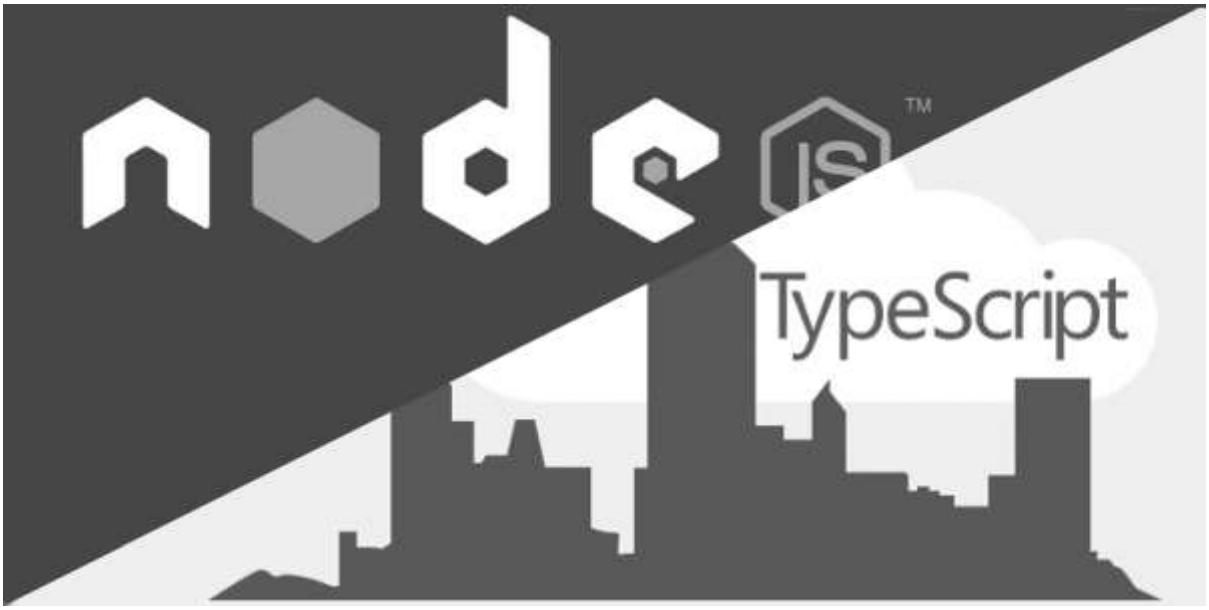
We've made changes to our Terms of Service and Privacy Policy. They take effect on September 1, 2020, and we encourage you to review them. By continuing to use our services, you agree to the new Terms of Service and acknowledge the Privacy Policy applies to you.

Continue

Building RESTful Web APIs with Node.js, Express, MongoDB and TypeScript — Part 4

Dale Nguyen
May 21, 2018 · 4 min read



(Image from OctoPerf)

There is a course about how to build a Web APIs on Lynda, but they didn't use TypeScript. So I decided to make one with TypeScript. There are lots of things that need to improve in this project. If you find one, please leave a comment. I'm appreciated that ;)

Part 1: Setting Up Project

Part 2: Implement routing and CRUD

Part 3: Using Controller and Model for Web APIs

Part 4: Connect Web APIs to MongoDB or others

Part 5: Security for our Web APIs

We've made changes to our Terms of Service and Privacy Policy. They take effect on September 1, 2020, and we encourage you to review them. By continuing to use our services, you agree to the new Terms of Service and acknowledge the Privacy Policy applies to you.

Continue

In this part, we will connect the RESTful API application to local MongoDB, but you can connect to any other database services. Please read Part 1 to install the MongoDB to your machine.

All that you need to do is to import **mongoose package**, and declare **URL** for your MongoDB in the **app.ts** file. After that you will connect your app with your database through **mongoose**.

```
// lib/app.ts

...
import * as mongoose from "mongoose";

class App {

    ...
    public mongoUrl: string = 'mongodb://localhost/CRMdb';

    constructor() {
        ...
        this.mongoSetup();
    }

    private mongoSetup(): void{
        mongoose.Promise = global.Promise;
        mongoose.connect(this.mongoUrl);
    }

}

export default new App().app;
```

After this, your application is ready to launch (*npm run dev*)

```
D:\Dale's Projects\node-express-api\crm [master +0 ~1 -0 !]> npm run dev

> crm@1.0.0 dev D:\Dale's Projects\node-express-api\crm
> ts-node ./lib/server.ts

Express server listening on port 3000
```

You can test your first route (GET /) through web browser (*http://127.0.0.1:3000*)

We've made changes to our Terms of Service and Privacy Policy. They take effect on September 1, 2020, and we encourage you to review them. By continuing to use our services, you agree to the new Terms of Service and acknowledge the Privacy Policy applies to you.

Continue

Remember that all the routes that we set is in **lib/routes/crmRoutes.ts** file.

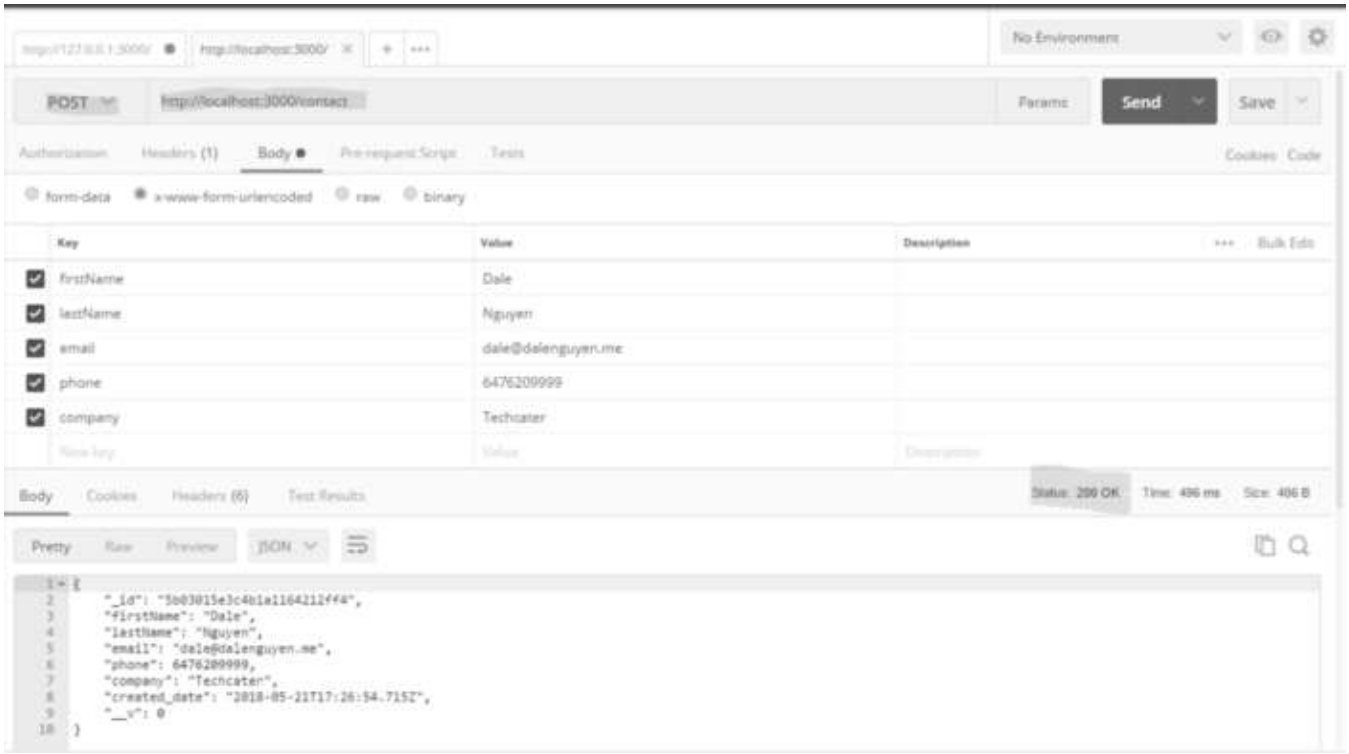
Now, we will test the **C**reate-**R**ead-**U**pdate-**D**elete feature though Postman.

1. **Create your first contact**

I will send a **POST** request to *http://127.0.0.1:3000/contact* with the information of a contact in the body.

Remember to set the content-type in Headers

```
Content-Type: application/x-www-form-urlencoded
```

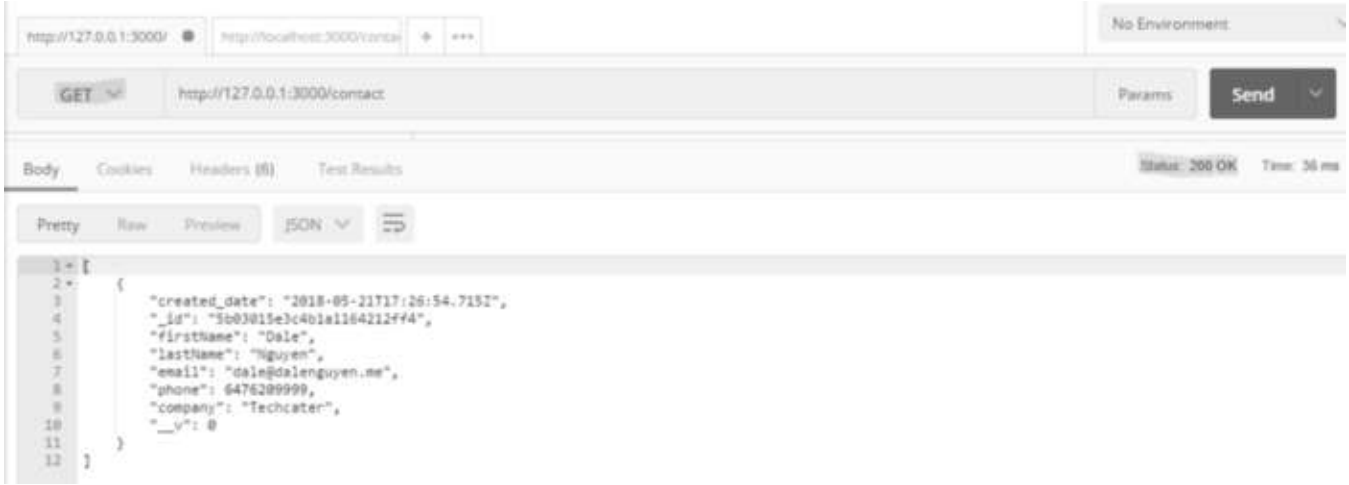


After sending, the server return the status 200 with contact information in the database.

We've made changes to our Terms of Service and Privacy Policy. They take effect on September 1, 2020, and we encourage you to review them. By continuing to use our services, you agree to the new Terms of Service and acknowledge the Privacy Policy applies to you.

Continue

Now there is only one contact that I just created.



3. Get contact by Id

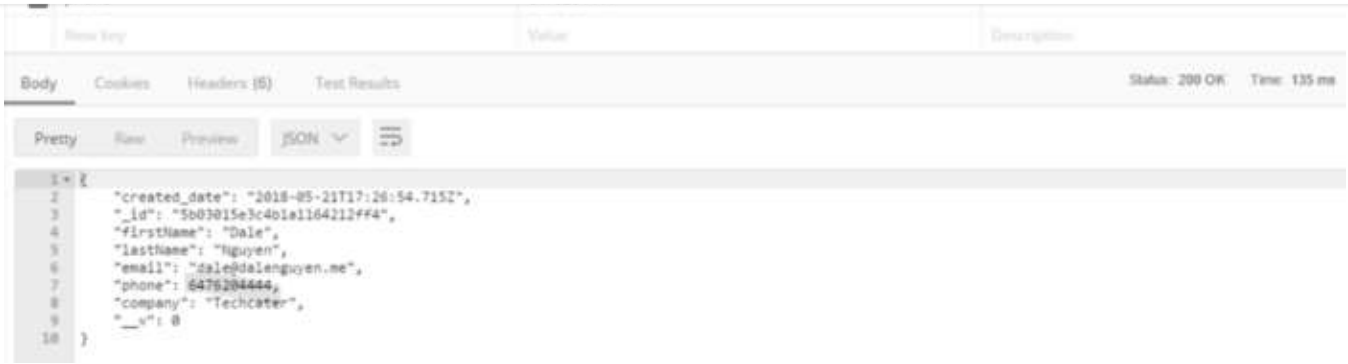
If we want to get a single contact by Id, we will send a **GET** request to *http://127.0.0.1:3000/contact/:contactId*. It will return an Object of your contact. Remember that the ID that we passed to the URL is the **_id** of the contact.

4. Update an existing contact

In case we want to update an existing contact, we will send a **PUT** request to the *http://127.0.0.1:3000/contact/:contactId* together with the detail. For example, I will update the phone number of the contact with **_id: 5b03015e3c4b1a1164212ff4**

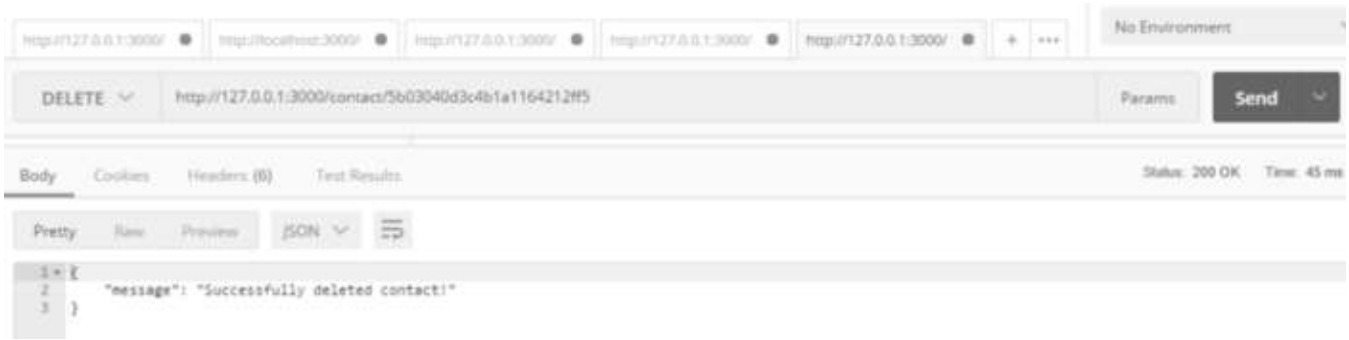
We've made changes to our Terms of Service and Privacy Policy. They take effect on September 1, 2020, and we encourage you to review them. By continuing to use our services, you agree to the new Terms of Service and acknowledge the Privacy Policy applies to you.

Continue



5. Delete a contact

To delete a contact, we will send a **DELETE** request to *http://127.0.0.1:3000/contact/:contactId*. It will return a message saying that "Successfully deleted contact!"



After this, now we have a fully working RESTful Web APIs application with TypeScript and Nodejs. In **part 5**, I will add some extra security for this project. In case you need to jump a head. Please visit my github repository for the full code.

https://github.com/dalenguyen/rest-api-node-typescript

Nodejs API Expressjs Typescript Mongodb

About Help Legal

Get the Medium app

We've made changes to our Terms of Service and Privacy Policy. They take effect on September 1, 2020, and we encourage you to review them. By continuing to use our services, you agree to the new Terms of Service and acknowledge the Privacy Policy applies to you.

Continue