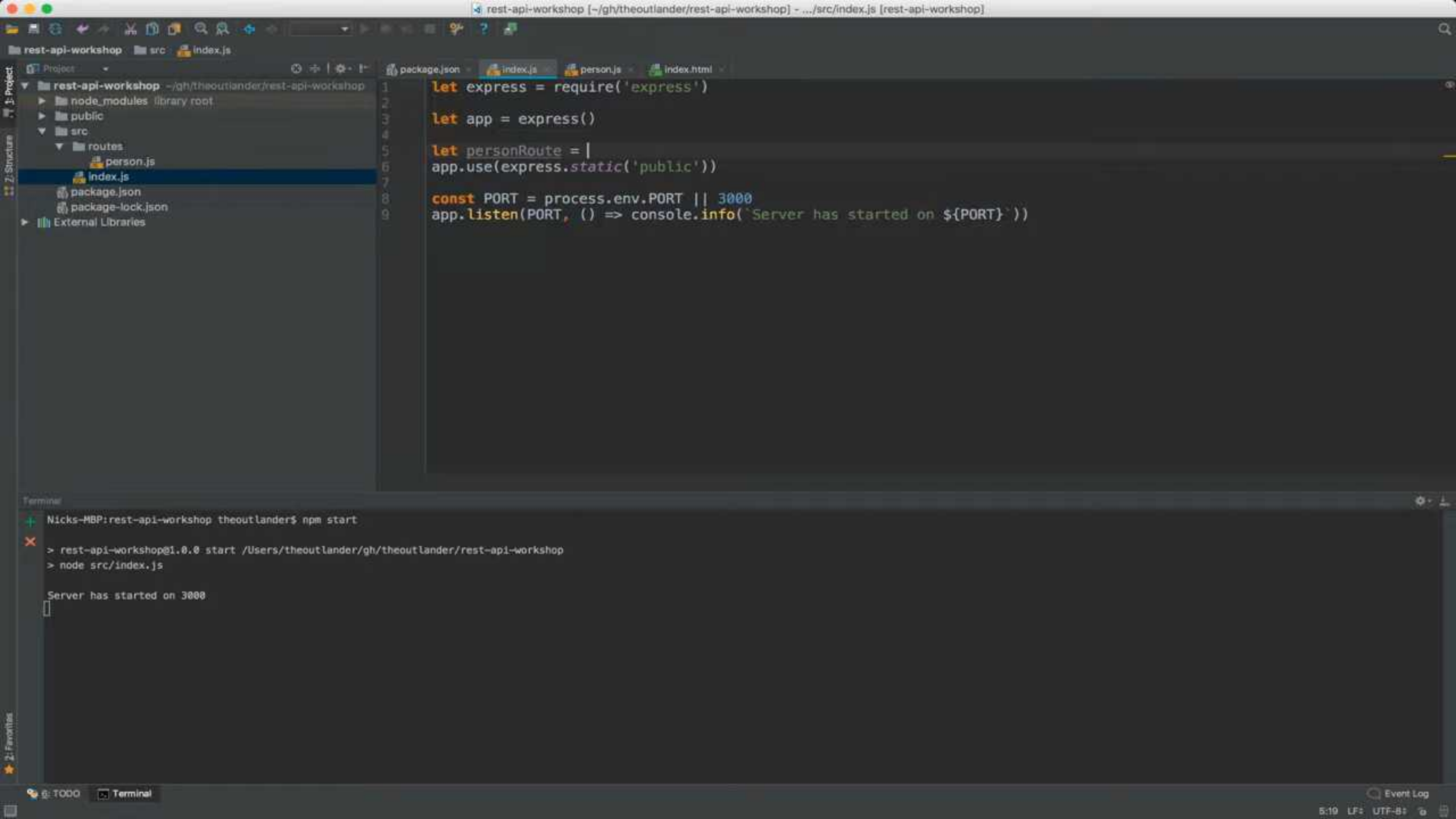
# Route

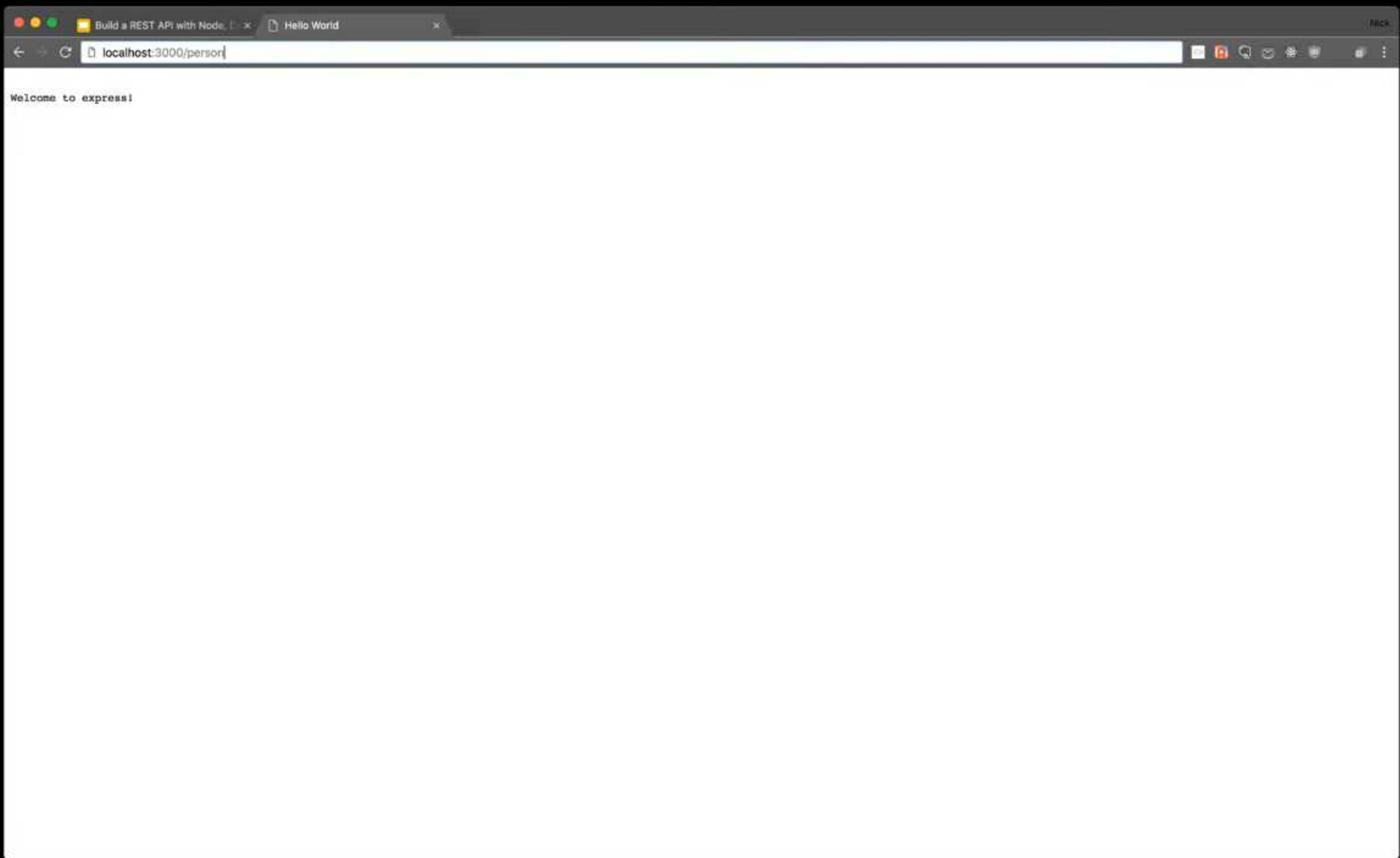
Mount to Express

- GET Request
- Request Parameters
- Query Parameters











No Environment Cookies

Filter

REST API Workshop

GET GET Person

Projects

- GET
- POST
- PUT
- PATCH
- DELETE
- OPTIONS
- HEAD
- Custom Method

Body Header Docs

  
Select a body type from above

- Send Request
- Focus URL Bar
- Manage Cookies
- Edit Environments



Insomnia

No Environment Cookies

Filter

REST API Workshop

GET GET Person

Projects

GET

POST

PUT

PATCH

DELETE

OPTIONS

HEAD

Custom Method

GET


https://api.myproduct.com/v1/users

Send

Body

Header

Docs



Select a body type from above

Send Request

⌘Enter

Focus Url Bar

⌘L

Manage Cookies

⌘K

Edit Environments

⌘E





GET http://localhost:3000/person

Send

200 OK

TIME 5.44 ms

SIZE 27 B



No Environment Cookies

Filter

REST API Workshop

GET GET Person

Projects

Body

Auth

Query

Header

Docs

Preview

Header

Cookie

Timeline



Select a body type from above

You have requested a person



## Generate Client Code

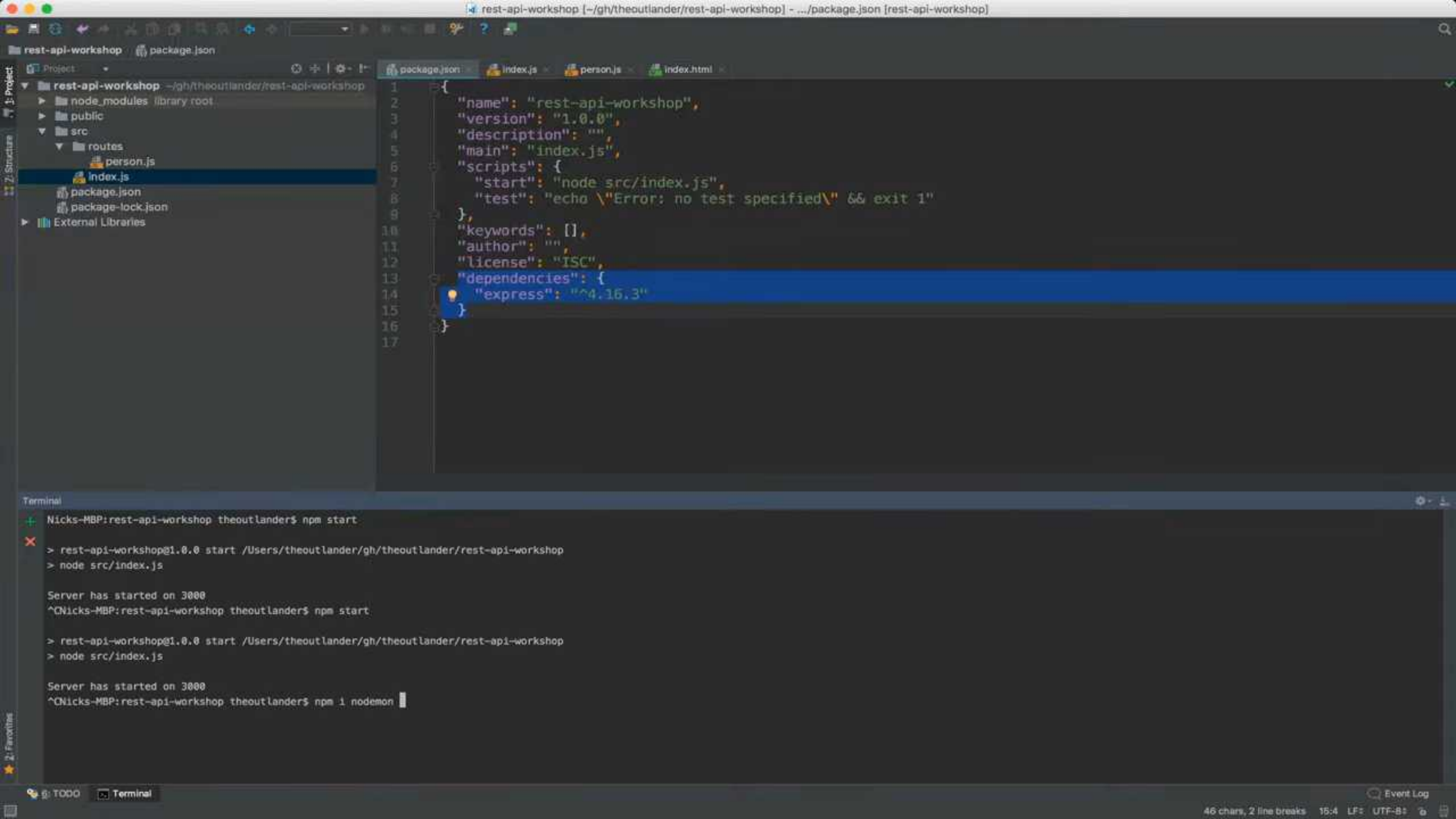


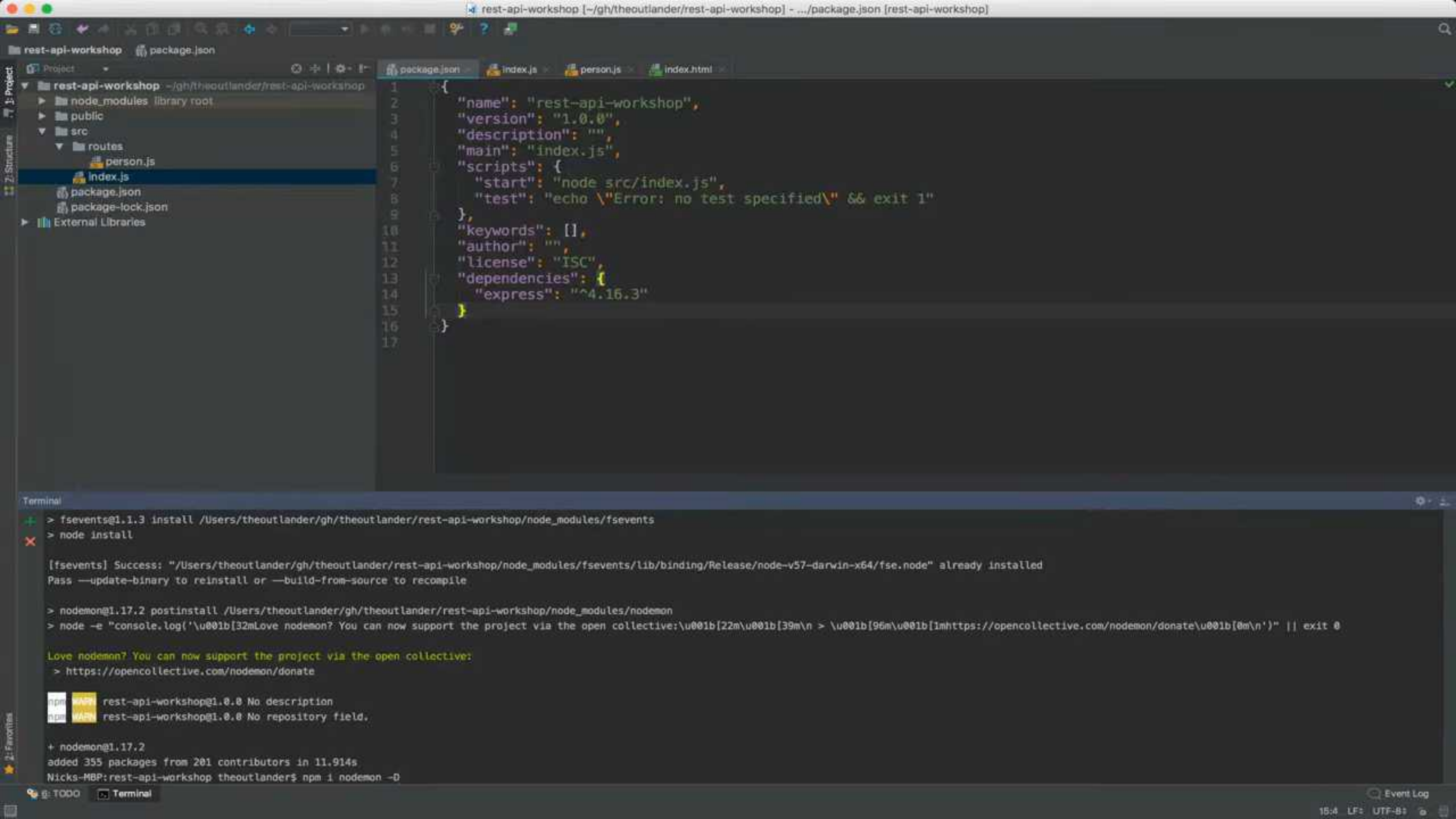
Node.js

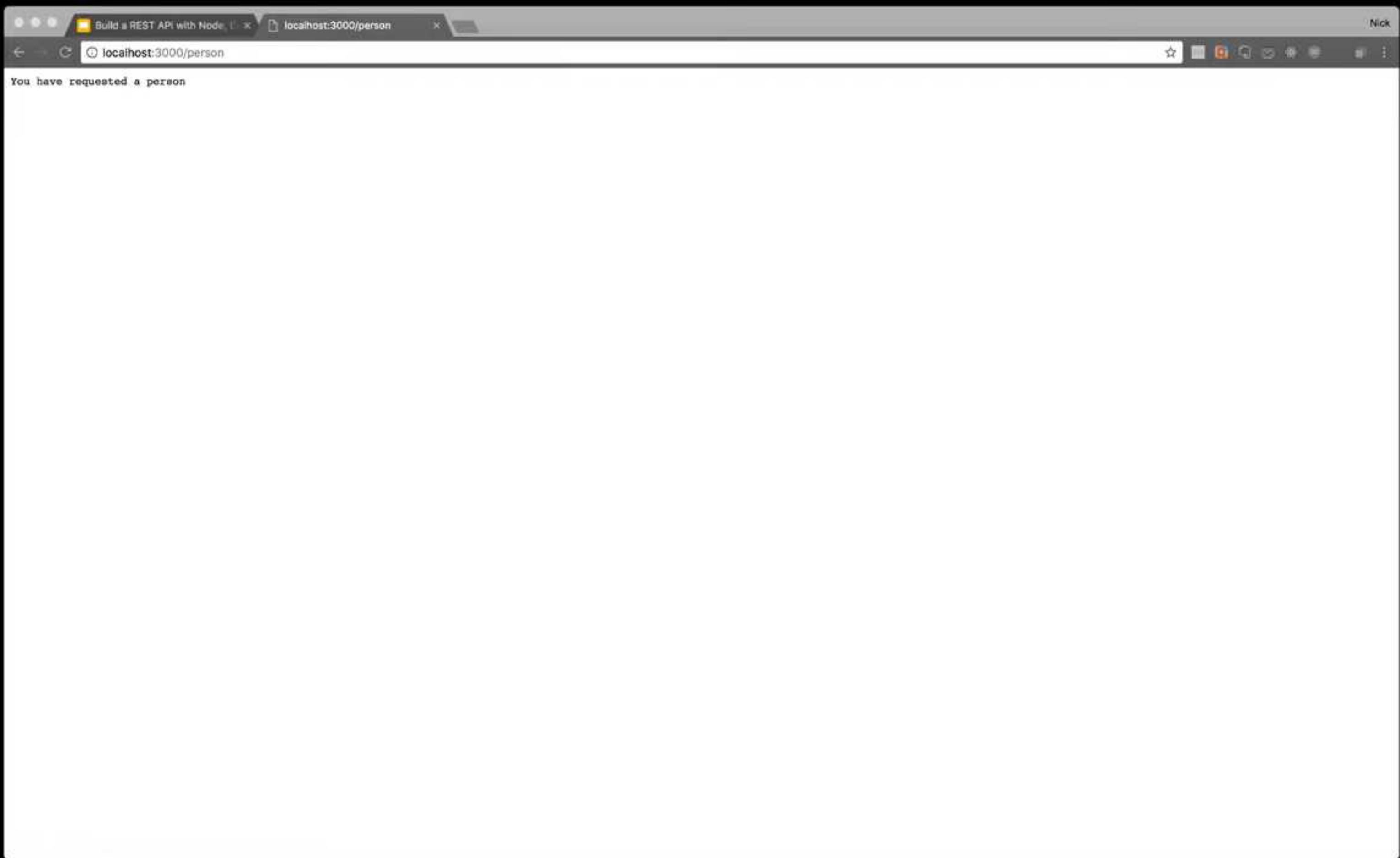
Request

Copy to Clipboard

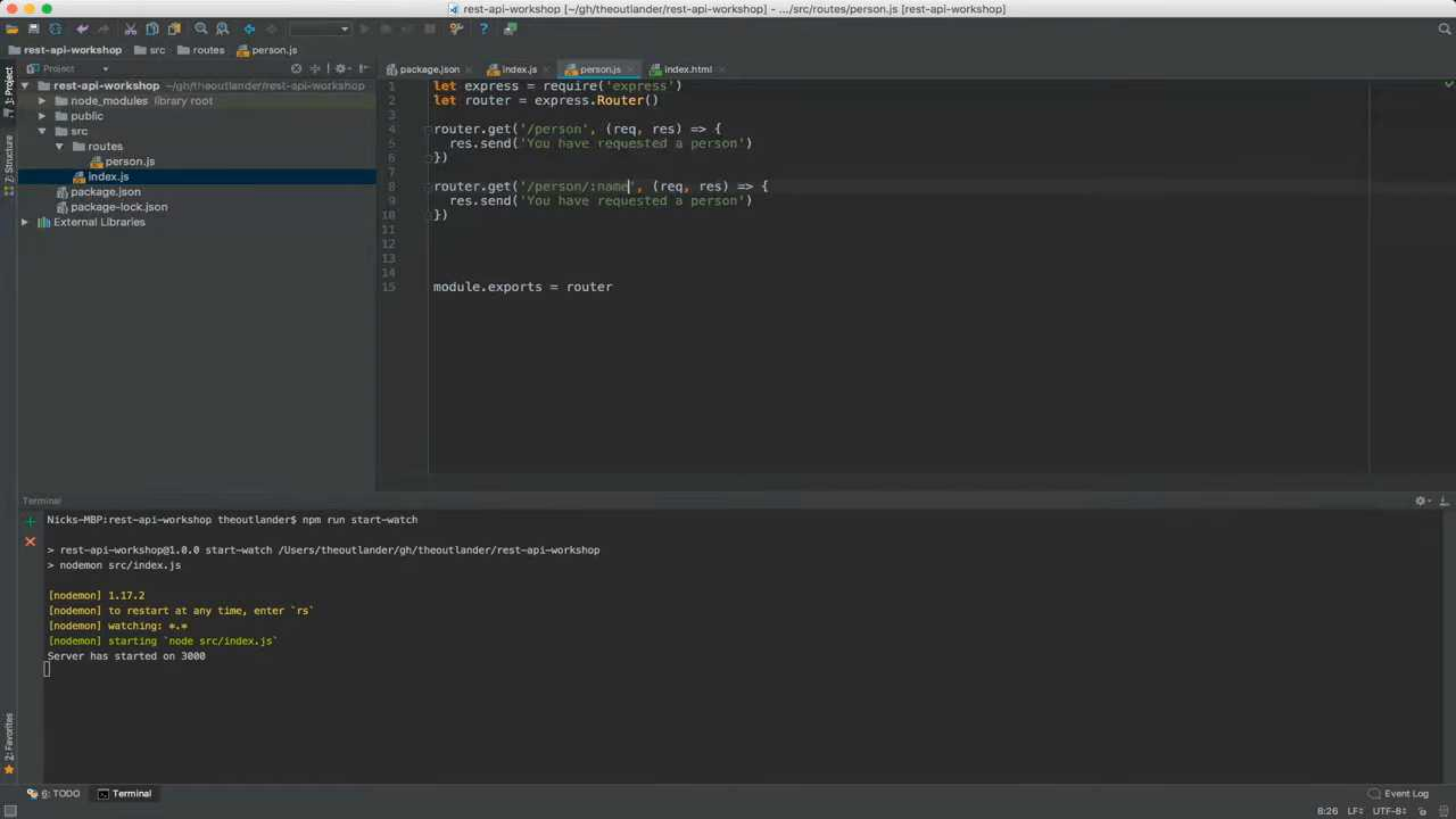
```
1 var request = require("request");
2
3 var jar = request.jar();
4 jar.setCookie(request.cookie("connect.sid=s%253A8IqZ3303G5mRRHoqgAX4t_7sVlZNBAfe.6Ys4HKIjv0h91dGSvXVo5zRSv7MzXc
  gFZC4zU84jbb8Y"), "http://localhost:3000/person");
5
6 var options = { method: 'GET',
7   url: 'http://localhost:3000/person',
8   jar: 'JAR' };
9
10 request(options, function (error, response, body) {
11   if (error) throw new Error(error);
12
13   console.log(body);
14 });
15
```

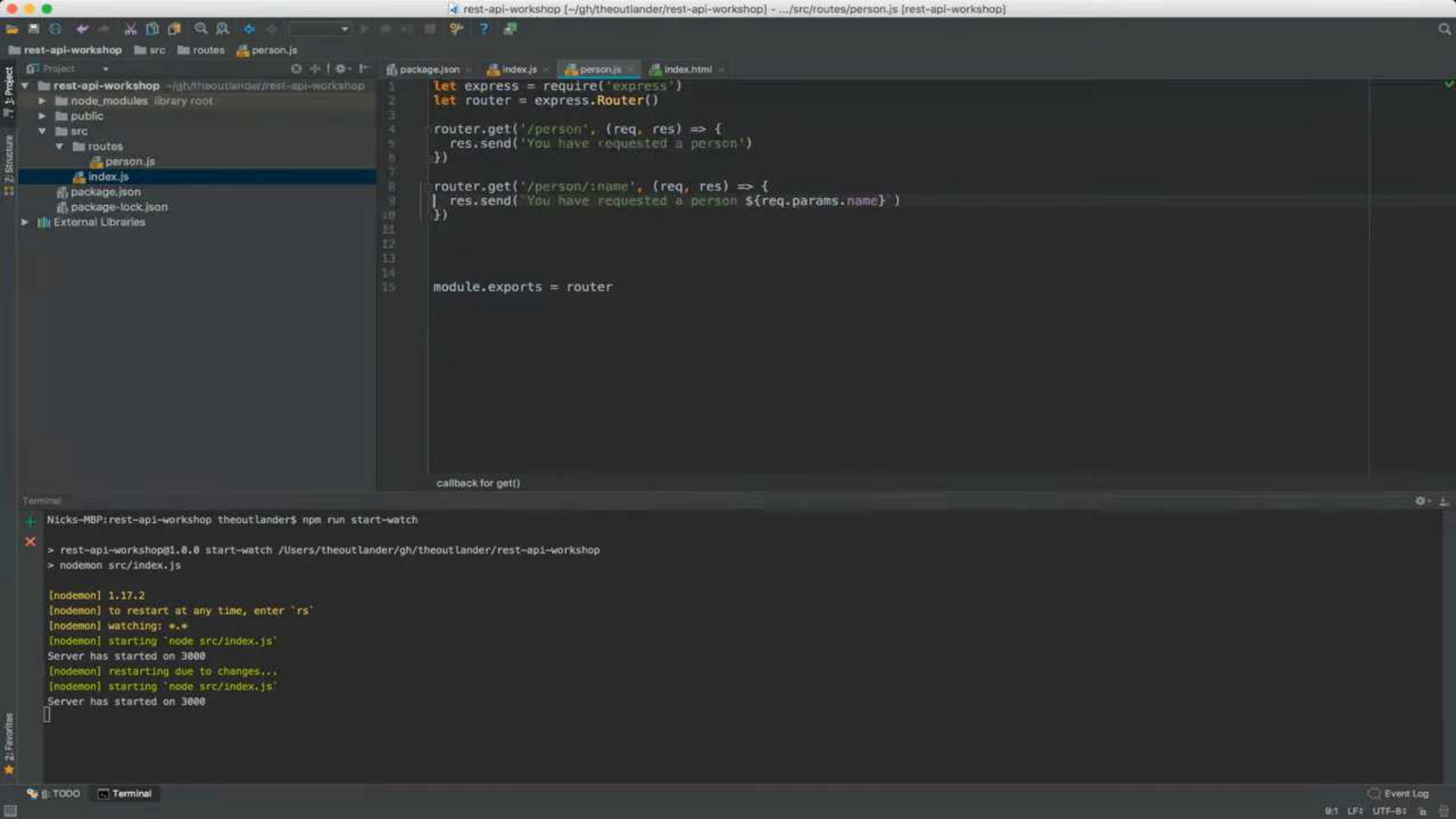


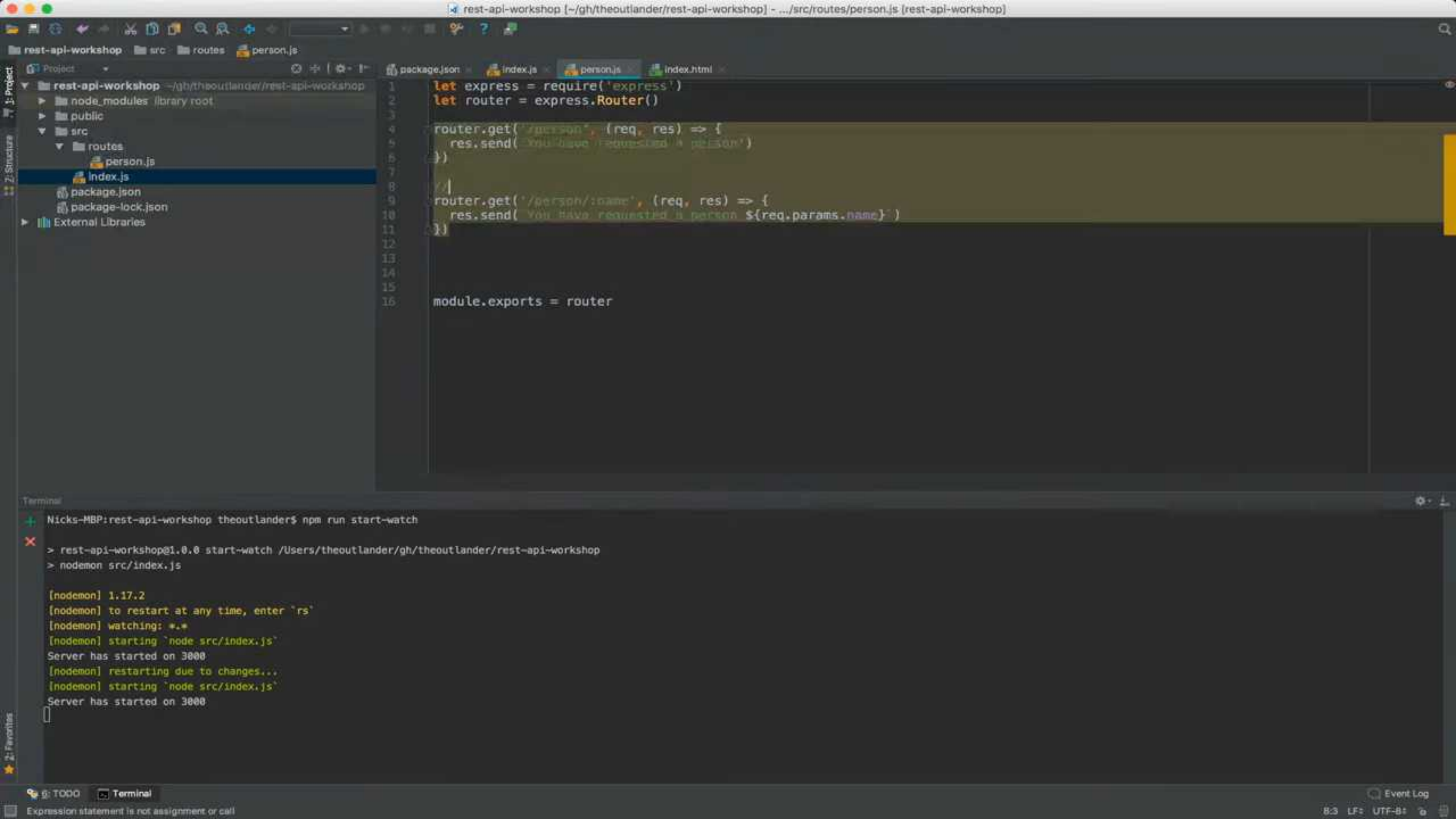




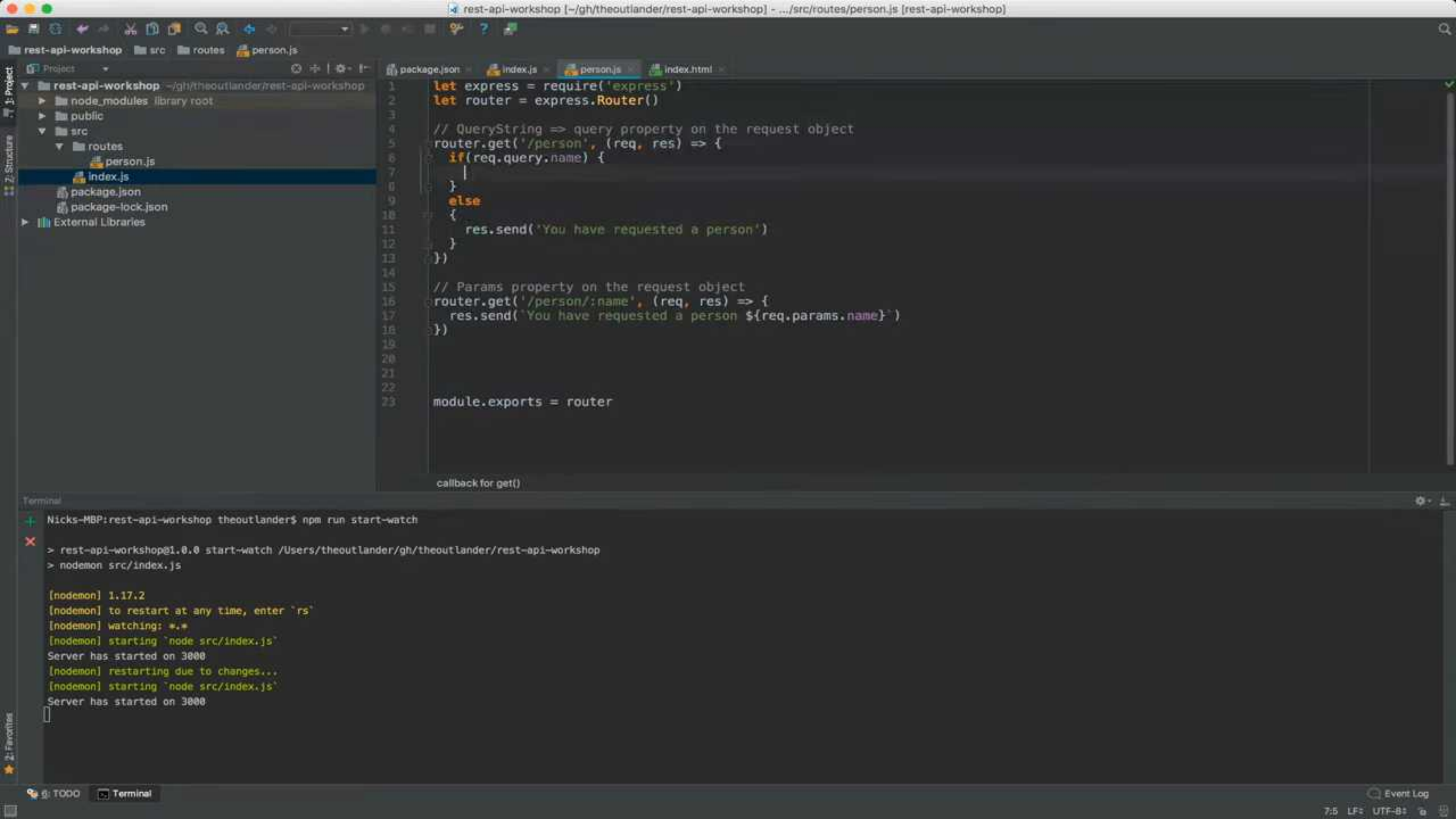


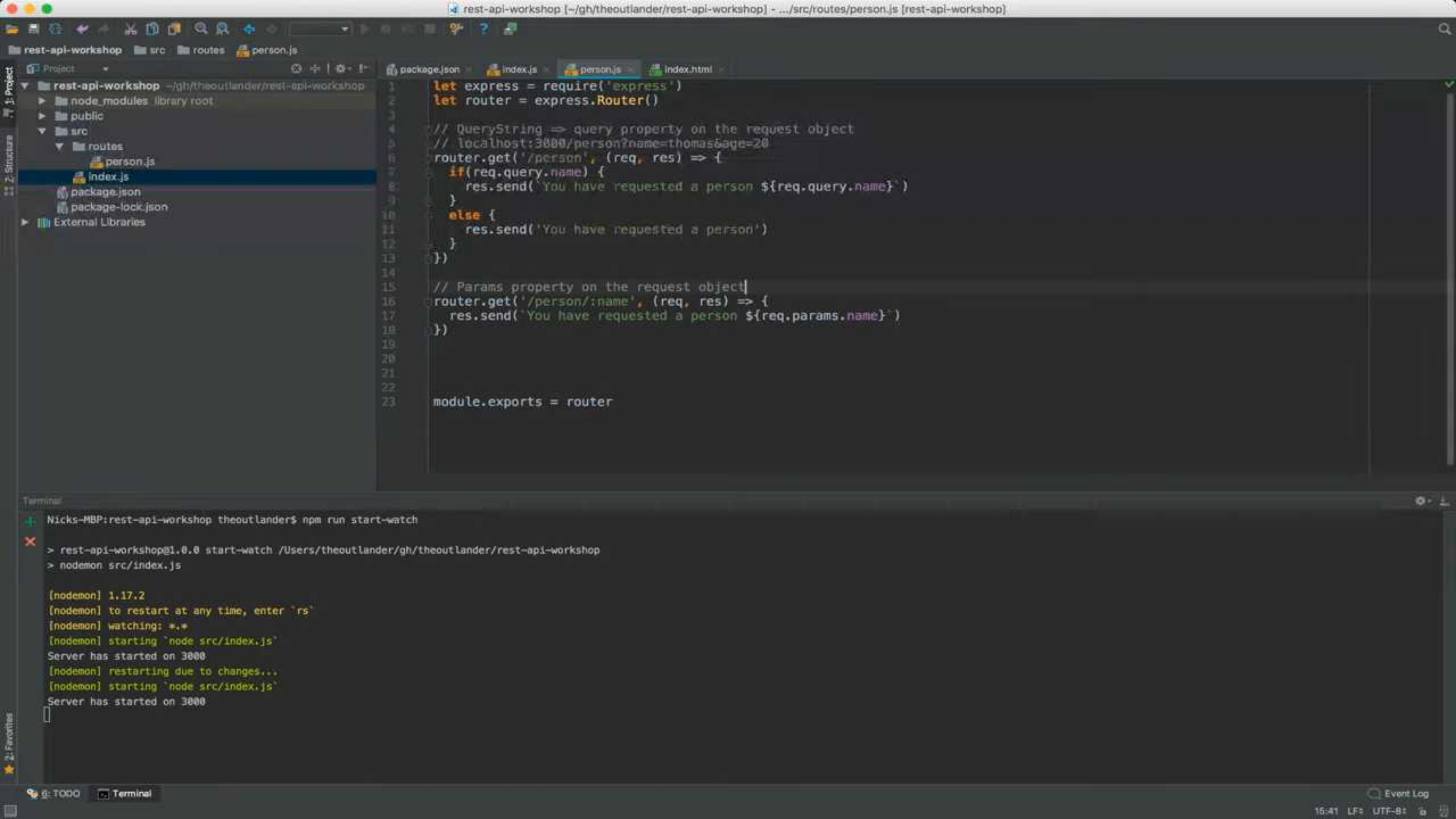














Insomnia

No Environment Cookies

filter

REST API Workshop

GET GET Person

GET GET Person by QueryString

GET GET Person by Name

Projects

GET http://localhost:3000/person/thomas

Send

200 OK


TIME 19.3 ms

SIZE 34 B

ⓘ

Body Auth Query Header Docs

Preview Header Cookie Timeline



Select a body type from above

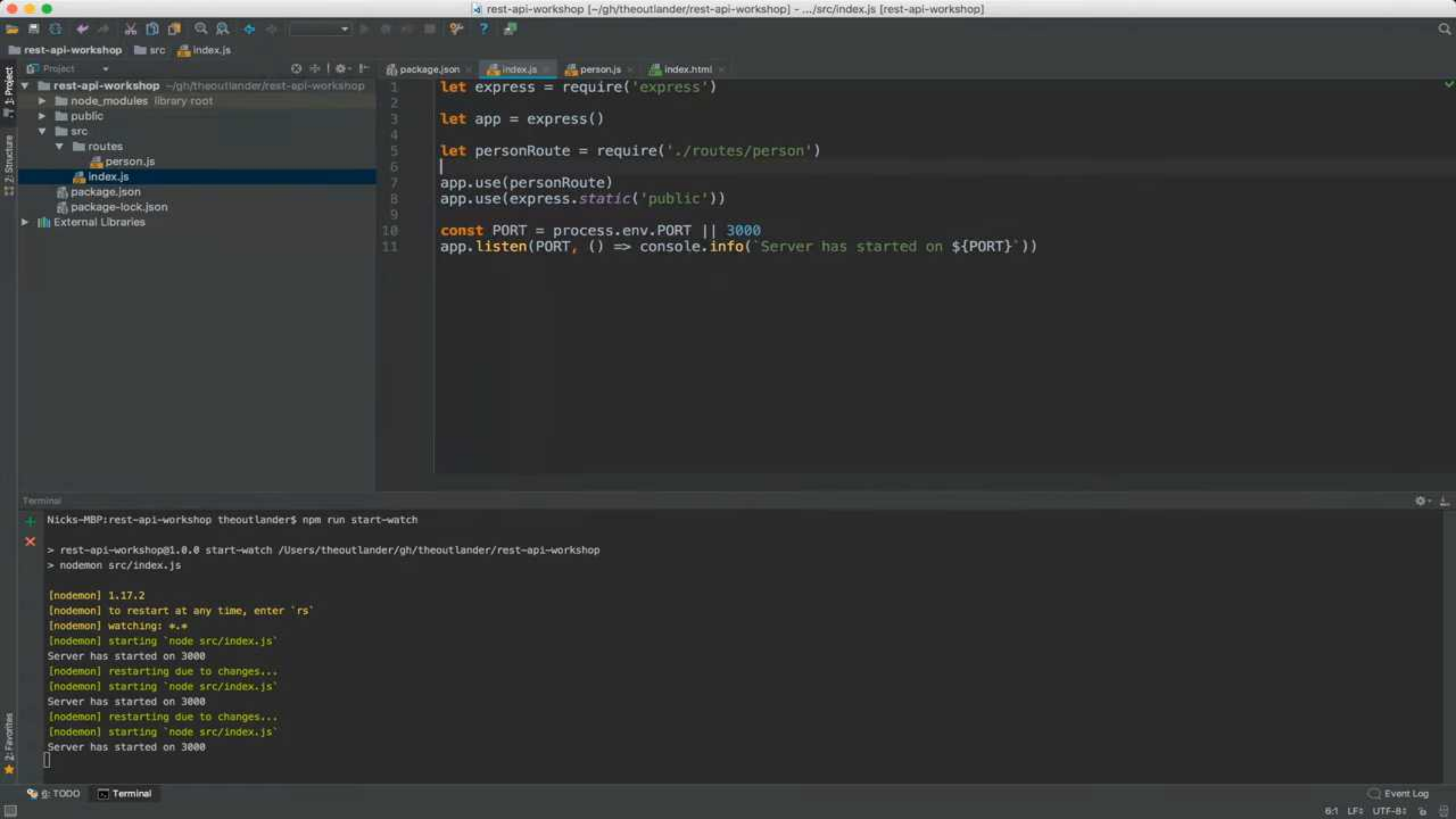
You have requested a person thomas

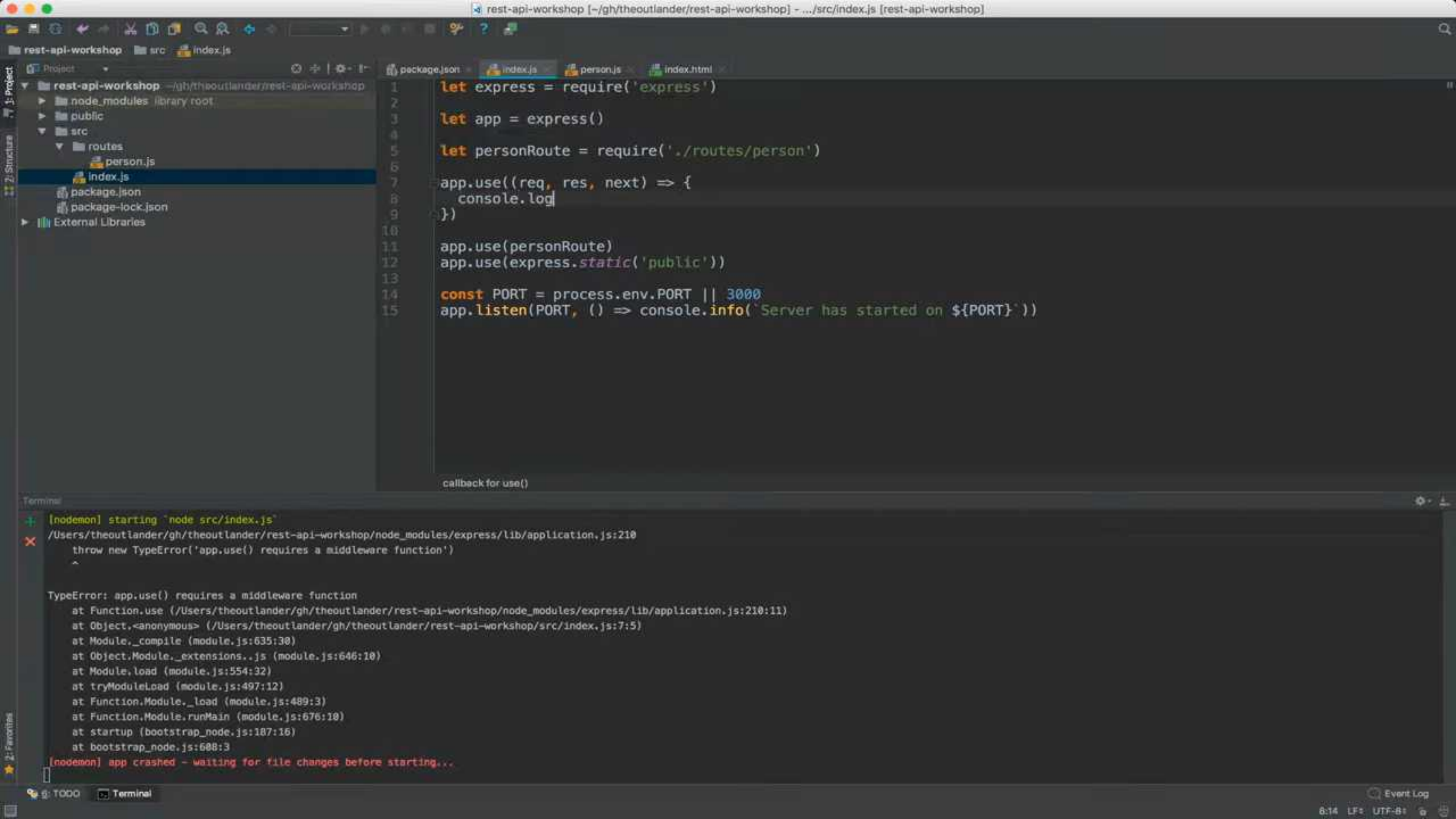
# Middleware

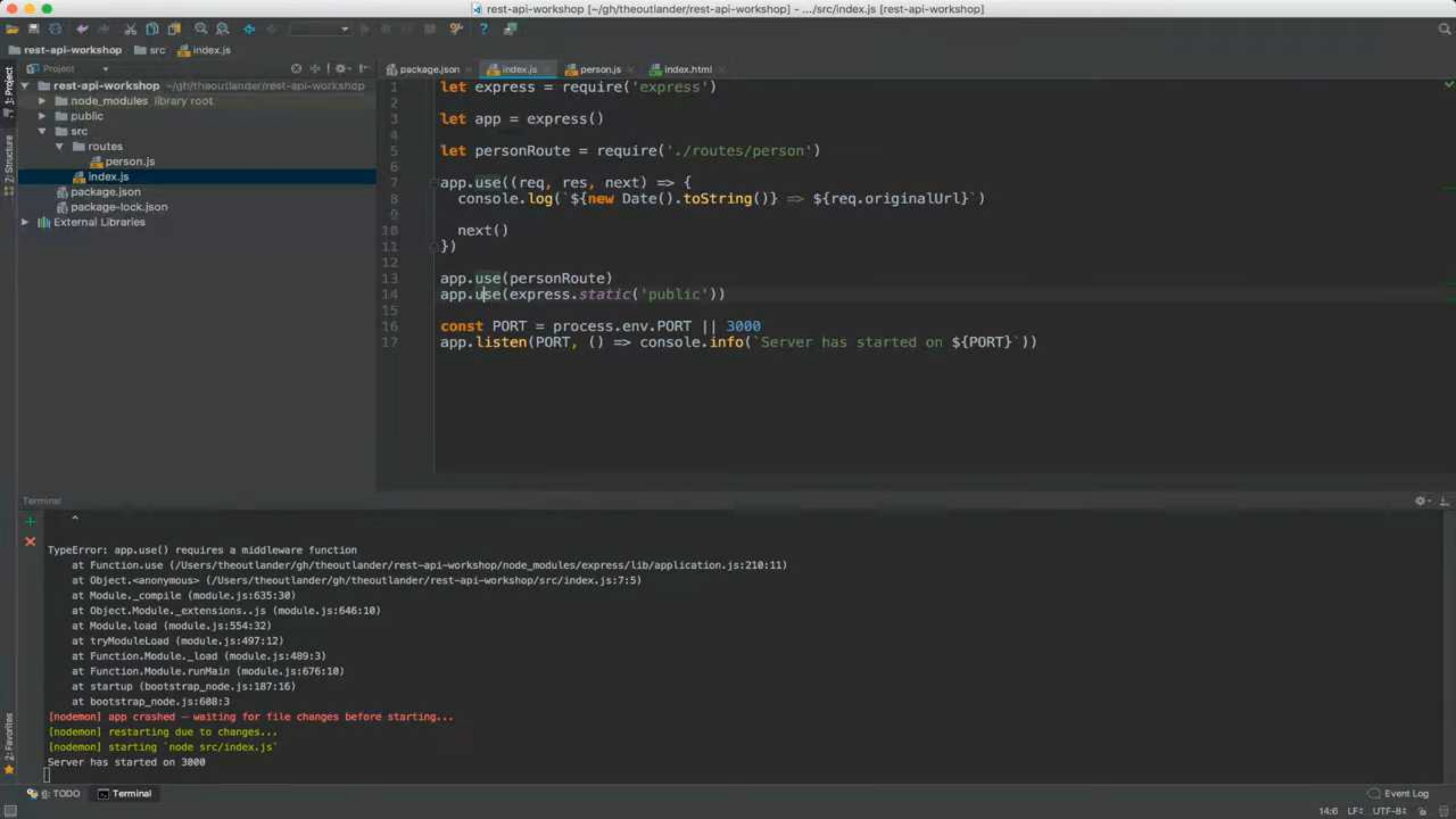
Functions that execute serially

- Access to request/response objects and the next middleware function in the pipeline
- Execute any code
- Make changes to the request and the response objects
- End the request-response cycle
- Call the next middleware function in the stack

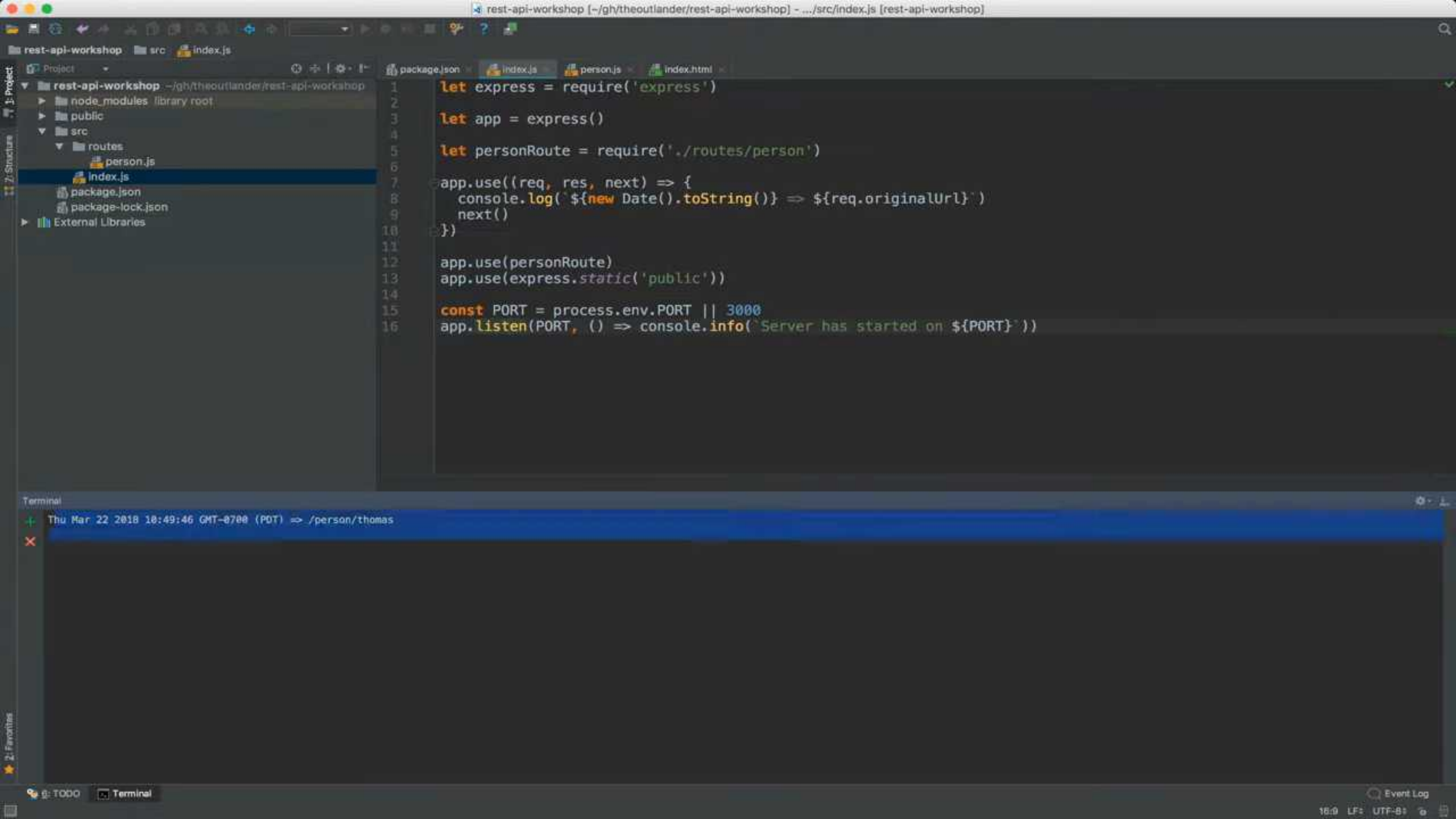










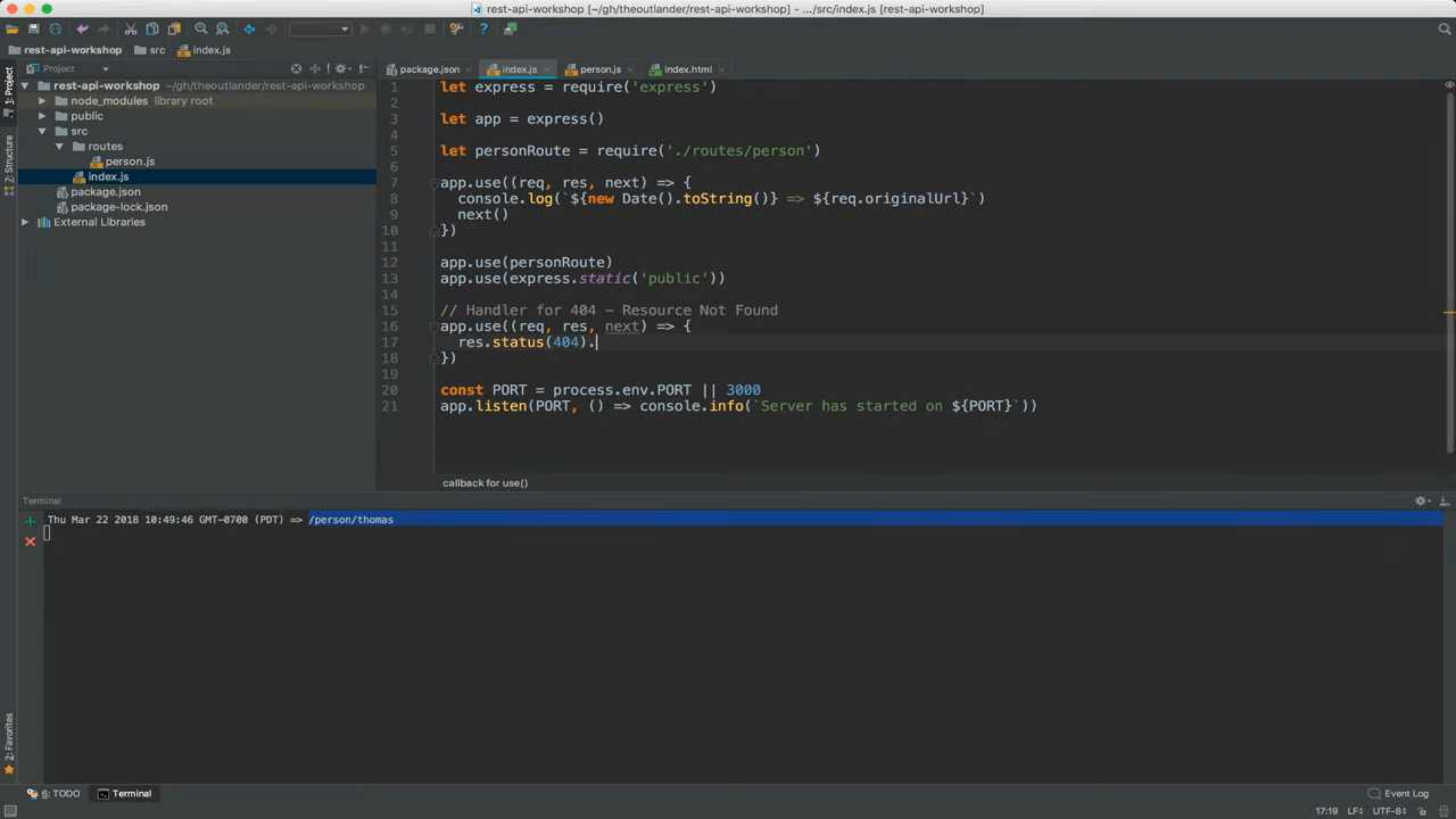


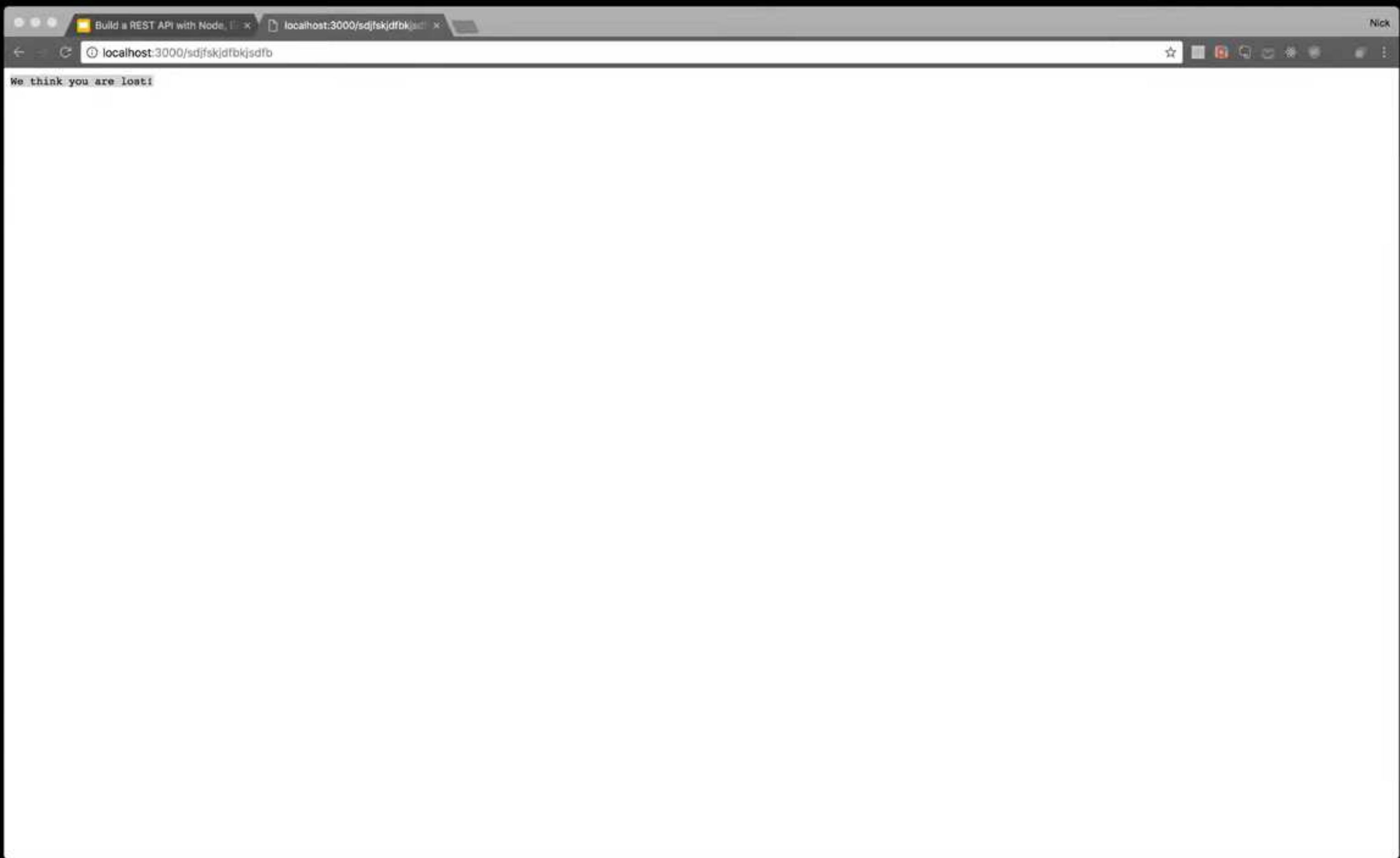


# Error Pages

Custom Handlers

- 404 Not Found Handler
- 500 Internal Server Error





Build a REST API with Node, Express, MongoDB, REST Client, Application Setup, Express, Serve Static Content, Route, Middleware, Error Pages

Secure [https://docs.google.com/presentation/d/1VSlo9JEsoVjNNH4DqQtSWlnKIs5p39ImHyAcLifSw/edit#slide=id.g361772e256\\_1\\_57](https://docs.google.com/presentation/d/1VSlo9JEsoVjNNH4DqQtSWlnKIs5p39ImHyAcLifSw/edit#slide=id.g361772e256_1_57)

18 B I U A

# Error Pages

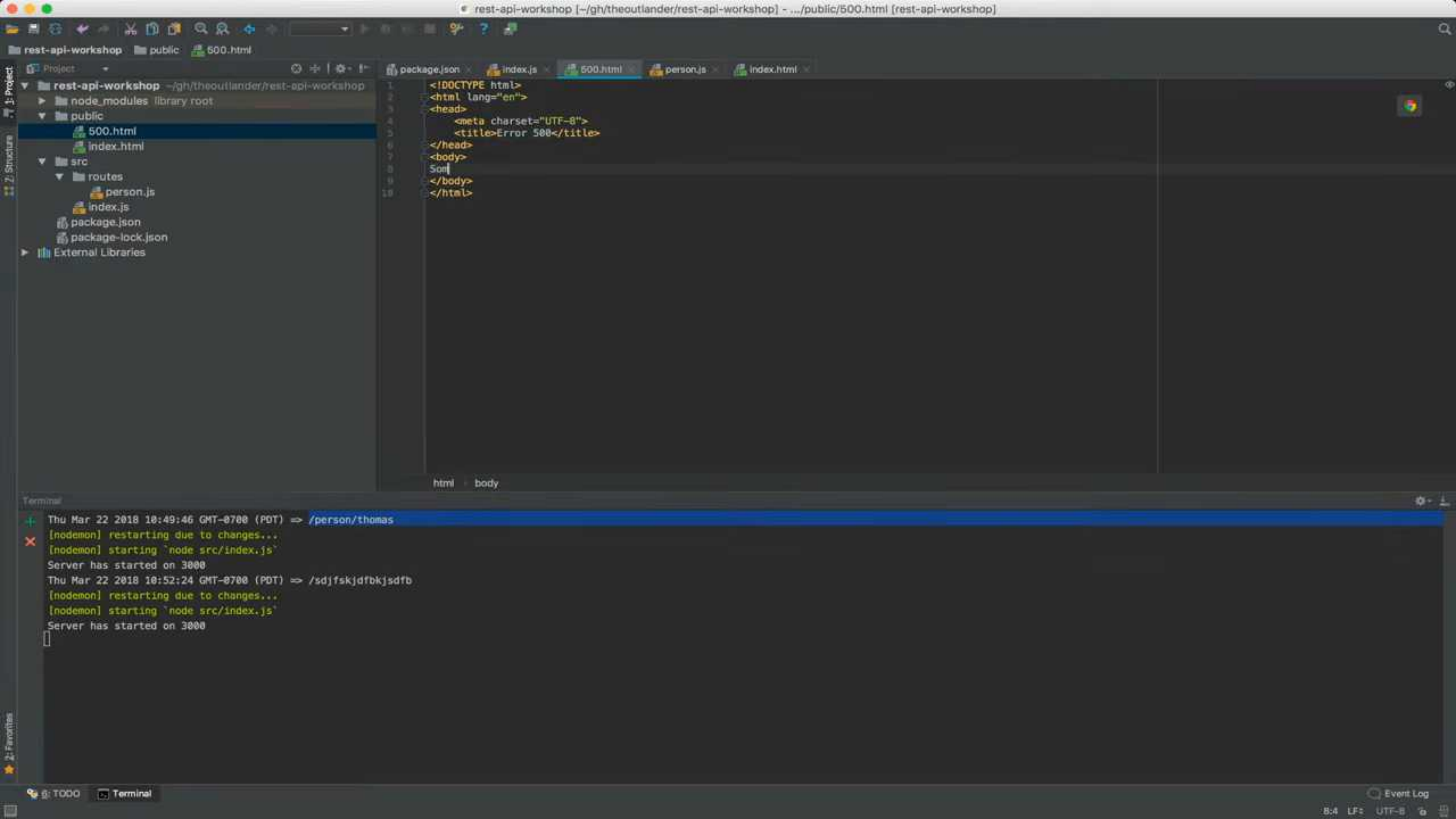
## Custom Handlers

- 404 Not Found Handler
- 500 Internal Server Error

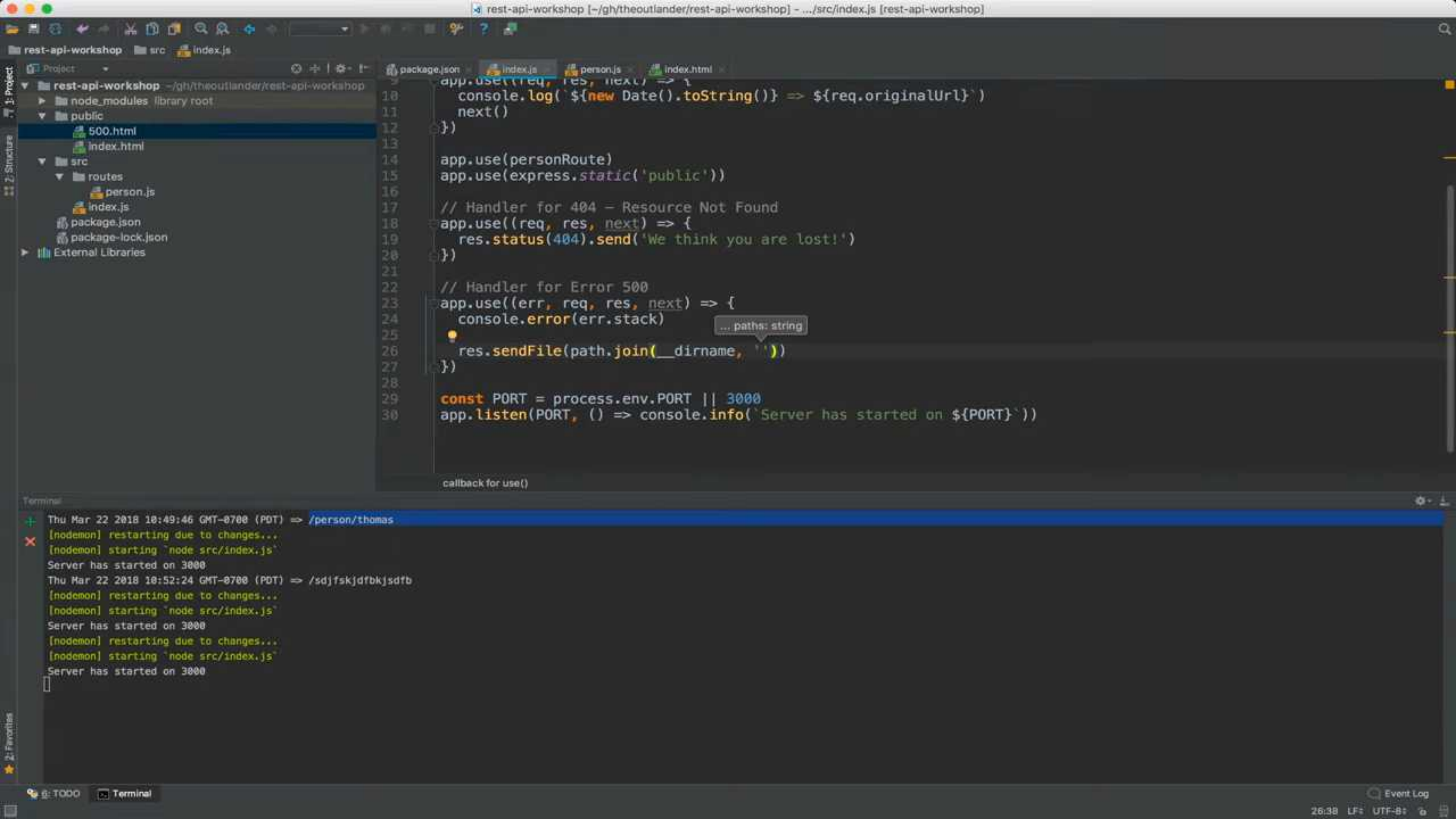
20

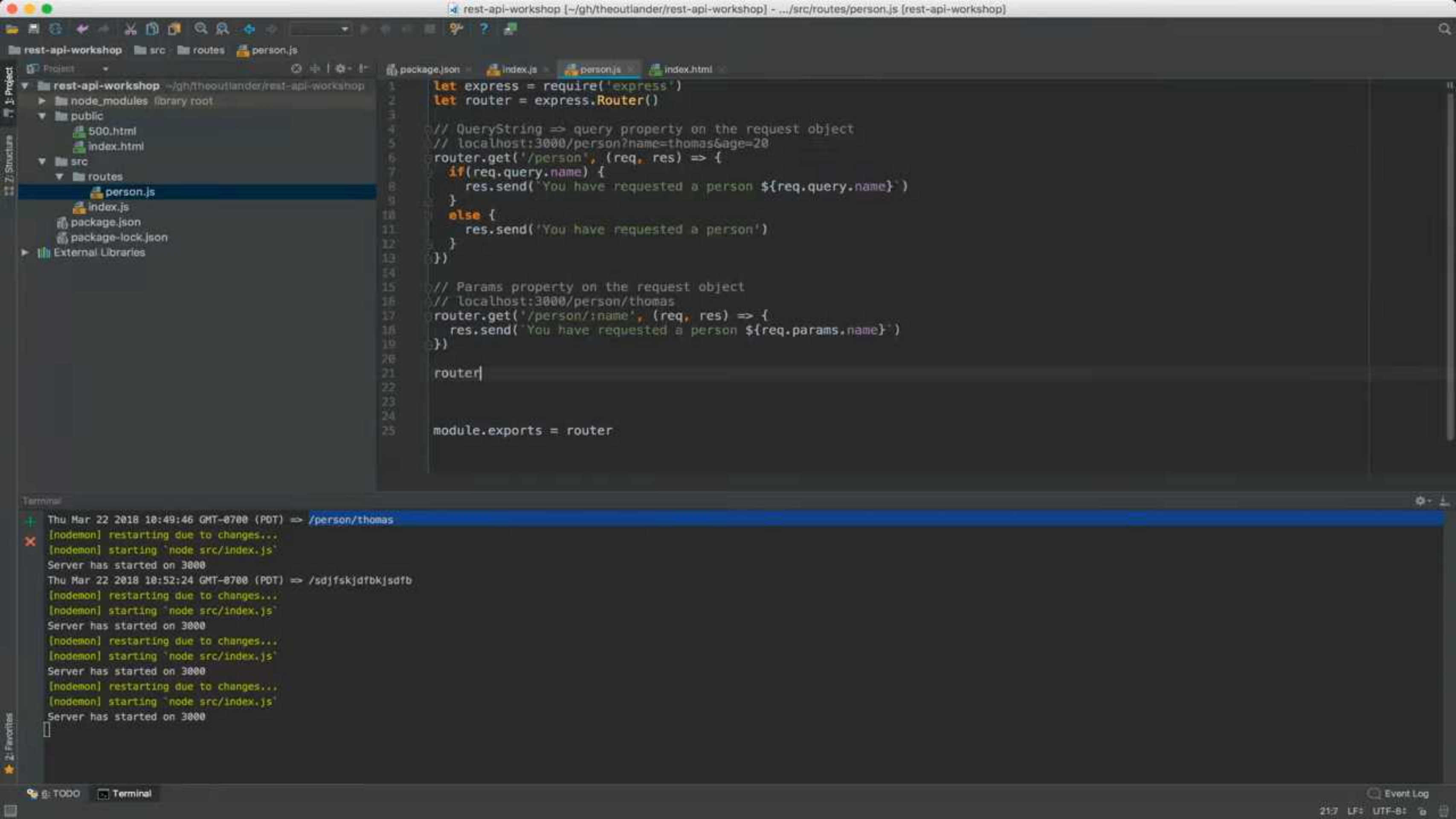
Click to add speaker notes

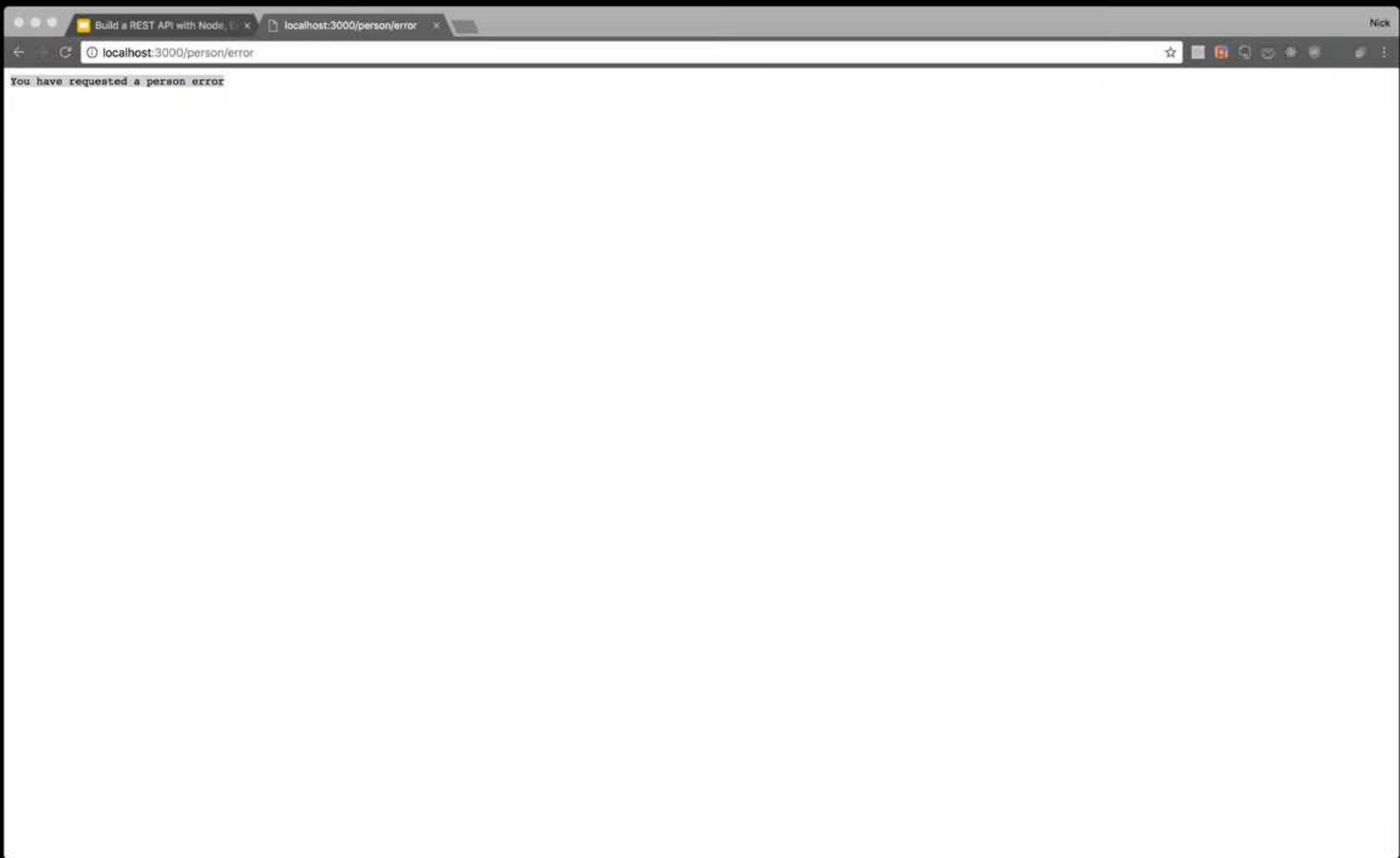
Explore



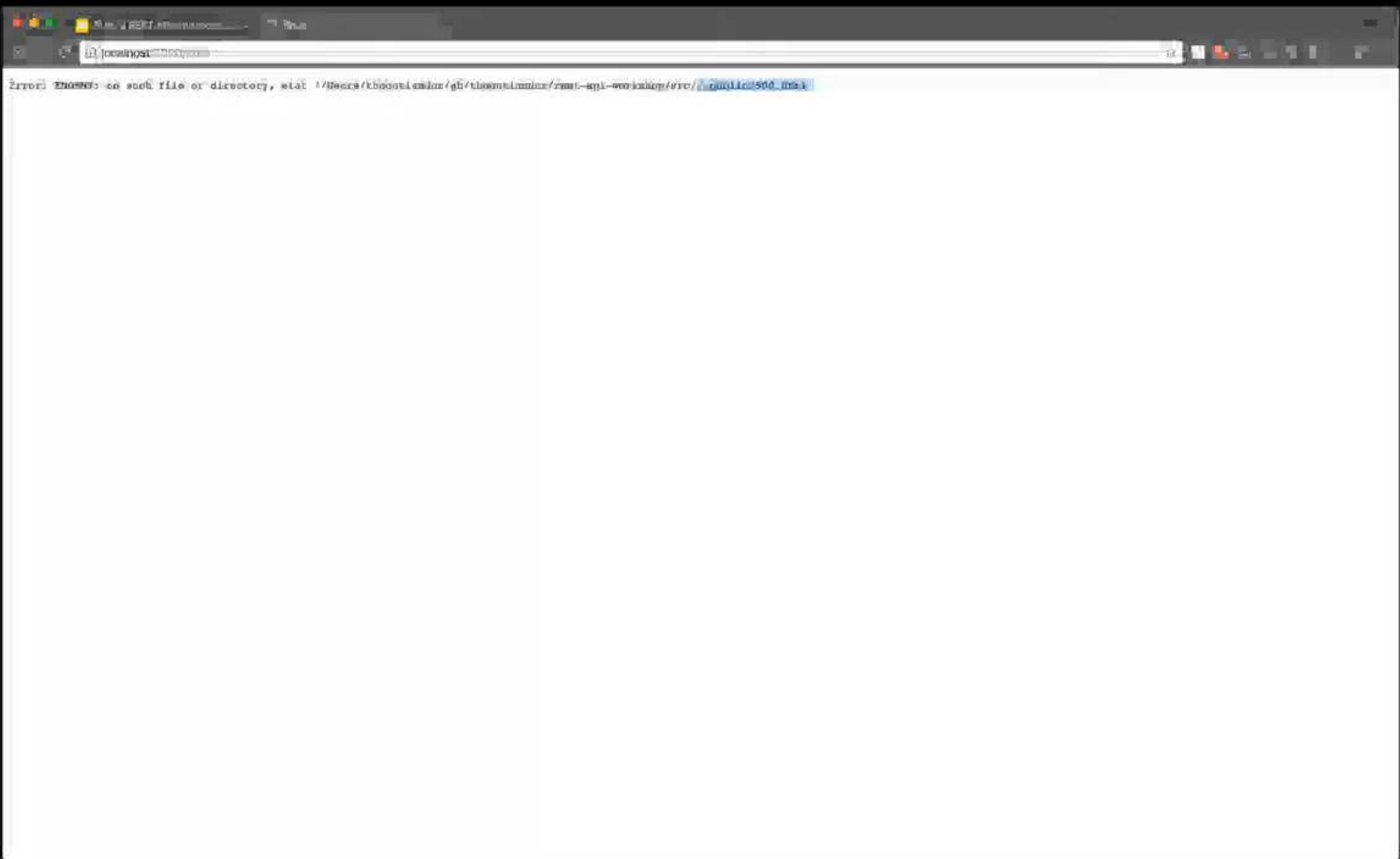












# Mongoose

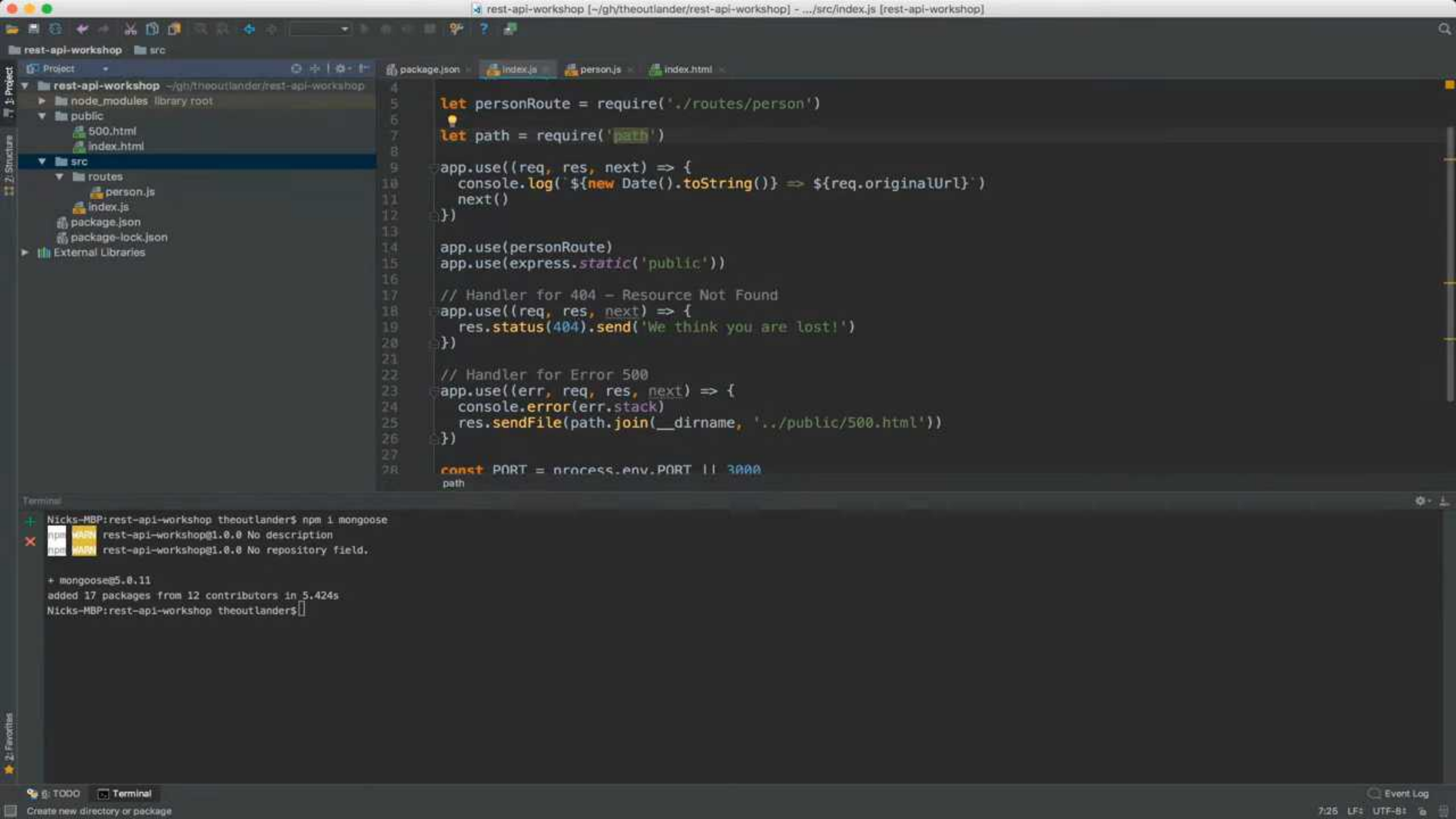
For MongoDB

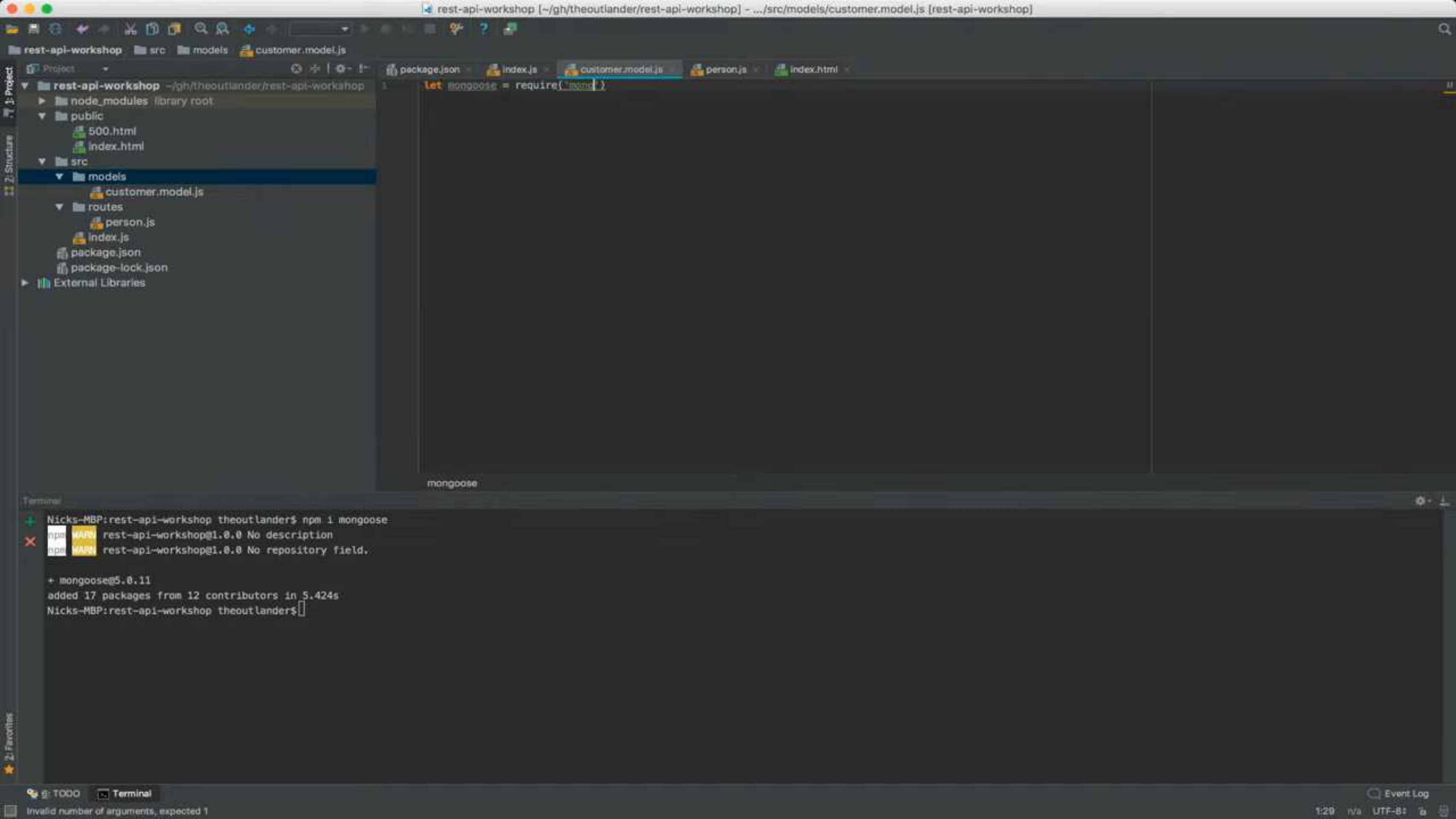
- Install dependency
- Reference Mongoose
- CRUD API

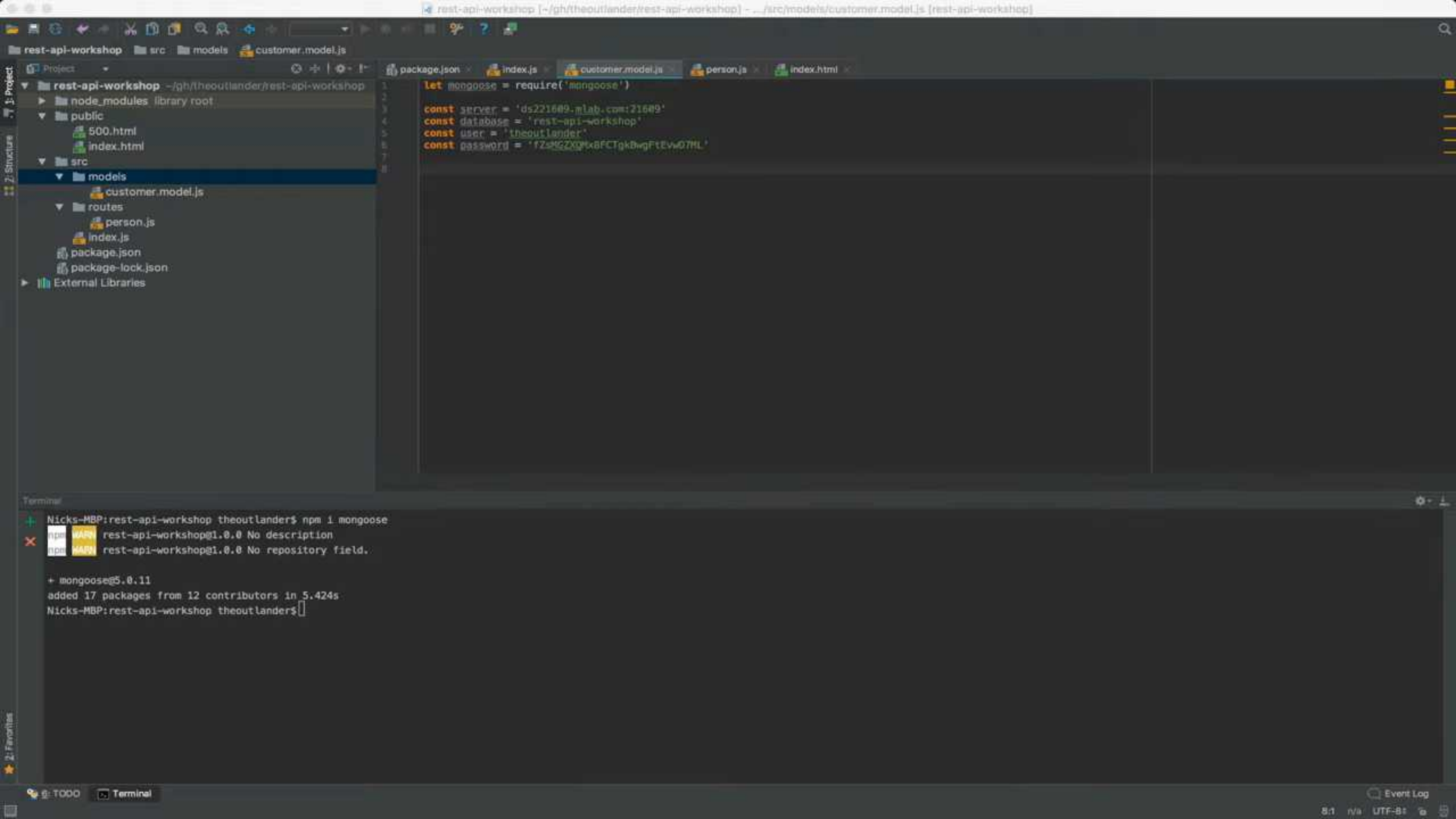
# Mongoose

For MongoDB

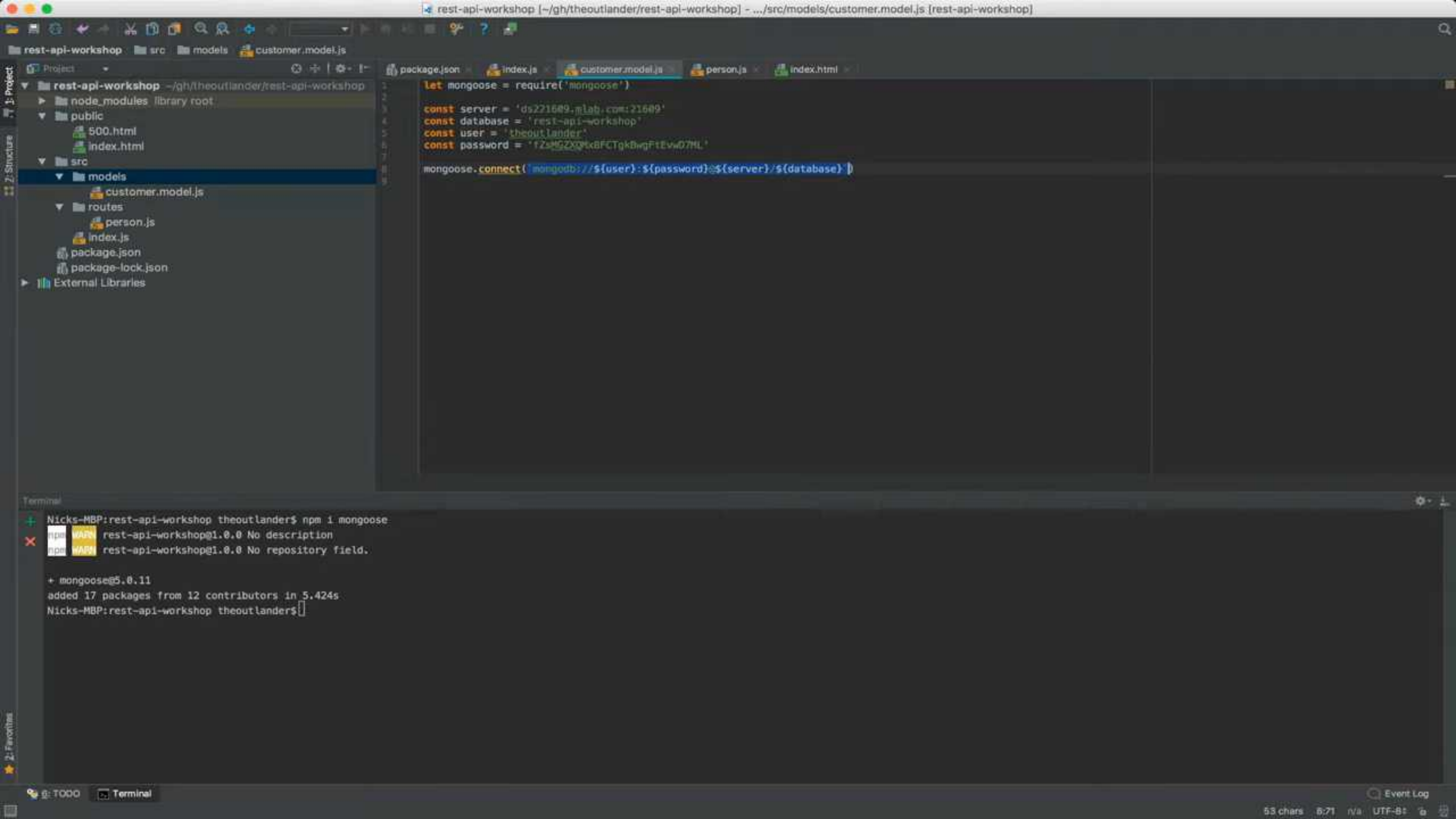
- Install dependency
- Reference Mongoose
- CRUD API

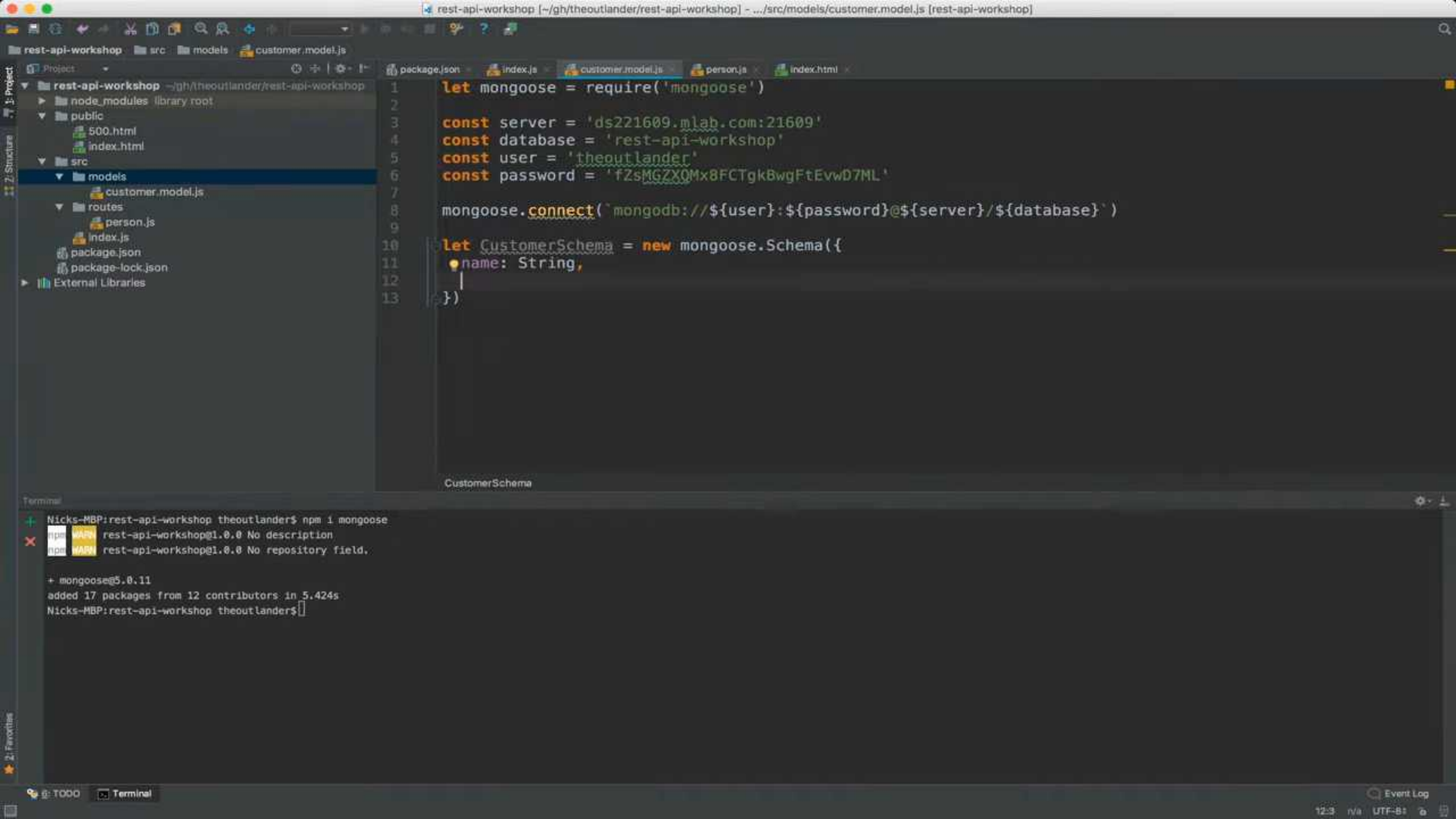




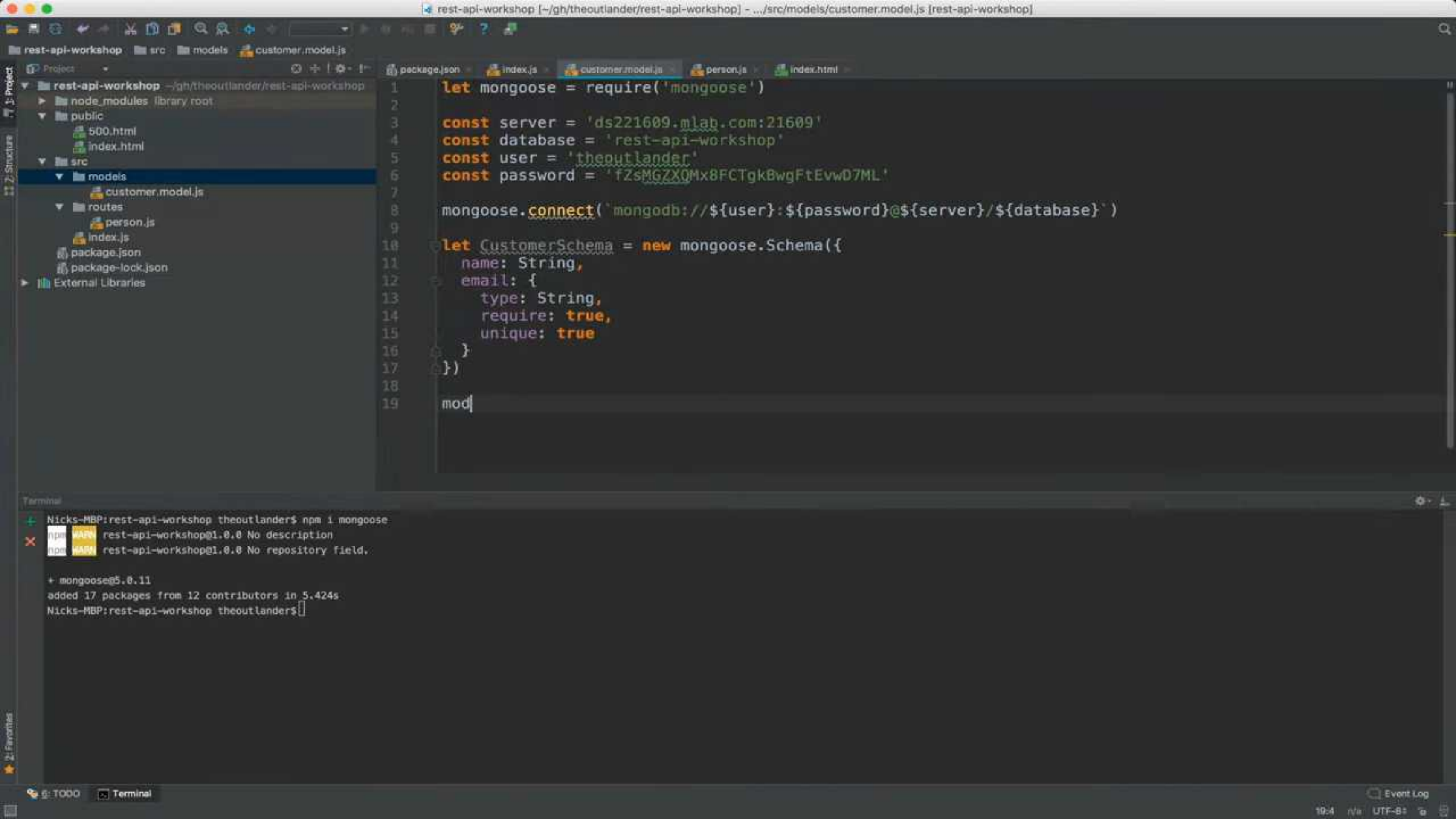


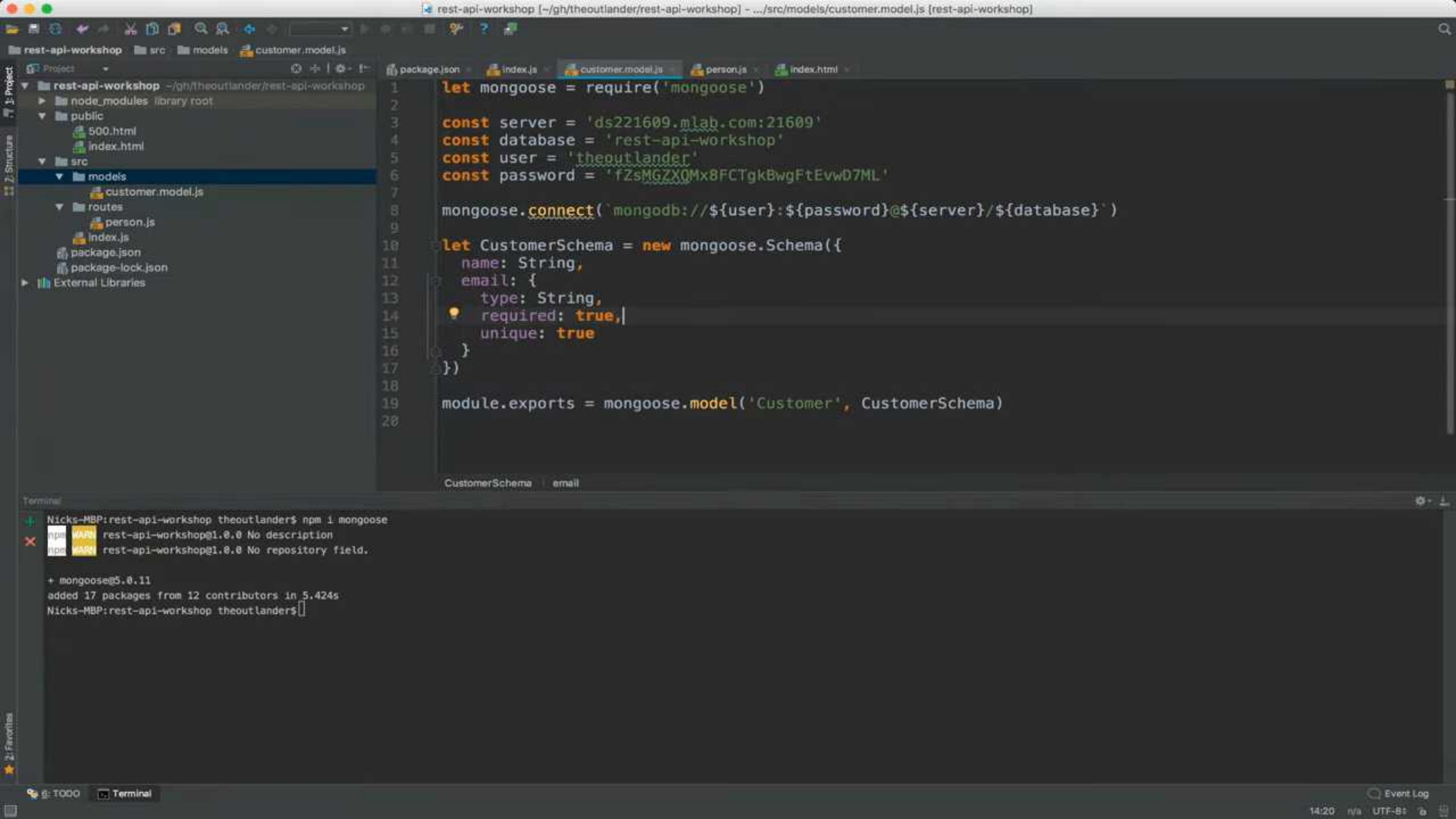


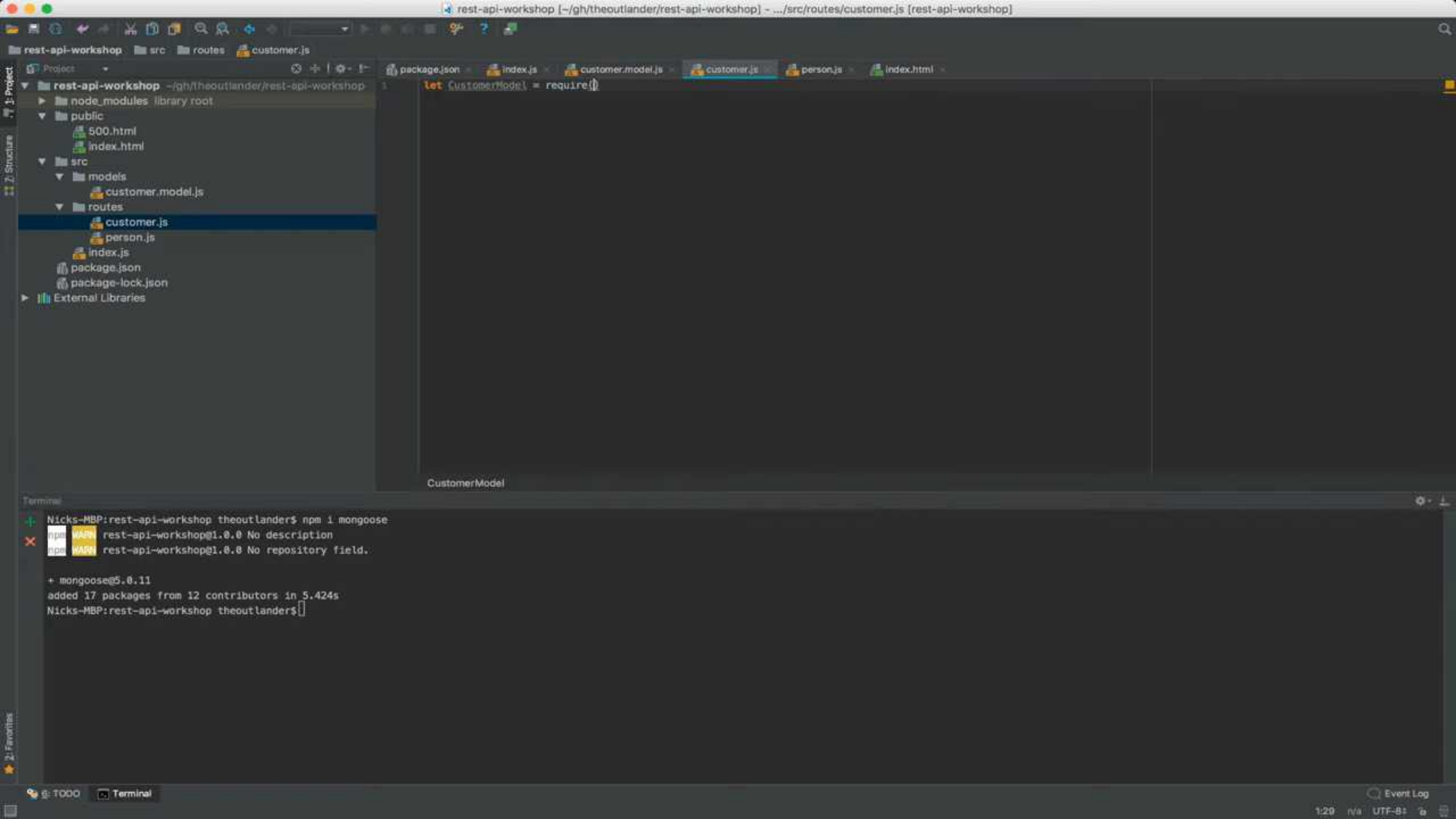


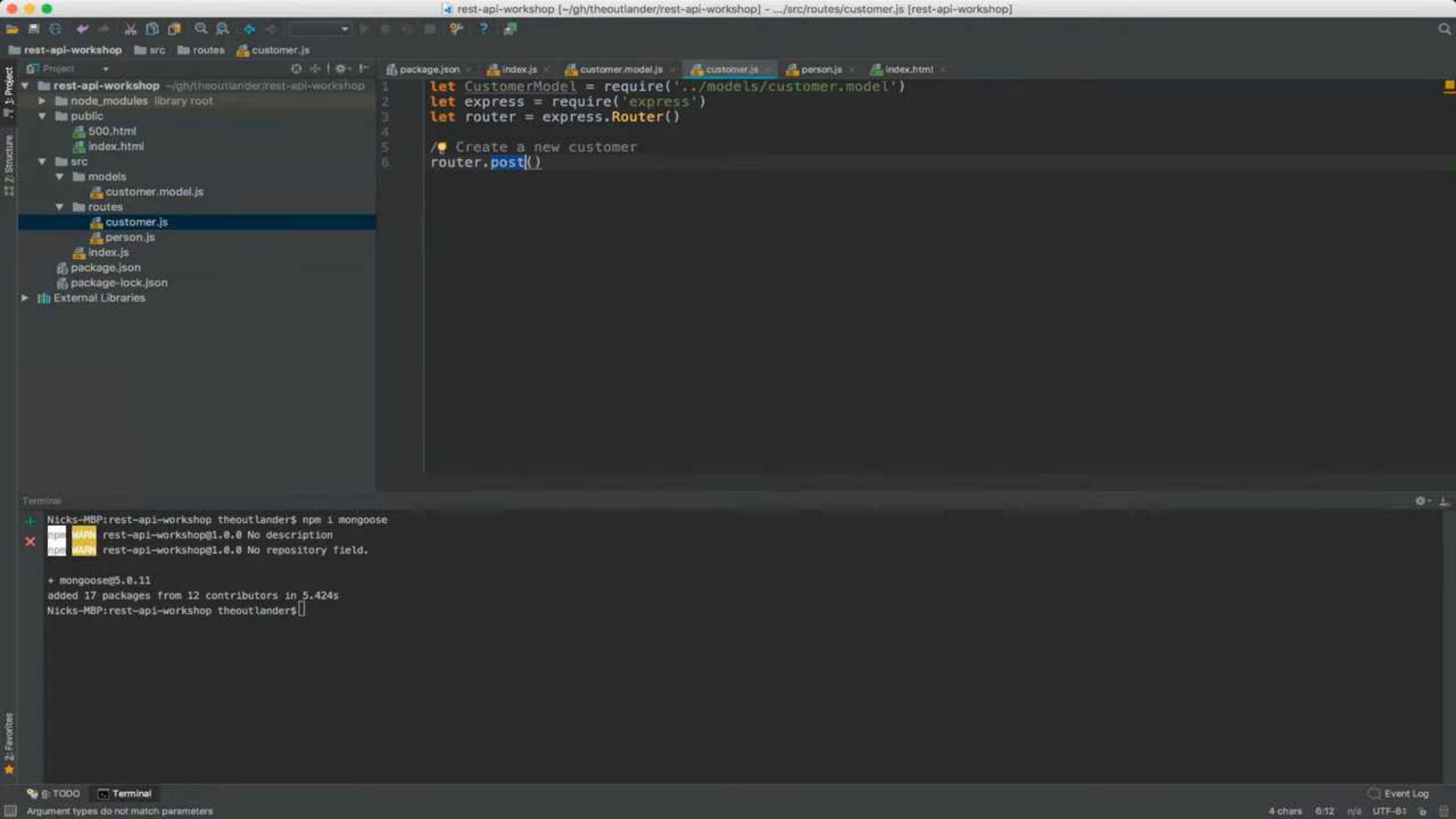




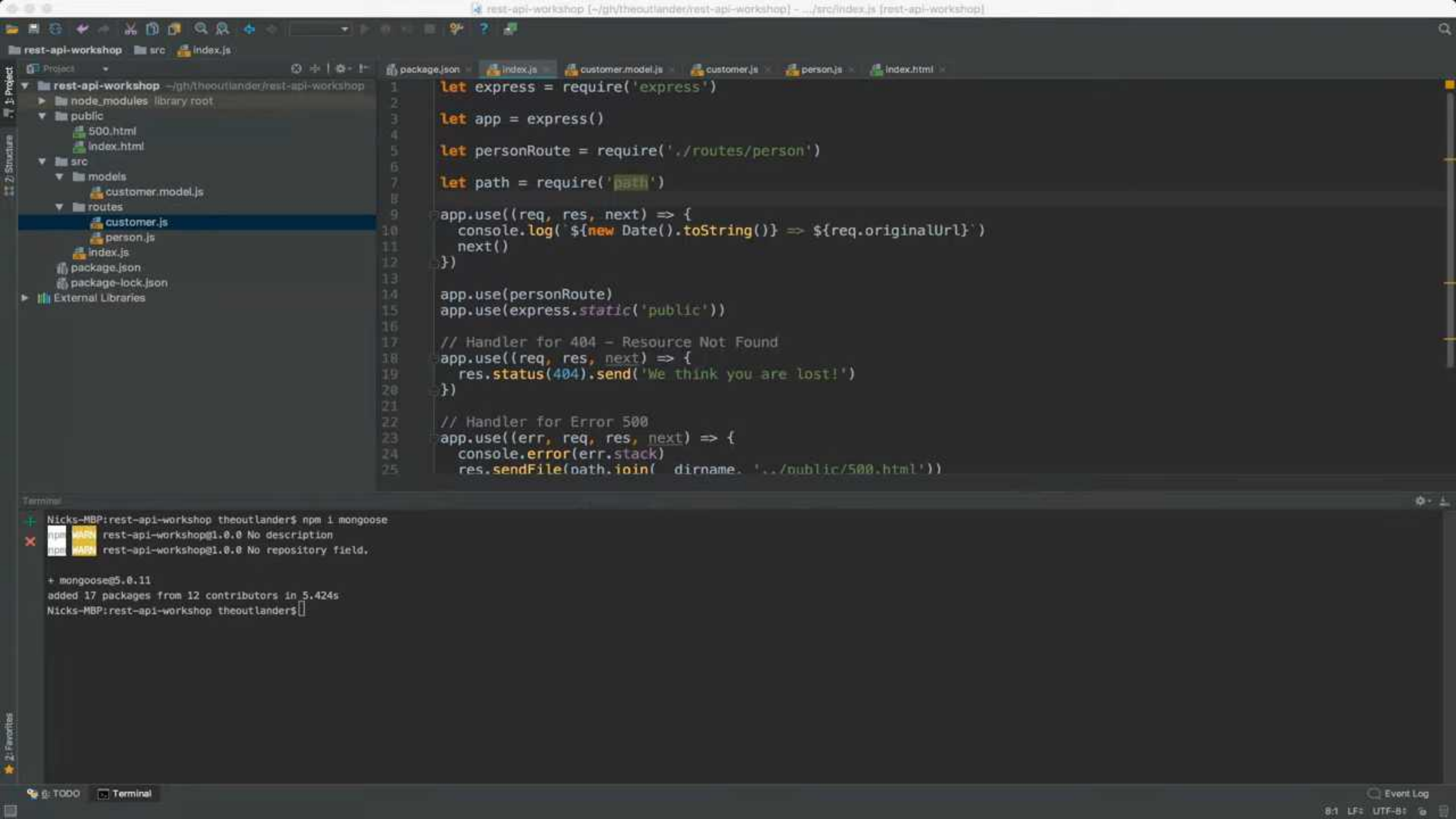


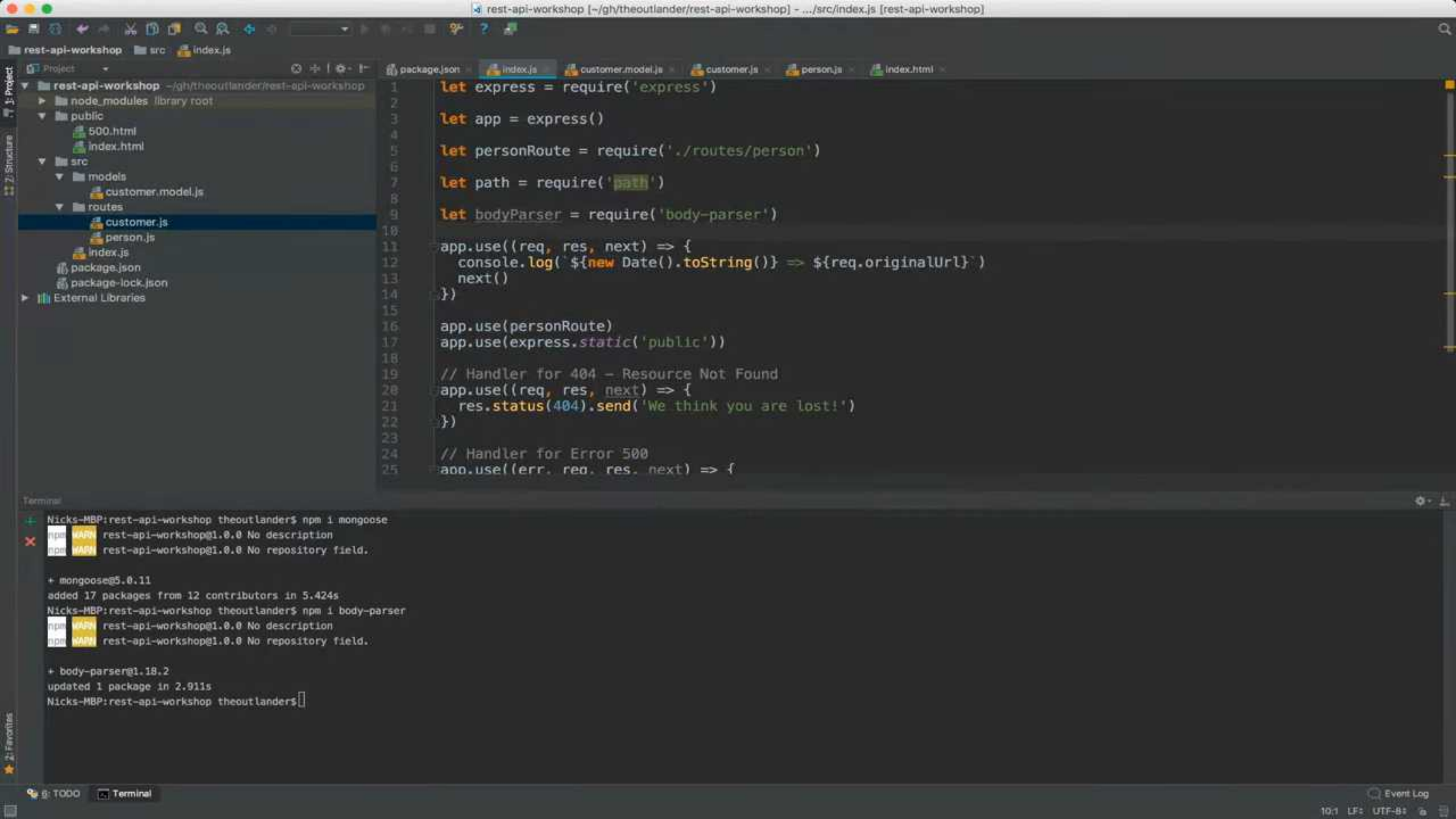


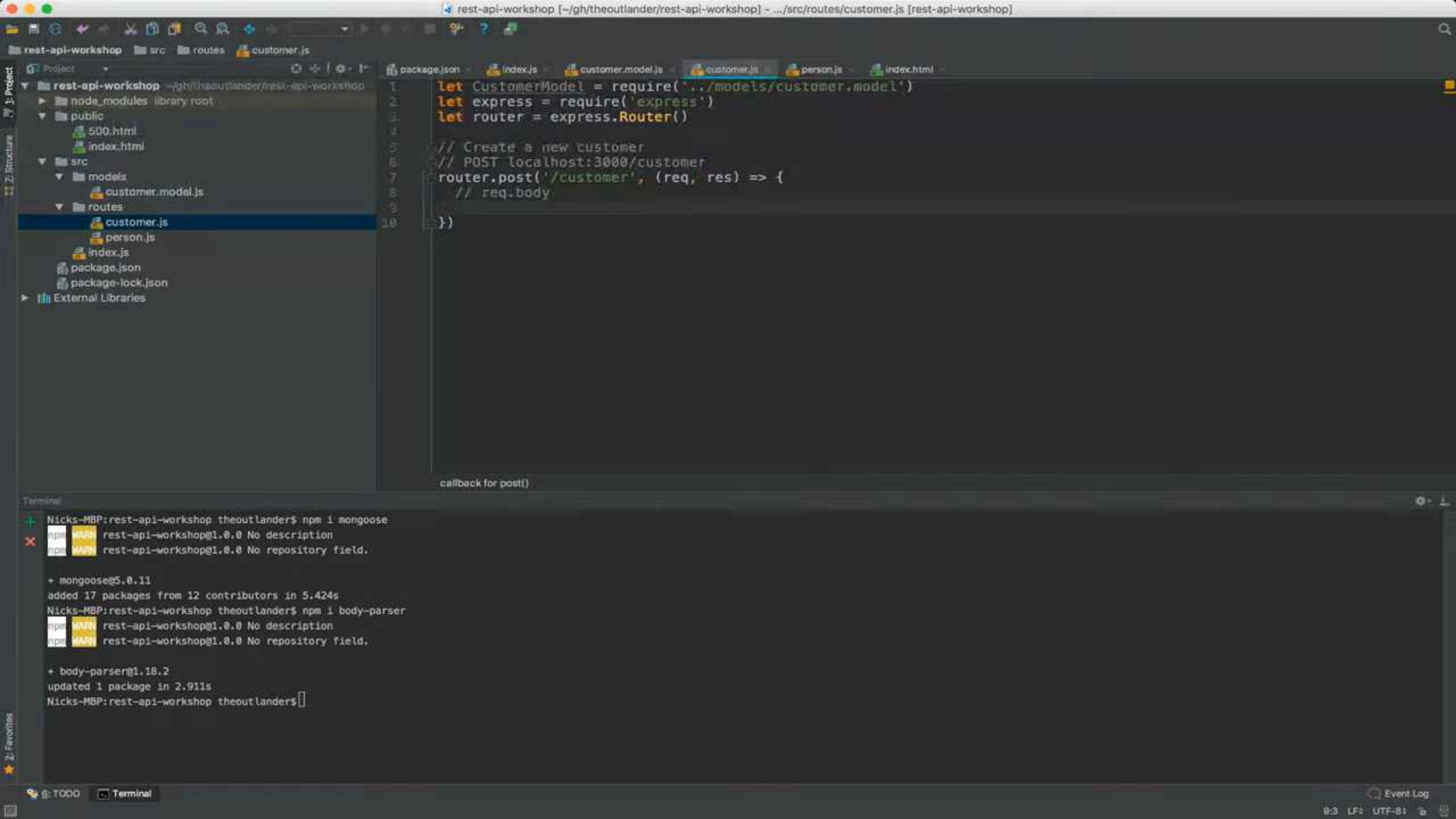




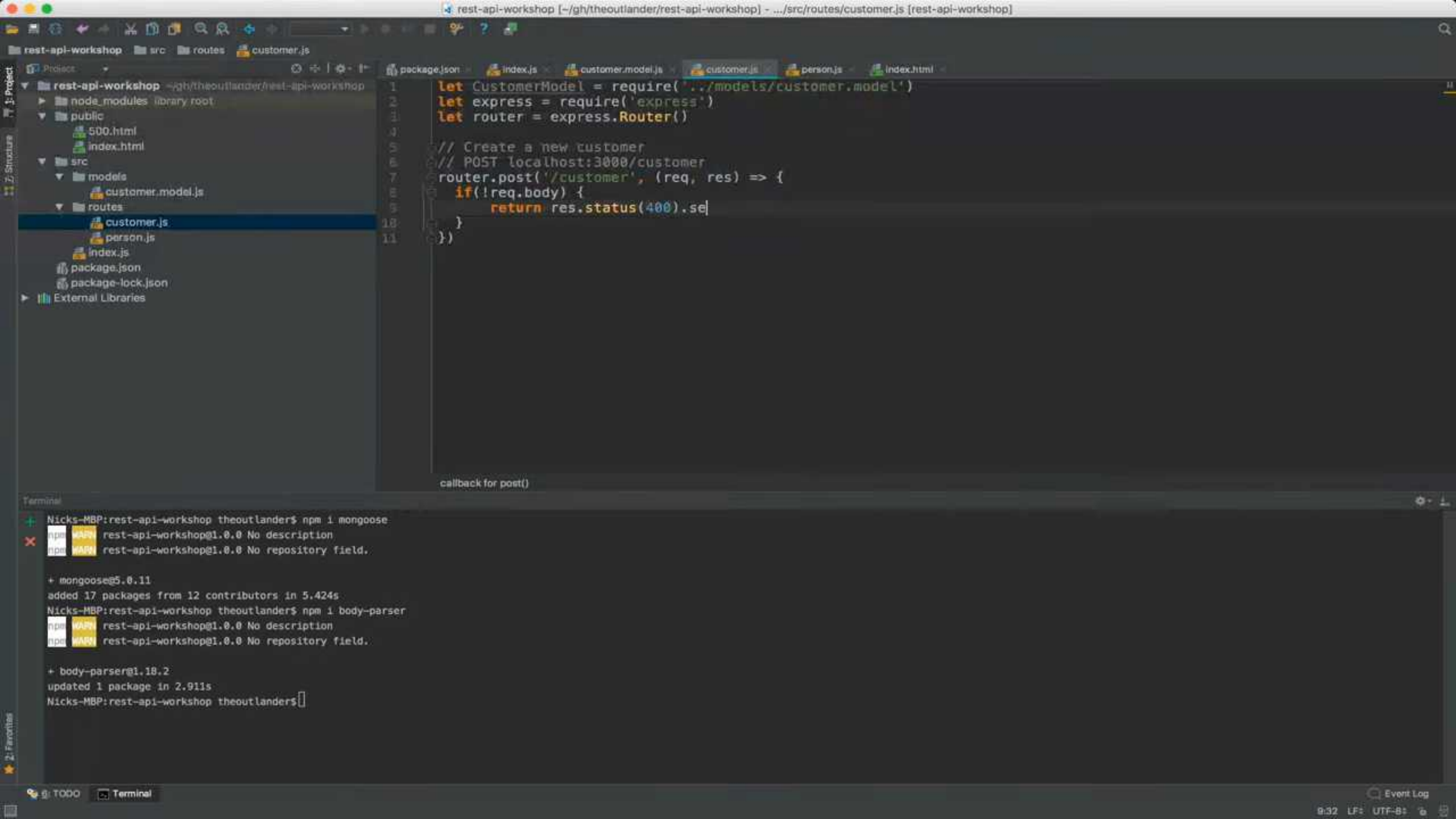




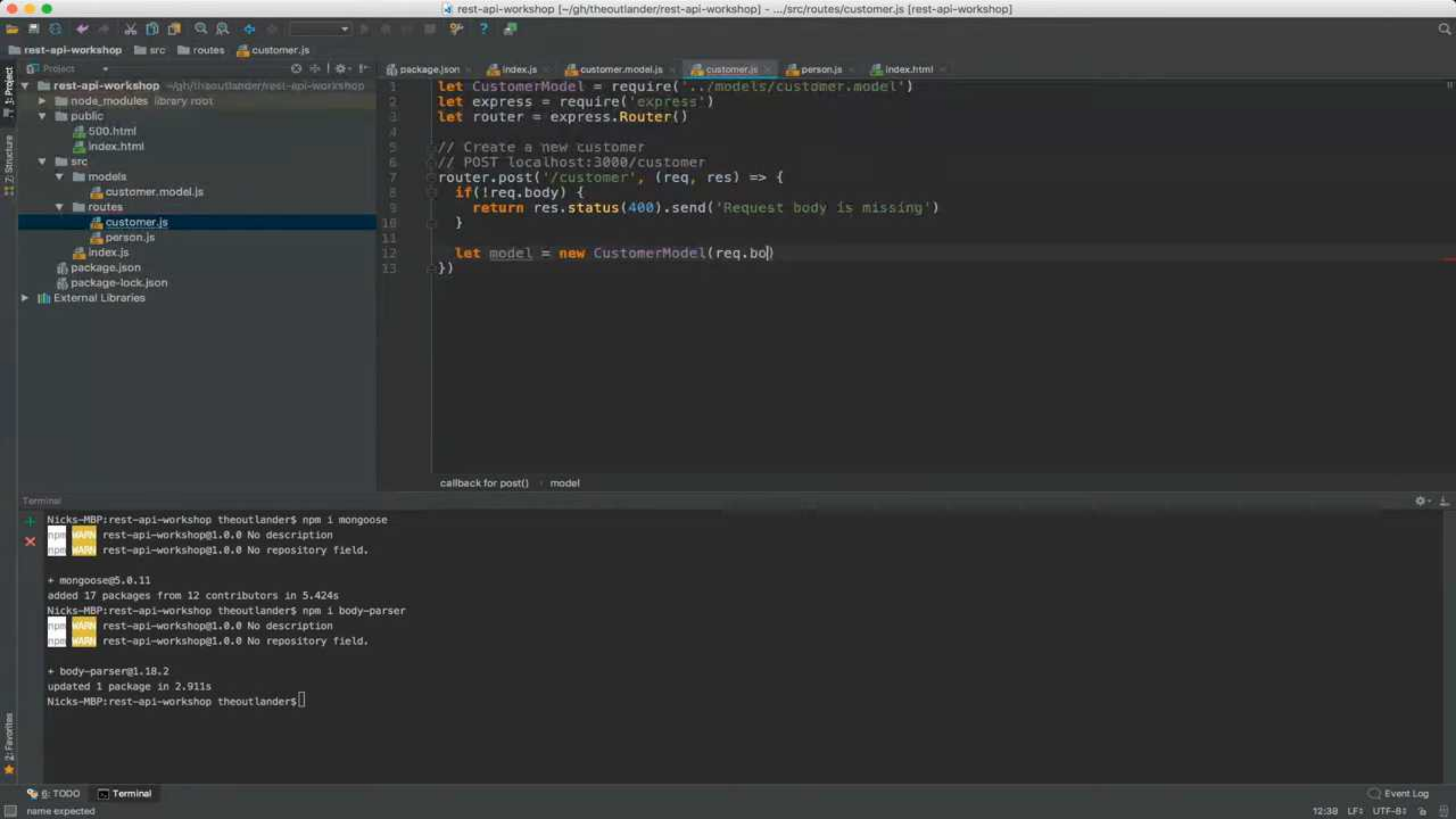


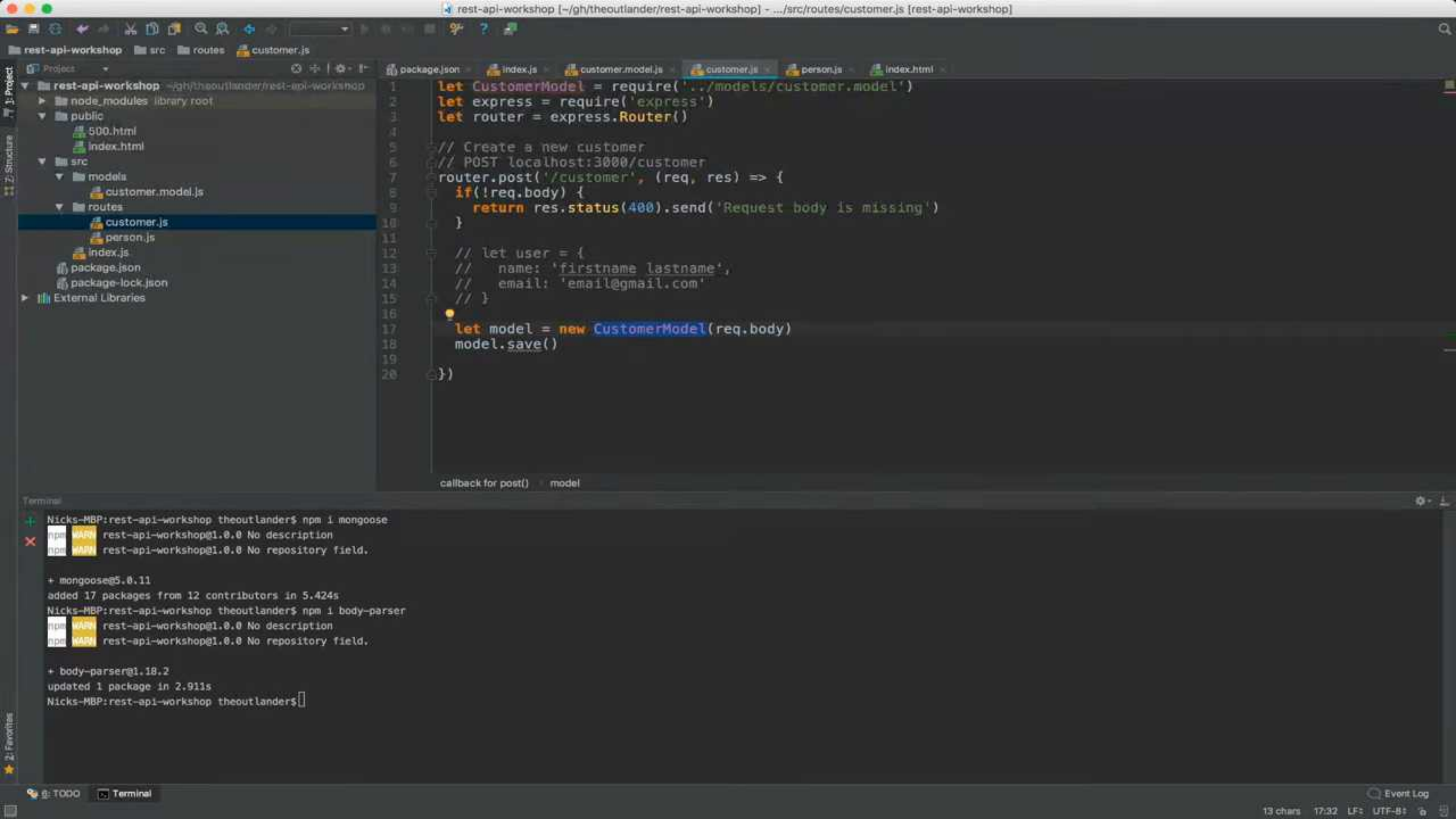


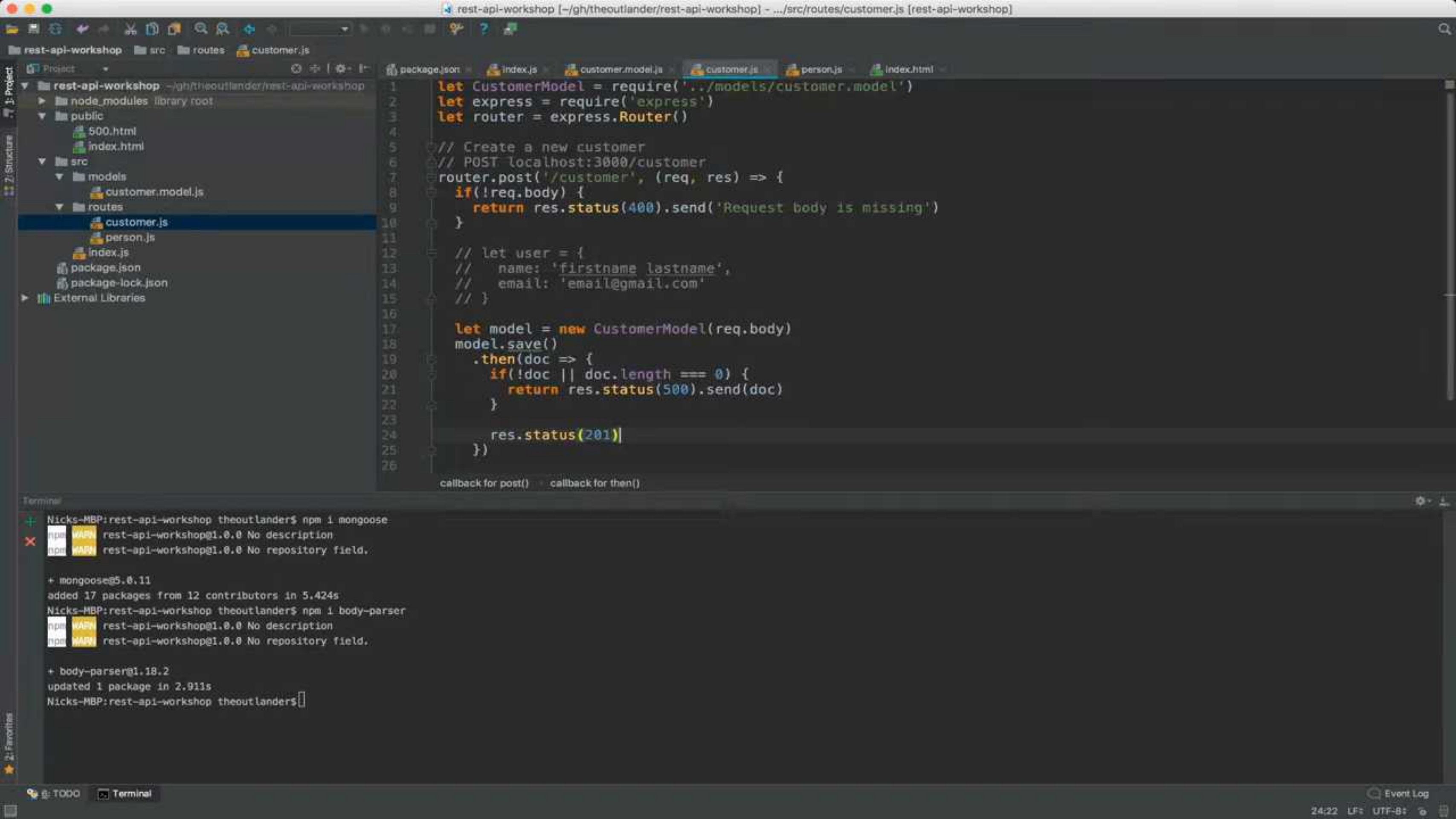




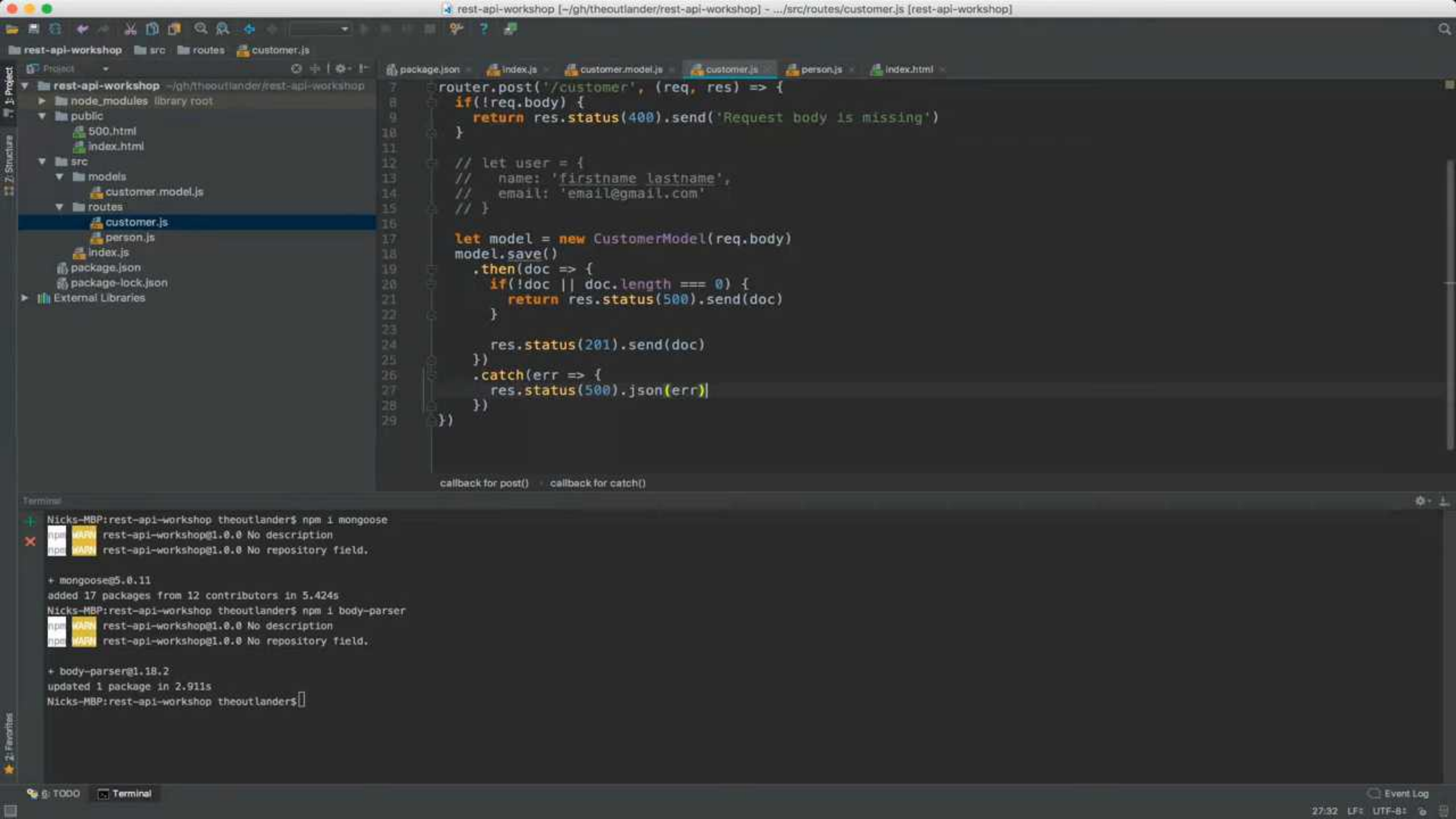


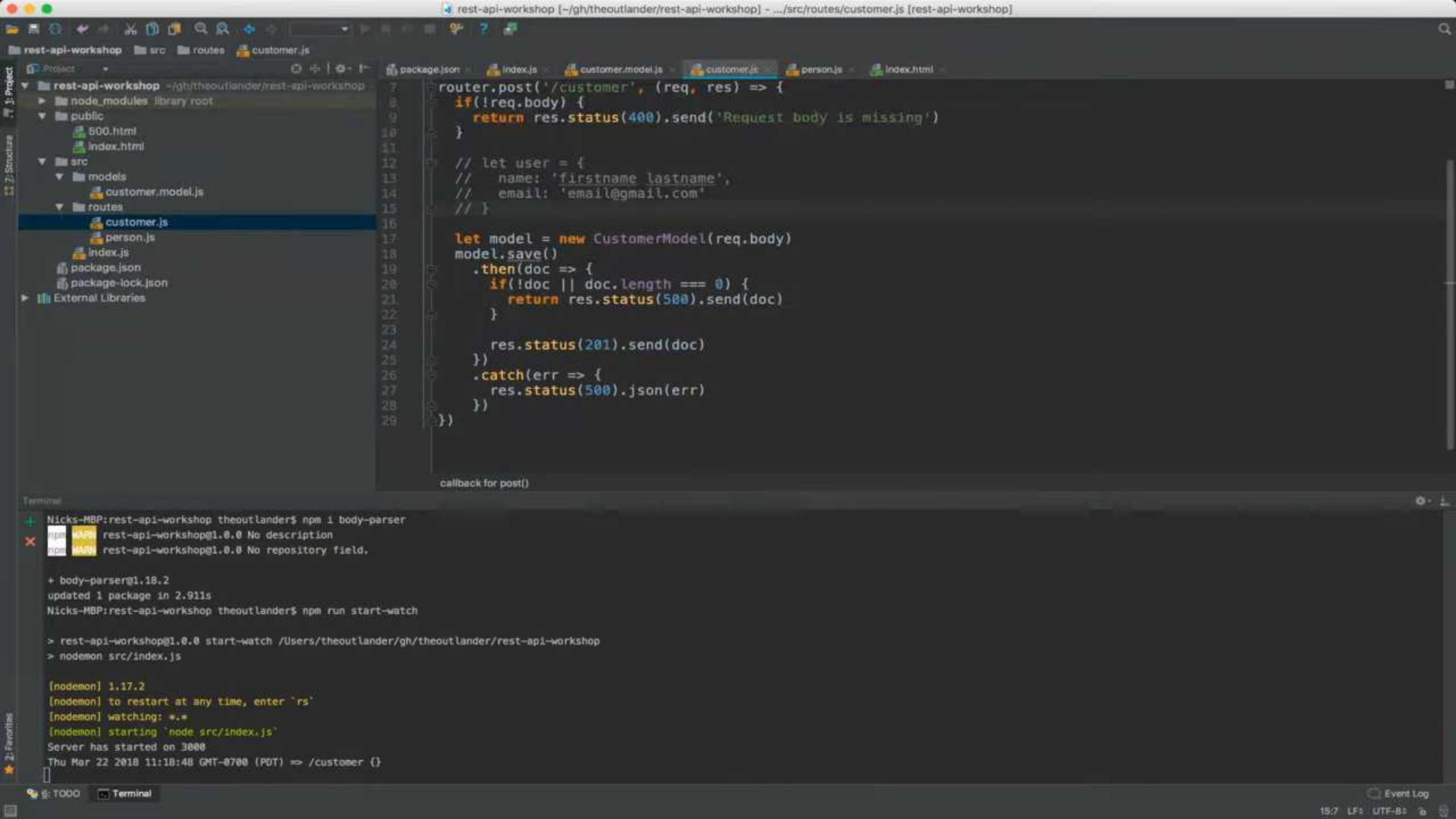




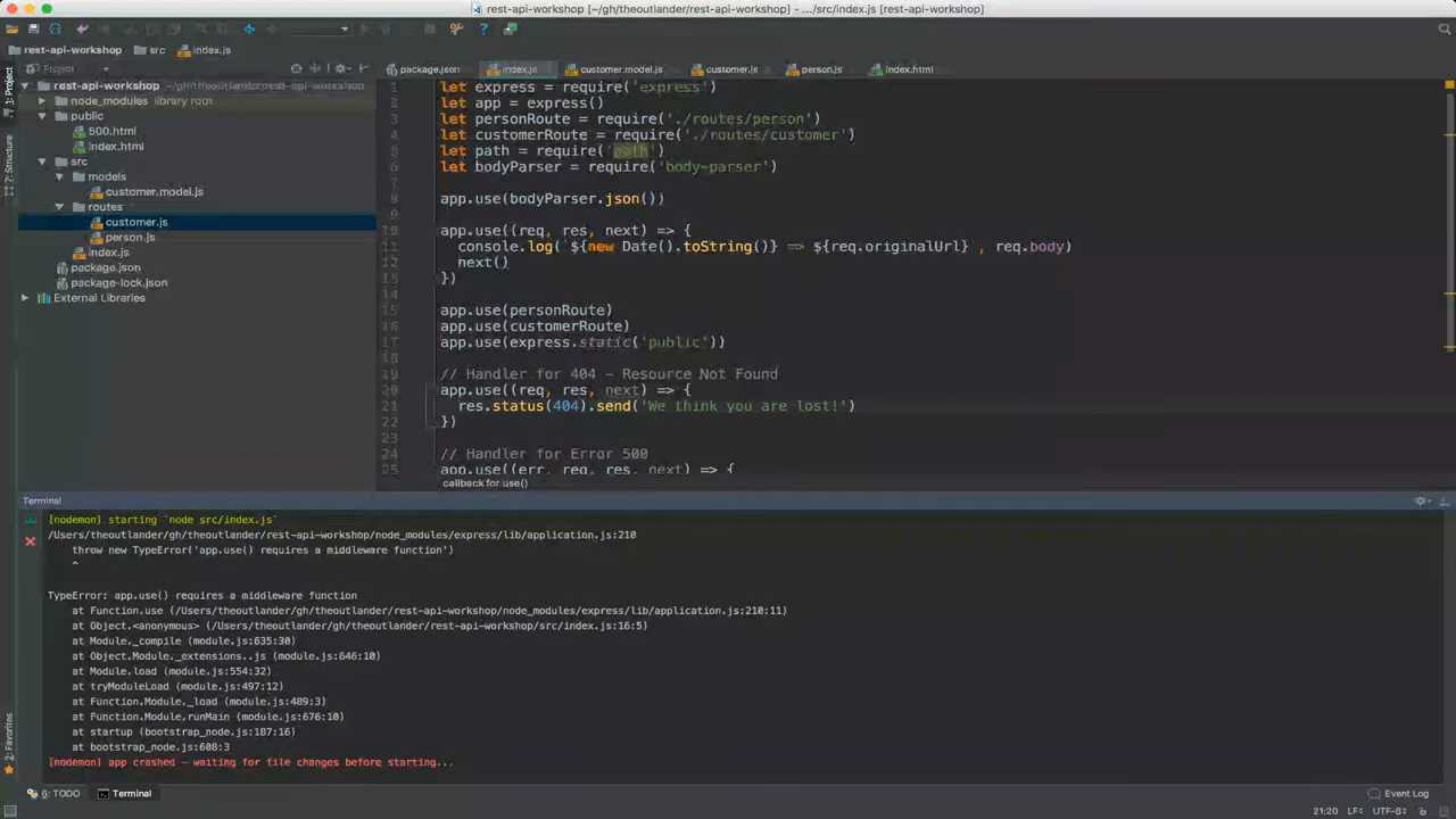
















Insomnia

No Environment Cookies

Filter

REST API Workshop

POST POST Customer

GET GET Person

GET GET Person by QueryString

GET GET Person by Params

Projects

POST localhost:3000/customer

Send

404 Not Found

TIME 21 ms

SIZE 22 B

⊙

Body

Auth

Query

Header


Docs

Preview

Header

Cookie

Timeline



Select a body type from above

We think you are lost!



Insomnia

No Environment Cookies

Filter

REST API Workshop

POST POST Customer

GET GET Person

GET GET Person by QueryString

GET GET Person by Params

Projects

POST localhost:3000/customer

Send

201 Created

TIME 116 ms

SIZE 94 B

⊙

JSON

Auth

Query

Header

Docs

1 - {

2   "\_\_id": "5ab3f442f3d76b4ae4cb433b",

3   "name": "Thomas Anderson",

4   "email": "thomas@gmail.com",

5 }

Preview

Header

Cookie

Timeline

1 - {

2   "\_\_id": "5ab3f442f3d76b4ae4cb433b",

3   "name": "Thomas Anderson",

4   "email": "thomas@gmail.com",

5   "\_\_v": 0

6 }

- mLab (1)
  - rest-api-workshop
    - Collections (2)
      - System
      - customers
    - Functions
    - Users

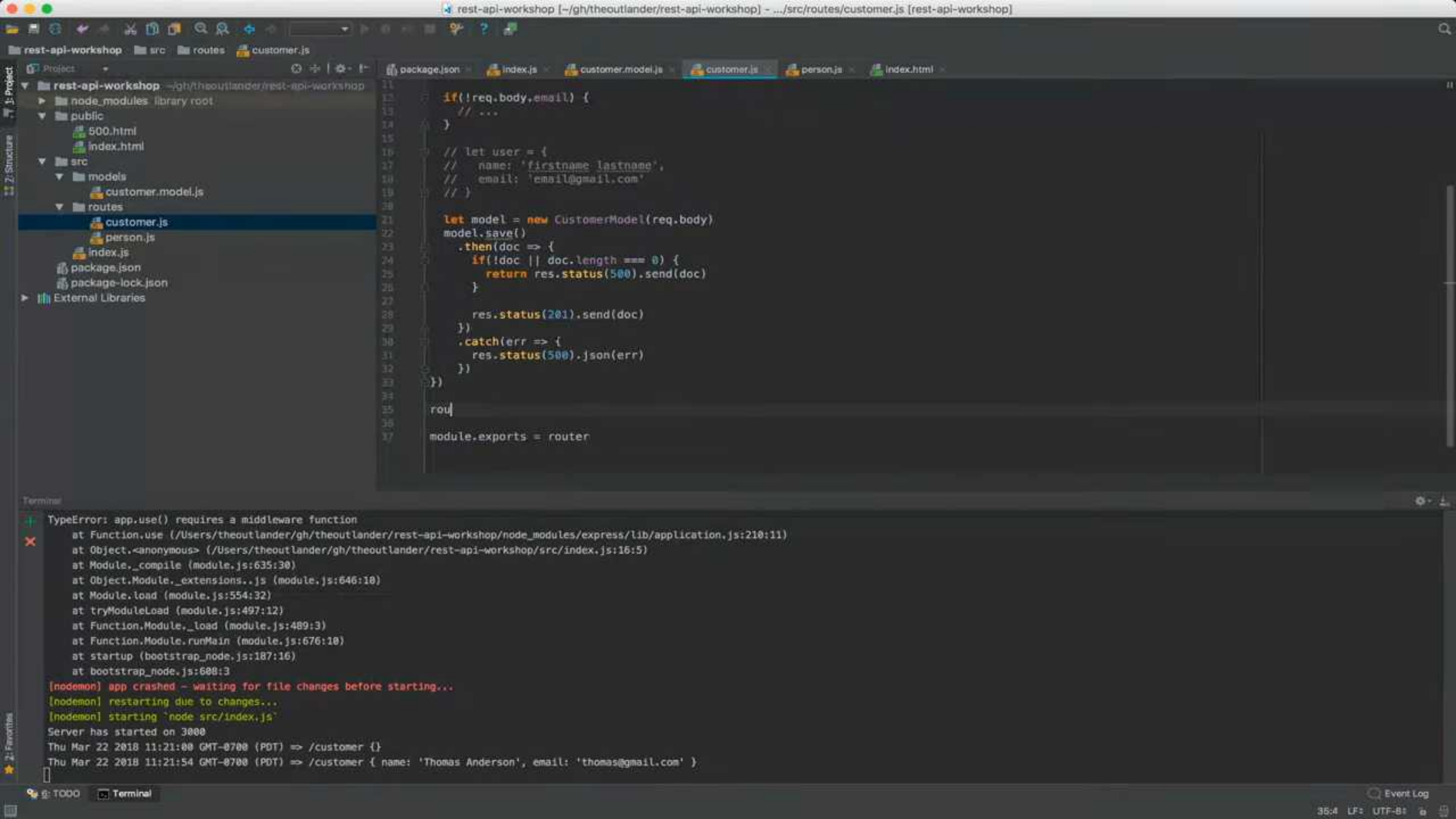
db.getCollection('customers').find({}) db.getCollection('customers').find({})

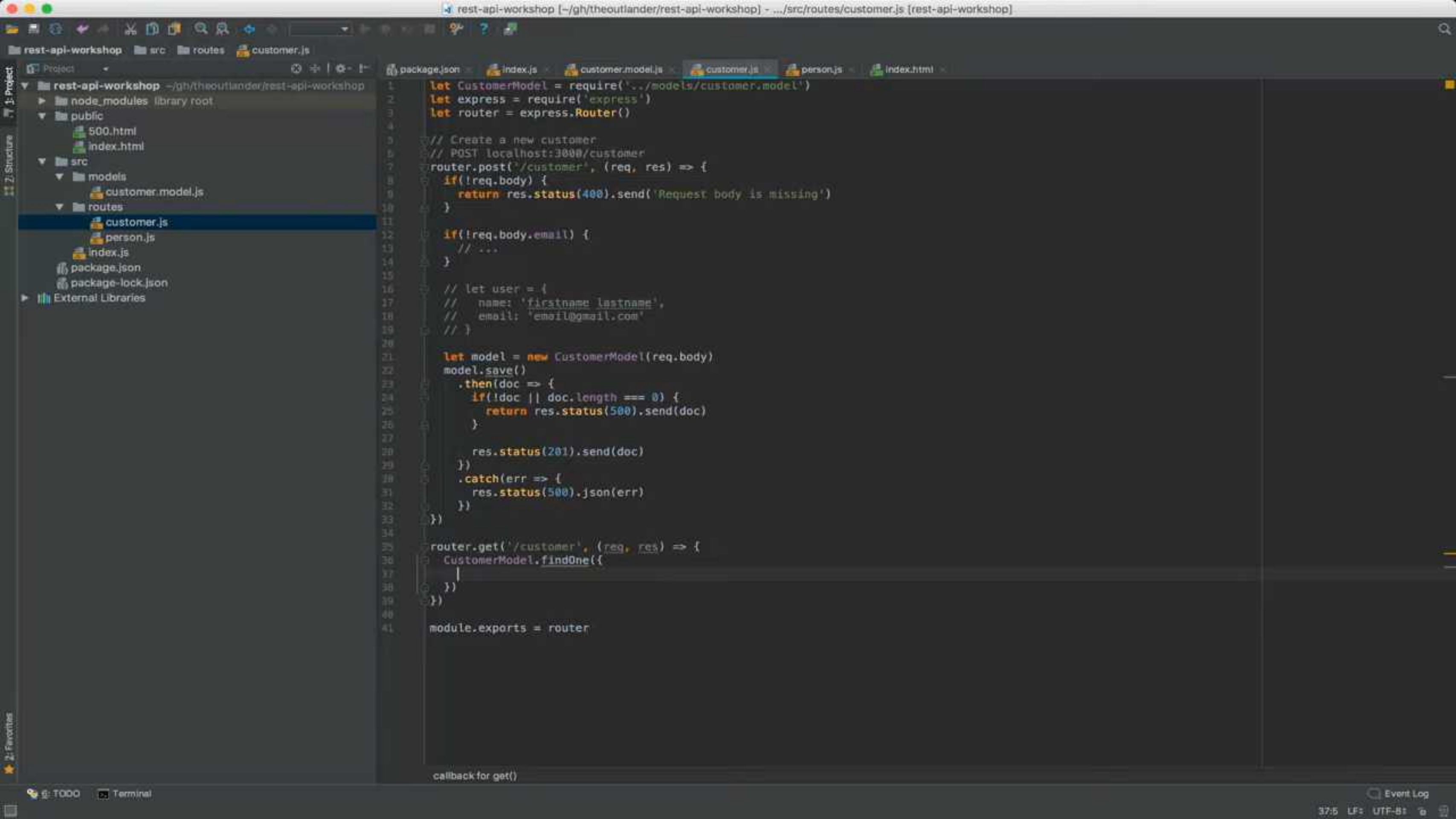
mLab ds221609.mlab.com:21609 rest-api-workshop

```
1 db.getCollection('customers').find({})
```

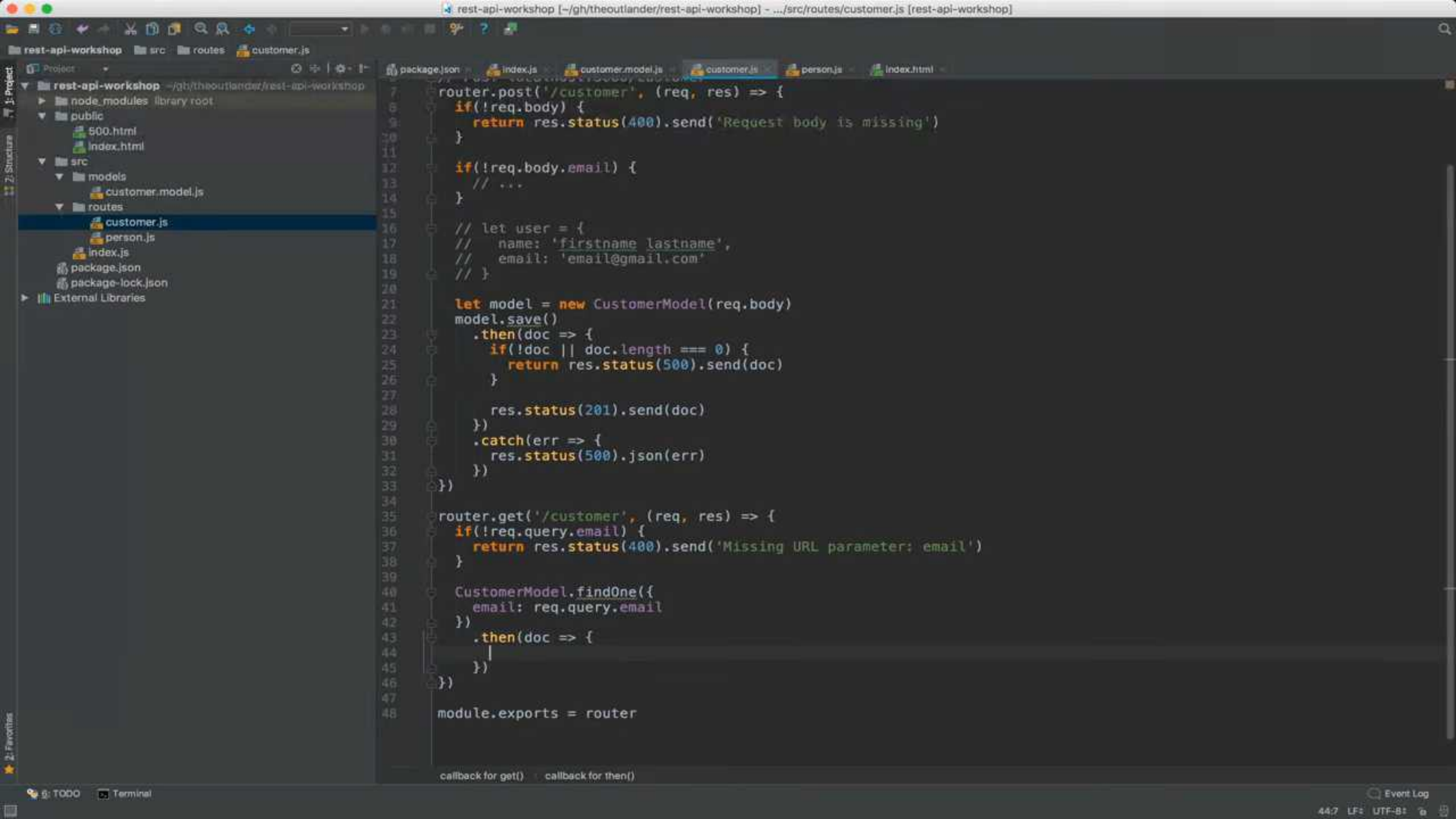
customers 0.089 sec.

Key	Value	Type
(1) ObjectId("5ab3f442f3d76b4ae4cb433b")	{ 4 fields }	Object
_id	ObjectId("5ab3f442f3d76b4ae4cb433b")	ObjectId
name	Thomas Anderson	String
email	thomas@gmail.com	String
_v	0	Int32

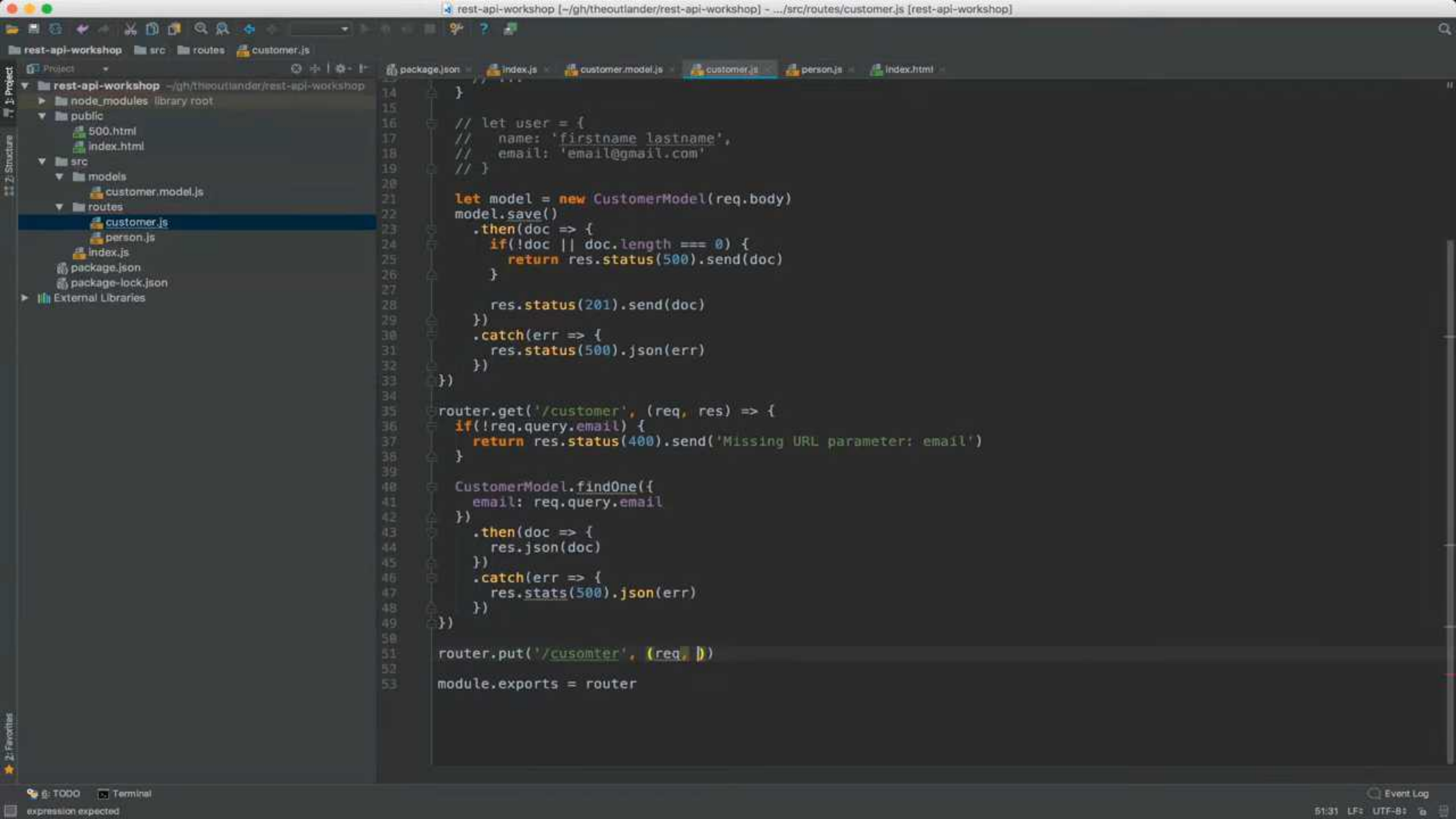


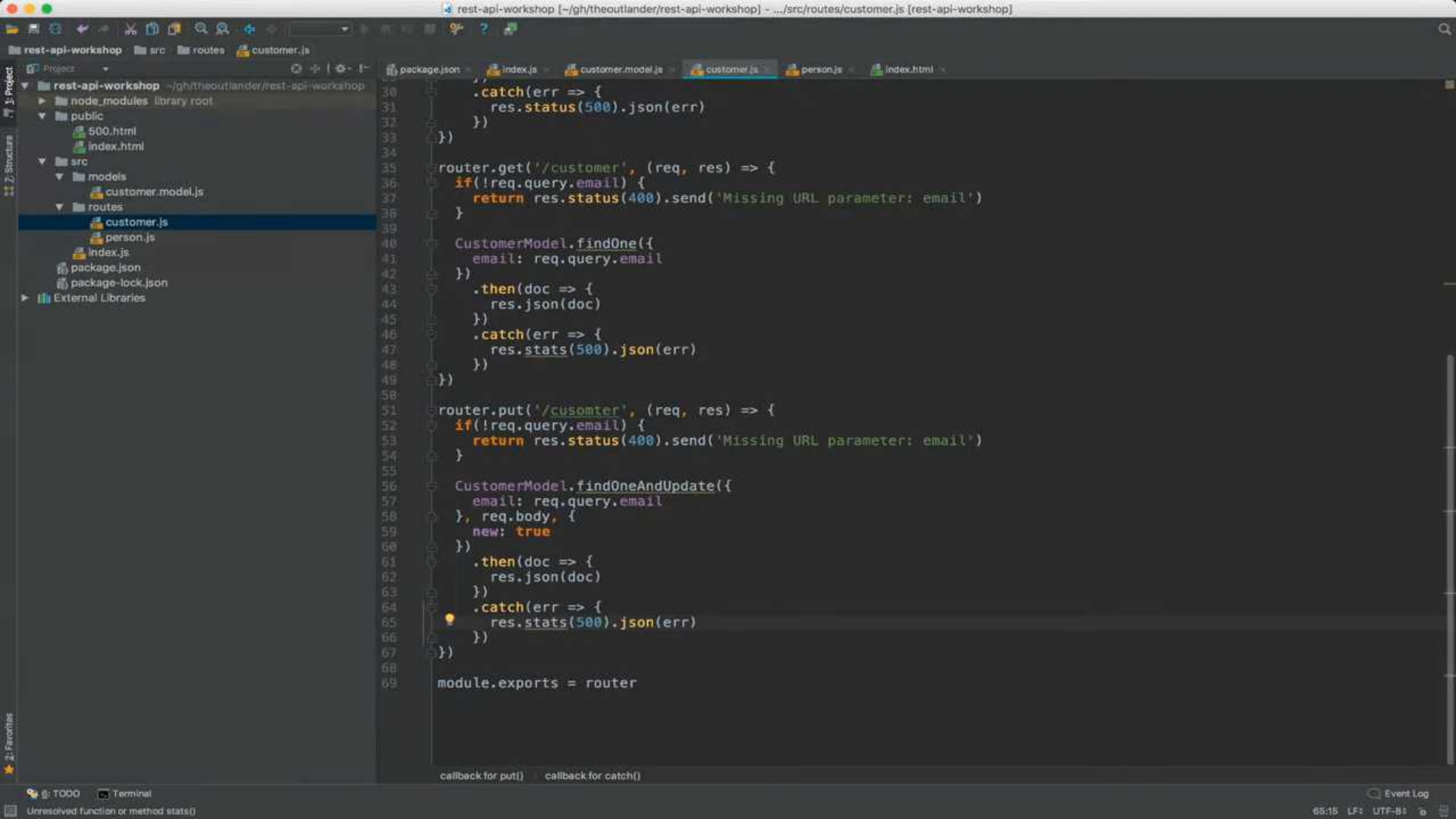


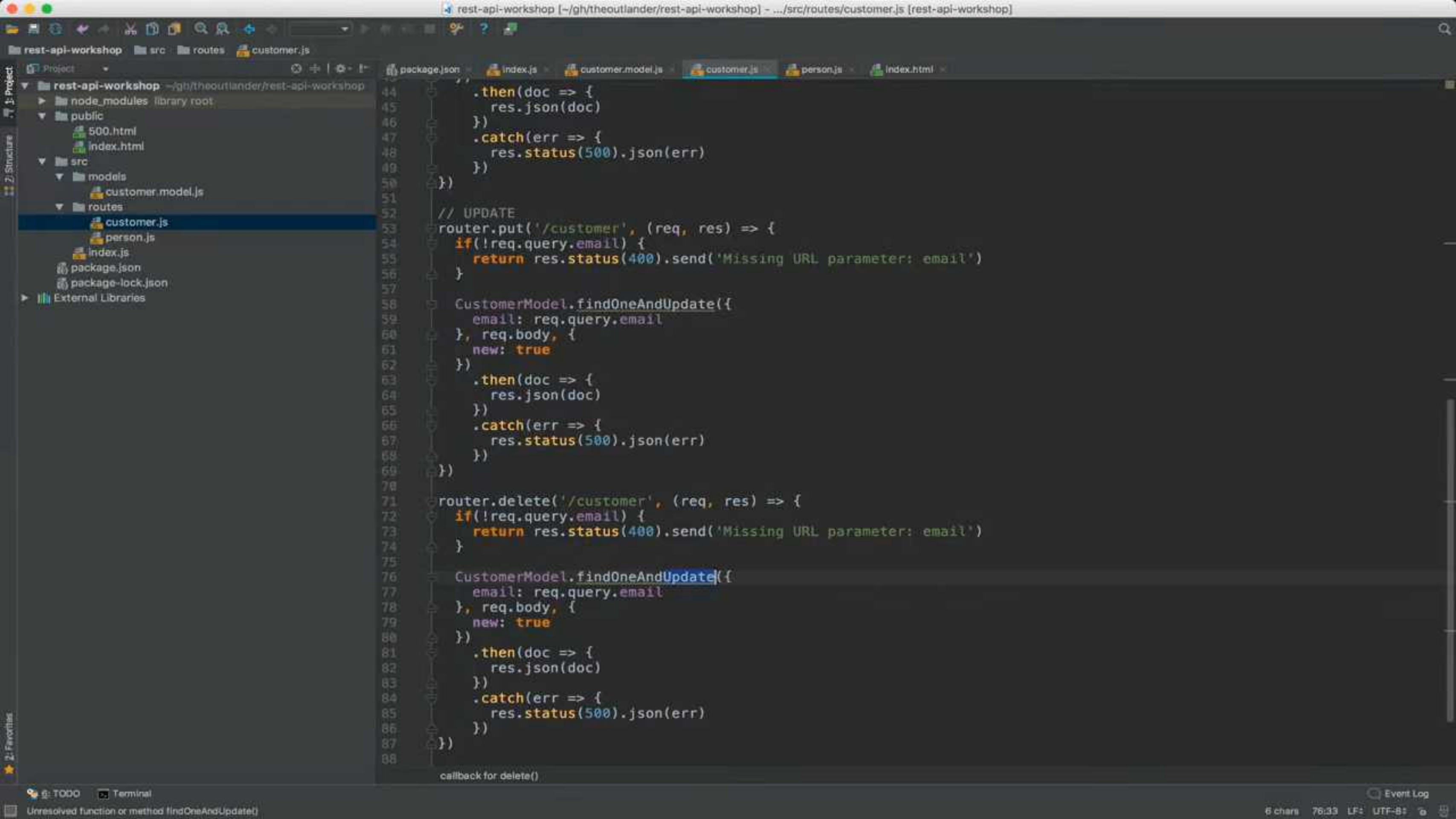
















Insomnia

No Environment Cookies

Filter

REST API Workshop

- PUT POST Customer
- POST PUT Customer
- GET GET Person
- GET GET Person by QueryString
- GET GET Person by Params

Projects

PUT localhost:3000/customer?email=thomas@gmail.comSend

201 CreatedTIME 116 msSIZE 94 B

JSONAuthQueryHeaderDocs

1 - {  
2 "name": "Thomas Updated Anderson",  
3 "email": "thomas@gmail.com"  
4 }

PreviewHeaderCookieTimeline

1 - {  
2 "\_id": "5ab3f442f3d76b4ae4cb433b",  
3 "name": "Thomas Anderson",  
4 "email": "thomas@gmail.com",  
5 "\_v": 4,  
6 }



Insomnia

No Environment Cookies

Filter

REST API Workshop

- PUT POST Customer
- POST PUT Customer
- GET GET Customer
- GET GET Person
- GET GET Person by QueryString
- GET GET Person by Params

Projects

GET localhost:3000/customer?email=thomas@gmail.com

Send

200 OK

TIME 97.3 ms

SIZE 102 B

⊙

JSON

Auth

Query

Header

Docs

1 ...

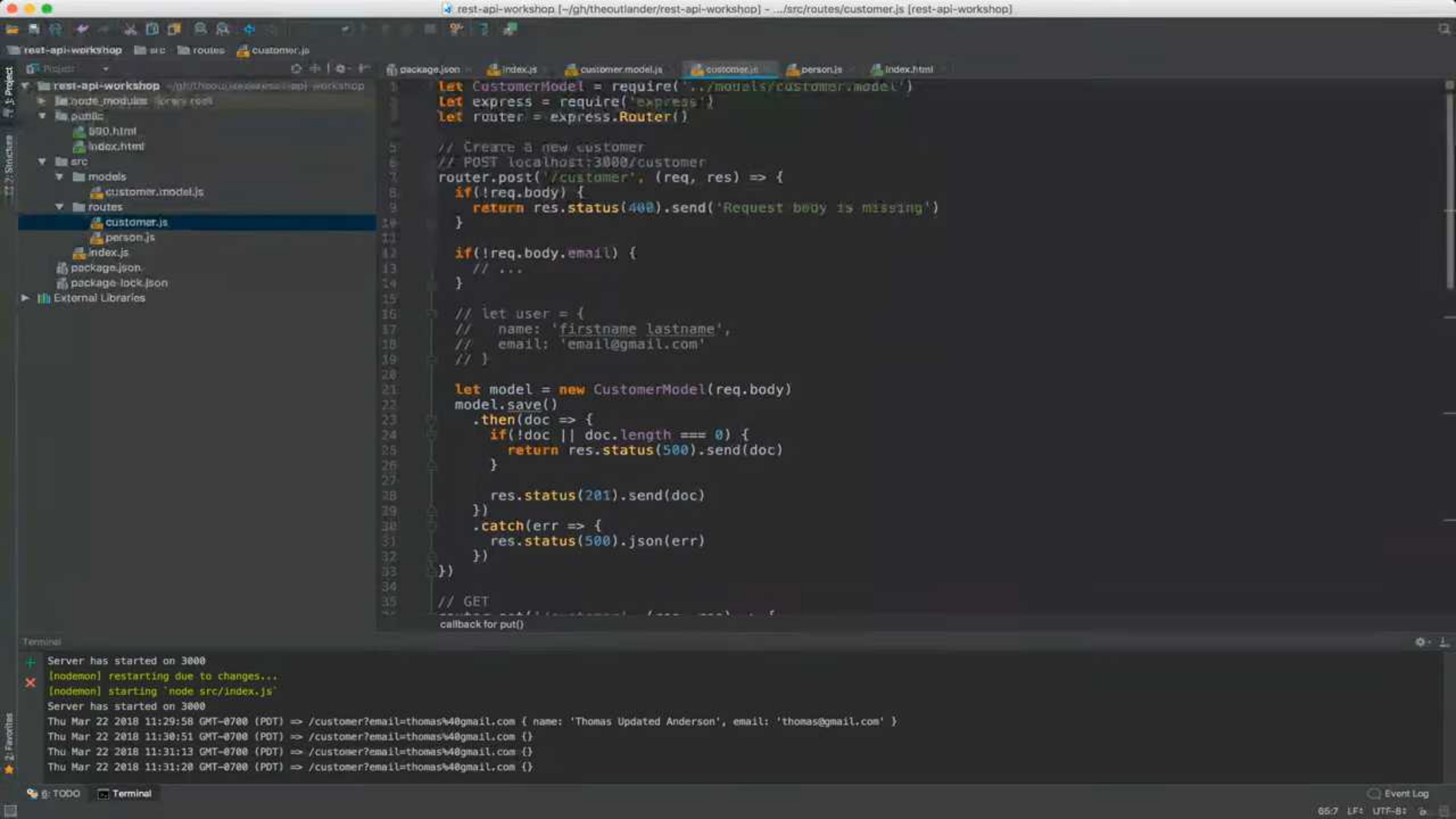
Preview

Header

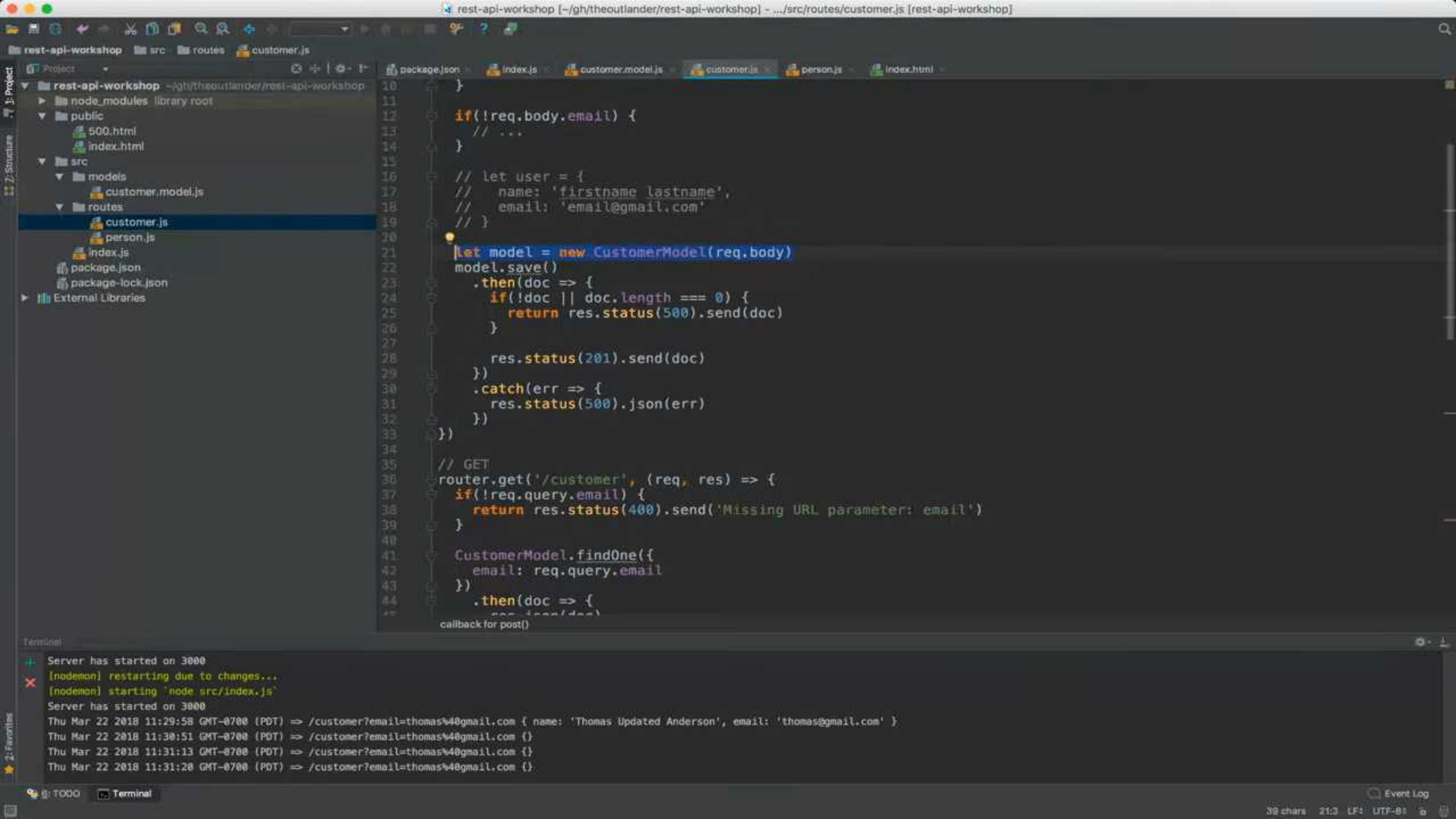
Cookie

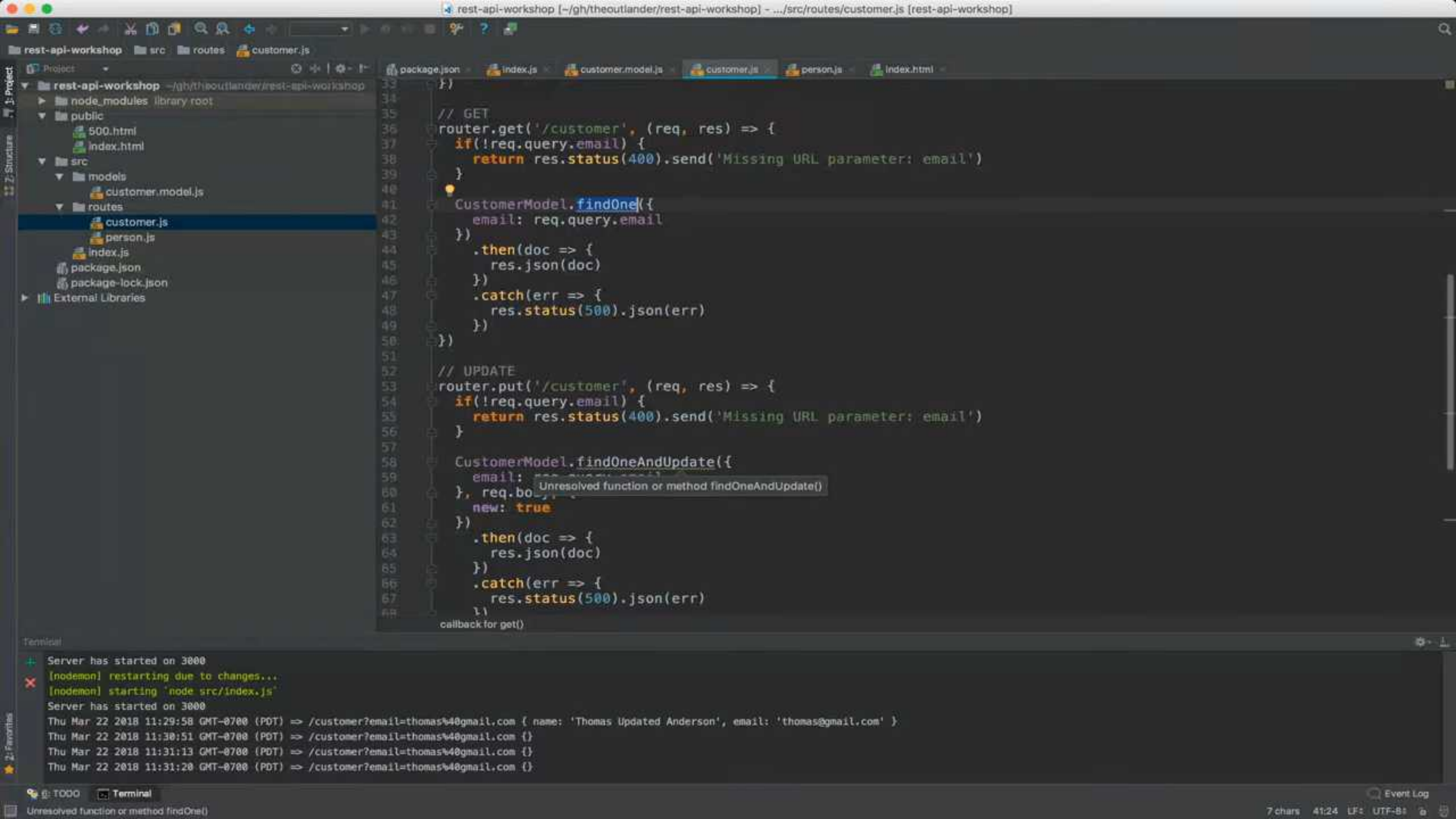
Timeline

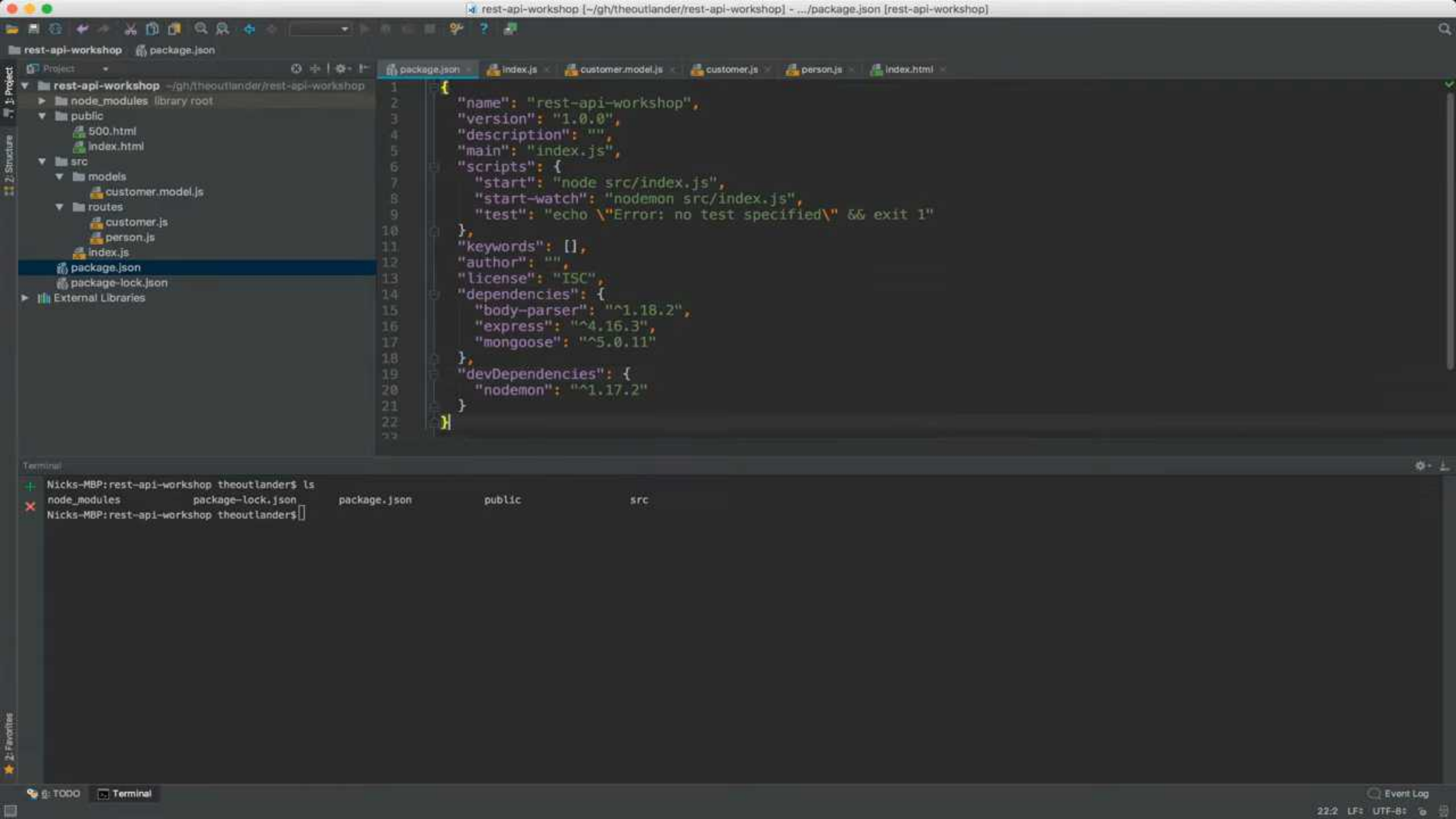
1 - {  
2 "\_id": "5ab3f442f3d76b4ae4cb433b",  
3 "name": "Thomas Updated Anderson",  
4 "email": "thomas@gmail.com",  
5 "\_v": 0  
6 }



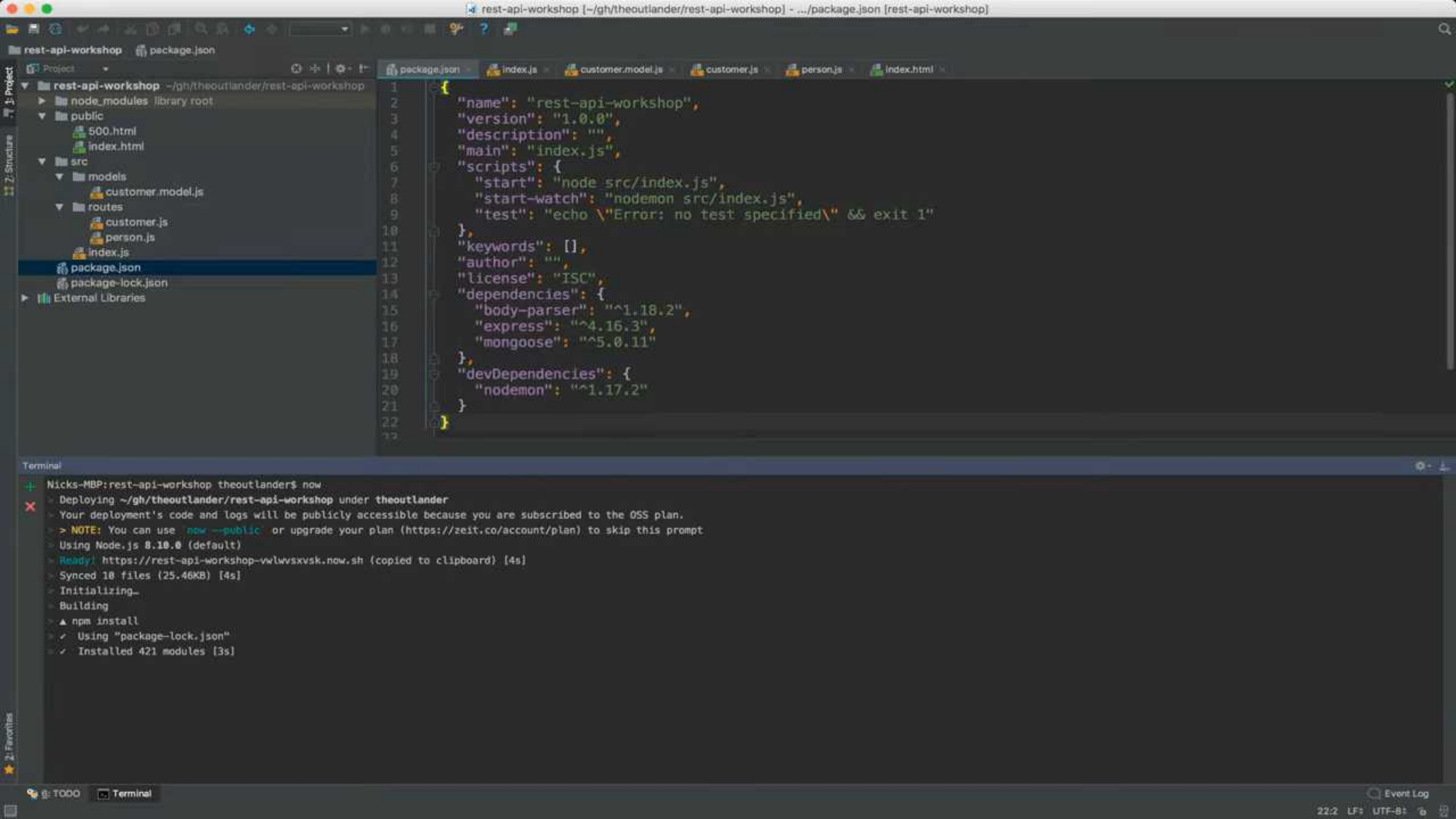


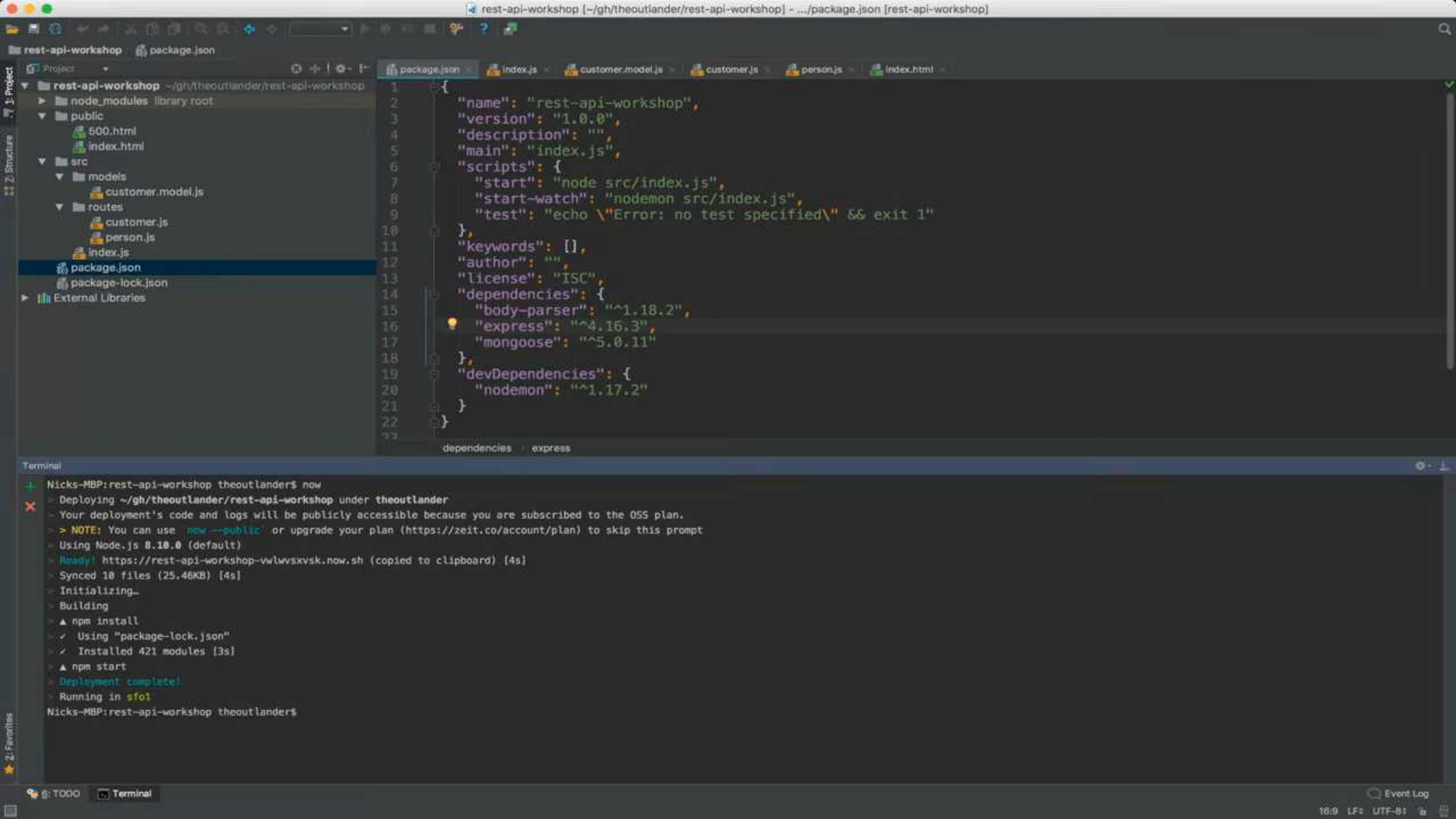
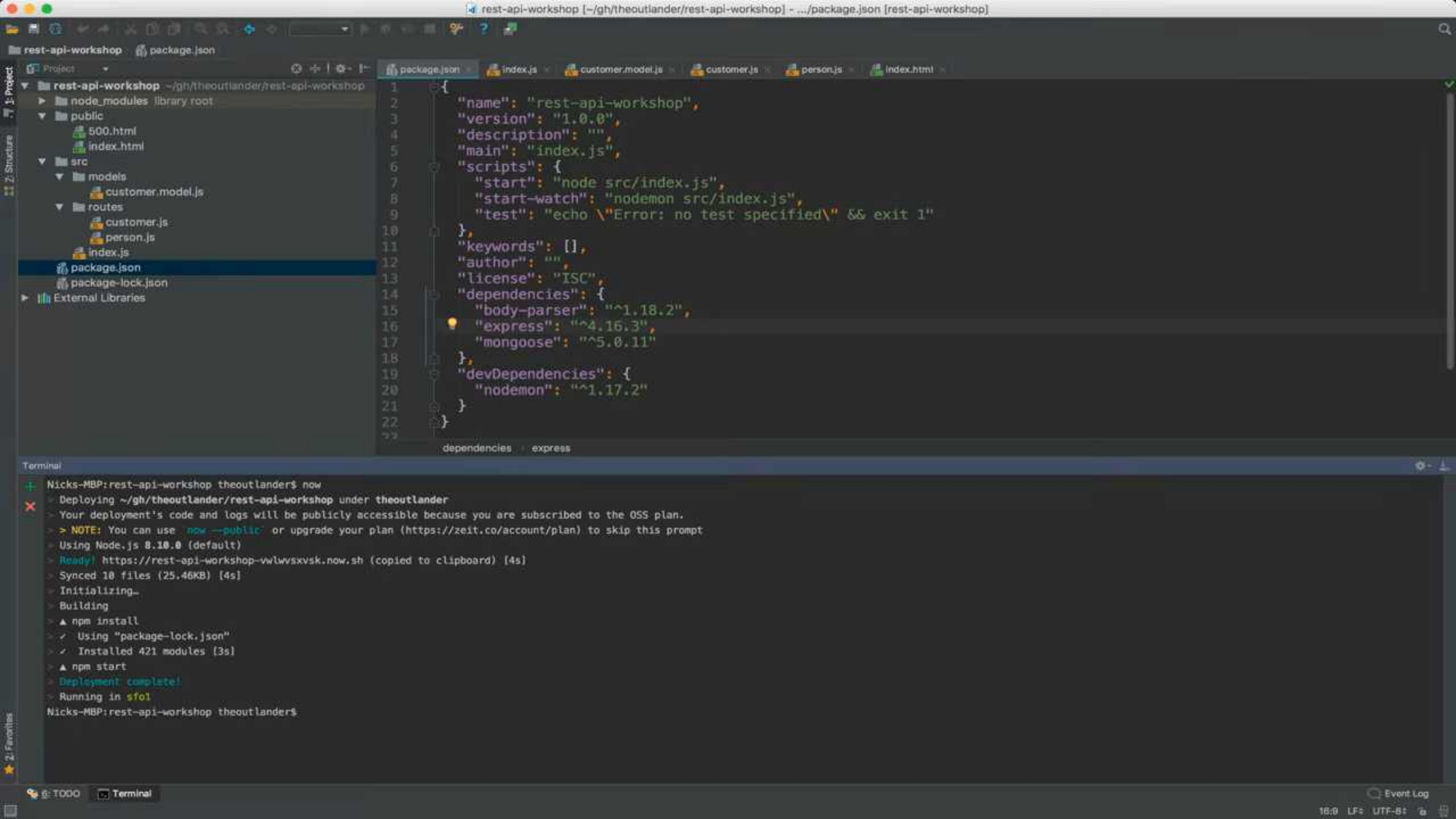


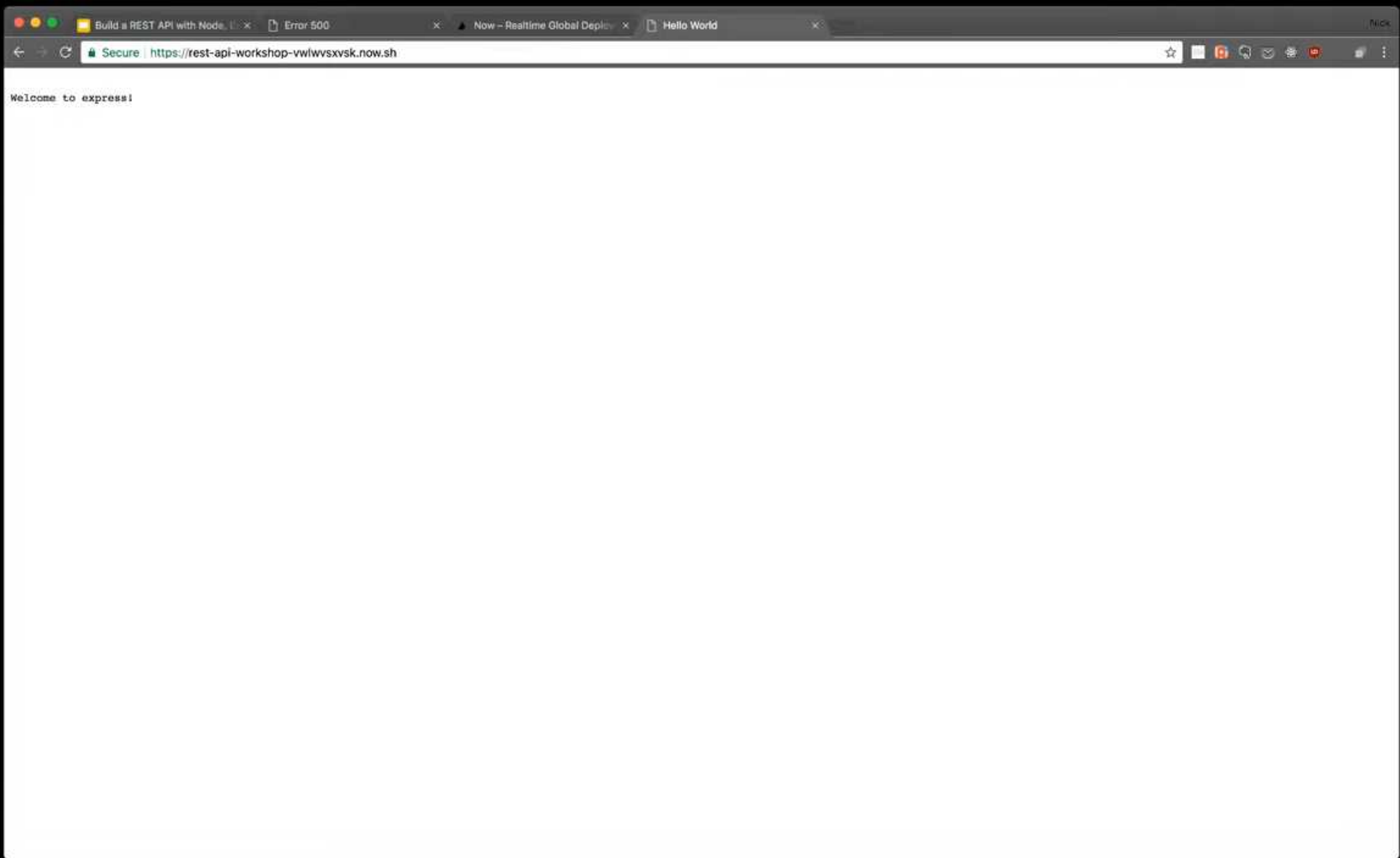




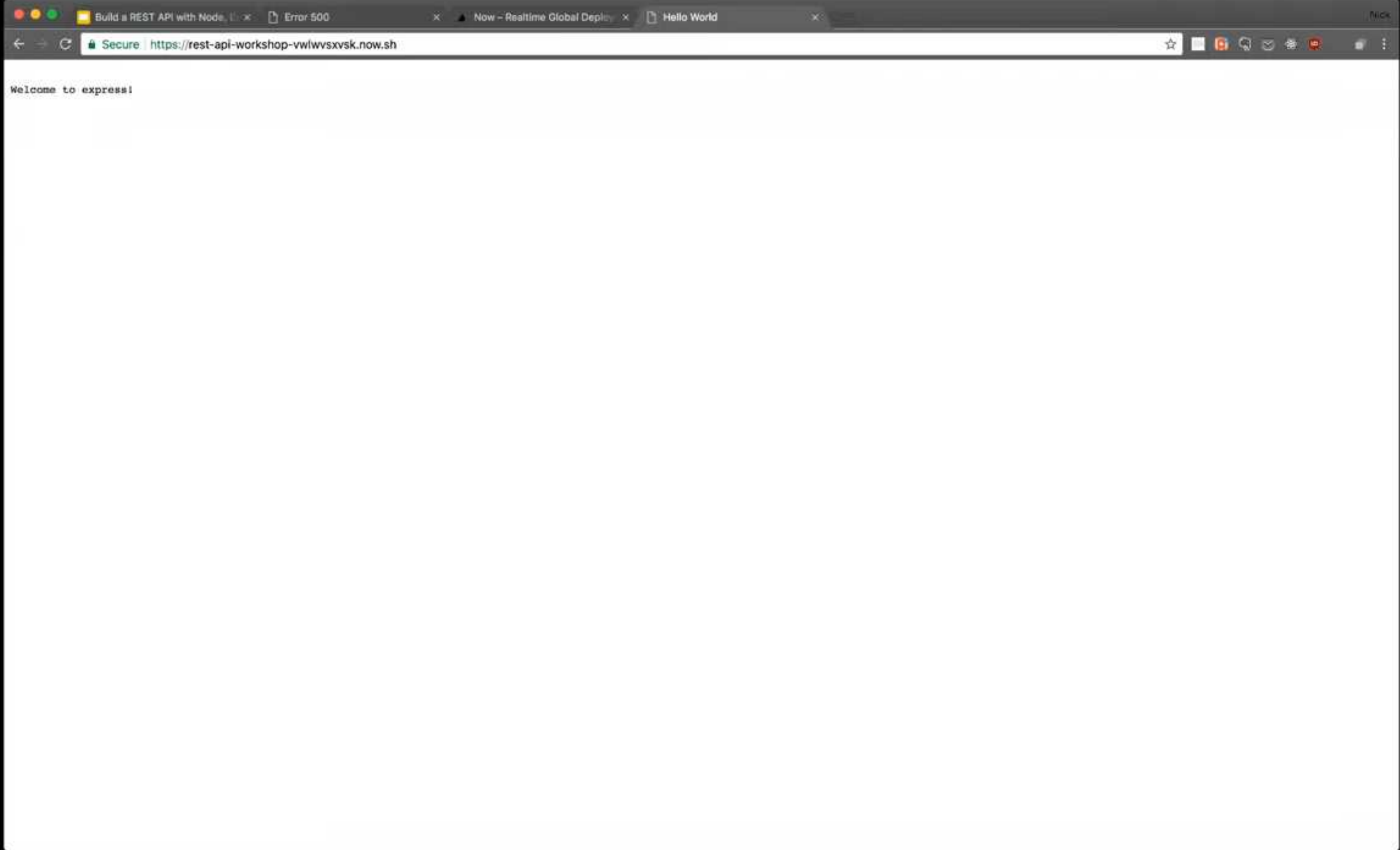














and over again.

2.1K



Let's assume your development stack consists of several libraries, such as React, Babel, Express, Jest, Webpack, etc. When you start a new project, you initialize all these libraries and configure them to work with each other.

With every new project that you start, you will be repeating yourself. You could also introduce inconsistencies in how these libraries are set up in each project. This can cause confusion when you switch between projects.

This is where boilerplates come in. A boilerplate is a template that you can clone and reuse for every project.

The modular Javascript ecosystem simplifies application development through various libraries, frameworks and tools. Boilerplates can be daunting if you don't understand the fundamentals of their underlying components. Let's learn about these basic building blocks while creating our own.

[Click here for source on GitHub](#)

*I am using Webstorm, Git, NodeJS 8.9, NPM 5.6, and React 16. Fire up your favorite IDE, create a blank project, and let's get started!*

## Git Repository: Setup



2.1K



5



Next story  
Insights from Stack Overflow...