**Making Your Backend World with Node.js -** [Karthika Velmurugan](https://medium.com/@karthikavel2000?source=post_page-----8e93babadb9e--------------------------------) May 16, 2022

So glad to come back again with wonderful article , This article really helpful for the beginners who are all very passionate about to create various CRUD API’s using Node.js.

Before diving in to the article, let me tell, what is the main responsibilities of a Back-End Developer? — *They need to design the DB properly and to deliver the necessary CRUD APIs to the Front-End Developer.*

What is CRUD ?  CRUD is nothing but, its short acronym for “CREATE (POST)”,”READ ( GET )”,”UPDATE ( PUT )” and “DELETE ( DELETE )” backend operations.

From every complex projects to small projects, everything should having this four kind of backend operations only.

**Why we need Node.js ?**

Normally, Millions of the frontend developers, are preferred to write JavaScript for client-side developing. Now able to write the server-side code in the same ECMA JS standards without the need of learning different language for backend.

*For ex* : *Express.js, Nuxt.js, Nest.js, etc.*— So, node.js creates the virtual runtime environment outside the browser to get execute these one of the Server-Side Scripting language.

Important Note :

1. Node.js is an Open Source and *JavaScript Run Time Environment* (It’s not a language)
2. Node.js runs the *V8 JavaScript* engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant .
3. Its mainly creating the Runtime Environment for executing the “ServerSide JavaScript” Code.

For more deep learning.

**[Introduction to Node.js](https://nodejs.dev/learn/introduction-to-nodejs" \t "_blank)**

[Getting started guide to Node.js, the server-side JavaScript runtime environment. Node.js is built on top of the Google…](https://nodejs.dev/learn/introduction-to-nodejs" \t "_blank)

[nodejs.dev](https://nodejs.dev/learn/introduction-to-nodejs" \t "_blank)

It’s a time for becoming Back End Developer. Let’s go and dive into the code.

Let’s take the simple project “ STUDENT MANAGEMENT SYSTEM API” as a example, now we are going to create CRUD API’s with proper error validations.

Its Interesting !!

Before that, Execute the following prerequisite node installation commands, for importing the necessary packages:

* First, Create the one project folder named as, Student\_Management\_System.
* Execute the, npm init : To initilizing the node project, make sure the npm creates “package.json” configuration file after executing this command.

Note :*what is package.json*?

This a heart of any node project. Which holds the information about, *npm packages dependencies*, *entry point of the project*(Either index.js / server.js ), *run scripts* and*metadata about the projects*.

* Execute the npm install express : To install the server side scripting framework. (Express.js)
* Execute the npm install body-parser : It is an npm library used to process data sent through an HTTP request body.
* Execute the npm install MySQL-promise : It is an npm library used to access and process the MySQL DB layer requests and responses asynchronously.

Apart from output of any project, Structuring the code is very important. So, we should maintain the layers properly in all our projects. I will explain the layer architecture for Node.js

Diagram

Description automatically generated

Let’s start to build the index.js🥸 (Entry point of the file), 🤩 🥳

Text

Description automatically generated

Here, we imported the express and body-parser packages, by the way we can start the express server and handling the HTTP request object.

*In the above code, app.listen(3005,()=>{})*→ Use this function to start the nodejs server in configured port “3005 — Its optional”

app.get(“/test”,(req,res)=>{}) → Most of the developers, are prefer to use “ test” API, for testing purpose.

How we know, whether the nodejs server is up or not after deploying the project in the server node?

*console.log()*→ Its print the output in the console. Not in the client side UI.

So, we created one “http://localhost:3001/test” API to get confirm the server working state.

The primary role of every backend developer is, to define the Database architecture perfectly.

Let’s create the DB for “*Student Management System*”,

Diagram

Description automatically generated

Once it is created, will go and jump into the code:

*(1) Database Layer :*

* Create one database/config.js file.This file should have the database connectivity credentials.

Text

Description automatically generated

* Create database/db.js file, for preparing the query string of the DB queries. This function will return the MySQL results as a outcome.

*Note : [mysql-promise](https://www.npmjs.com/package/promise-mysql" \t "_blank) , We use this package for accessing and querying the DB.*

Text

Description automatically generated

Now, we will write the CRUD operation API one by one. First, we see how to fetch the student records from the DB with the help of those layers.

* 1. *GET : Fetching All Student Records from DB*

(2) Controller Layer :

* Create *controllers / getAllStudents.controller.js* file.

Text

Description automatically generated

(3) Service Layer :

* Create *services/getAllStudents.service.js* file.

Text

Description automatically generated

*(B) POST : Register the new student into the DB*

(2) Controller Layer :

* Create *controllers / createStudent.controller.js* file.

Text

Description automatically generated

(3) Service Layer :

* Create *services/createStudents.service.js* file.

*Error handling :*

* MobileNumber should be given by the student in 10 digit only.

Text

Description automatically generated

*(c) PUT: Update the student record in the DB*

(2) Controller Layer :

* Create *controllers / updateStudent.controller.js* file.

Text

Description automatically generated

(3) Service Layer :

* Create *services/updateStudent.service.js* file

Text

Description automatically generated

*(D) DELETE : Delete the student record from DB*

(2) Controller Layer :

* Create *controllers / deleteStudent.controller.js* file.

Text

Description automatically generated

(3) Service Layer :

* Create *services/deleteStudent.service.js* file

Text

Description automatically generated

(4) Routing Layer :

* Create *routes/route.js* file for routing the incoming request to the respective services.

Text

Description automatically generated

(5) Index.js

* Import the router function inside the index.js main entry file.

Text

Description automatically generated

Using app.use(router) function ,we imported the router layer functionalities here.

To access this API’s,

GET : <http://localhost:3001/>

POST : <http://localhost:3001/>

Request Body :

Text

Description automatically generated

PUT : http://localhost:3001/{student\_id}

DELETE : [http://localhost:3001/{student\_id}](http://localhost:3001/%7Bstudent_id%7D)

Now you did outstanding job, you developed CRUD API.

For your more reference, the source code is available in [Github](https://github.com/KarthikaVelmurugan/Student_Management_System_NodeJS" \t "_blank). I hope, this article really helpful to the beginners who are all have a passion about to create and learn about the backend APIs. Feel free to star the repository