Get data from MongoDB through NodeJS in ReactJS using Axios

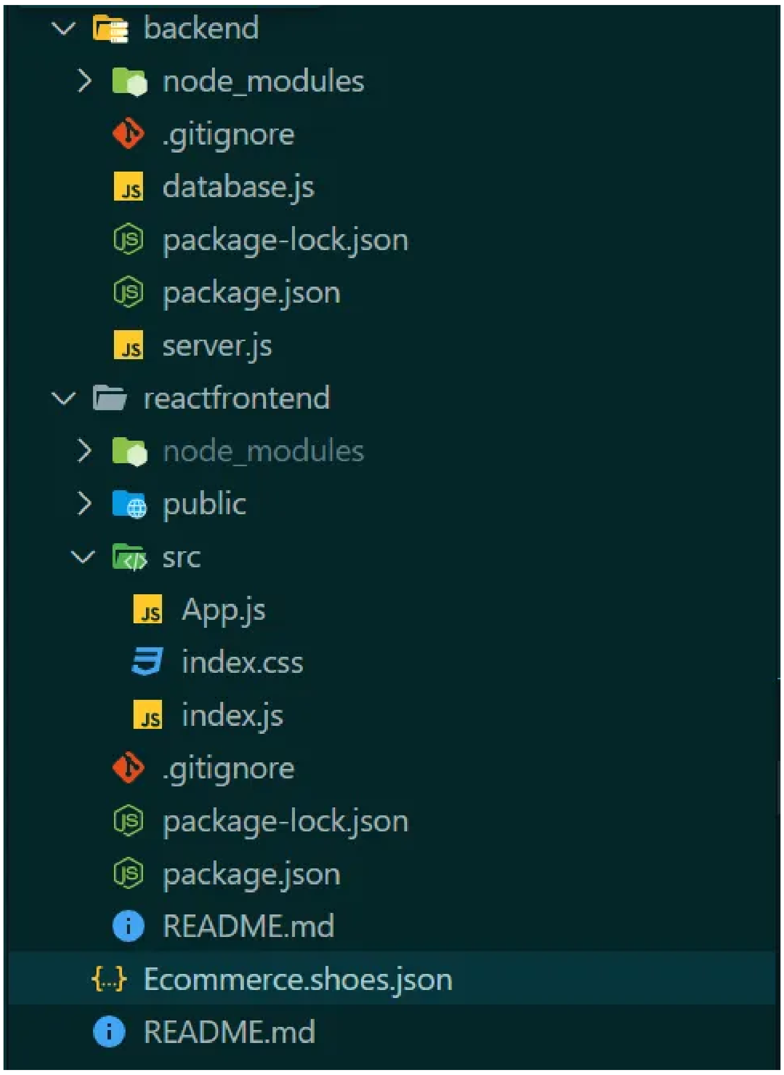
Maisha Maliha · Follow
4 min read · Jul 21, 2023

Listen Share

In this post, the goal is to get data or an array of objects from MongoDB (Not Mongoose) through NodeJS (not Express) and then let ReactJS get those data from NodeJS using Axios.

ReactJS gets data from the server through an API call. In our NodeJS server, we will send data or JSON data whenever we get a request from our front-end, that is, ReactJS.

Here are our files and folder structure:



```
backend
  node_modules
  .gitignore
  database.js
  package-lock.json
  package.json
  server.js
reactfrontend
  node_modules
  public
  src
    App.js
    index.css
    index.js
  .gitignore
  package-lock.json
  package.json
  README.md
Ecommerce.shoes.json
README.md
```

I have separate folders for all the backend and the frontend files. Our root folder will have two folders named “backend” and “reactfrontend”. Name them whatever you want.

Database setup

First, we will work in our backend. Let's set up our MongoDB database. In my MongoDB database, I created a *database* named “**Ecommerce**”; inside that database, I made a *Collection* named “**shoes**”. And in that collection, I have the following data:

```
[{
  "_id": {
    "$oid": "64b97610a5b563aadd83b266"
  },
```

```
    "black",
    "white"
  ],
  "size": [
    32,
    34,
    36,
    40
  ],
  "shop_id": 2
},
{
  "_id": {
    "$oid": "64b9773da5b563aadd83b267"
  },
  "id": 2,
  "shop_id": 1,
  "name": "maily 201",
  "colors": [
    "black",
    "white"
  ],
  "size": [
    32,
    36,
    40
  ]
},
{
  "_id": {
    "$oid": "64b97764a5b563aadd83b268"
  },
  "id": 3,
  "shop_id": 2,
  "name": "k201",
  "colors": [
    "black",
    "white"
  ],
  "size": [
    32,
    36,
    40
  ]
},
{
  "_id": {
    "$oid": "64b97790a5b563aadd83b269"
  },
  "id": 4,
  "shop_id": 1,
  "name": "M452 shulai",
  "colors": [
    "black",
    "white"
  ],
  "size": [
    32,
    36,
    40
  ]
},
{
  "_id": {
    "$oid": "64b977a0a5b563aadd83b26a"
  },
  "id": 5,
  "shop_id": 1,
  "name": "Ma52 shulai",
  "colors": [
    "black",
    "white",
    "green"
  ],
  "size": [
    36,
    40
  ]
},
{
  "_id": {
    "$oid": "64b97849a5b563aadd83b26b"
  },
  "id": 6,
  "shop_id": 2,
  "name": "Pikachula",
  "colors": [
    "black",
    "pink",
    "green"
  ],
  "size": [
    32,
    34,
    36
  ]
},
{
  "_id": {
    "$oid": "64ba1b534660ca90972ad6a6"
  },
  "id": 7,
  "shop_id": 1,
  "name": "kaluka",
  "colors": [
    "white",
    "black"
  ],
  "size": [
    40,
    44
    ]
}]
```

So your database should be ready.

Backend setup

Let's start our work in the "backend" folder. We are to get data from the database through NodeJS. So, we create a file named **"database.js"**. Here is the code inside that file.

```
// ******* backend/database.js *******

const { MongoClient } = require("mongodb");
const uri = "mongodb://0.0.0.0:27017";

const client = new MongoClient(uri);
client.connect();

async function shoes() {
    try {
        const dataset = await client.db('Ecommerce').collection('shoes').find().toArray();
        return JSON.stringify(dataset);
    }
    catch {
        console.log("db closed");
        await client.close();
    }
}
module.exports = {shoes};
```

We got all the data from the **"shoes"** collection and converted it into an array. And then we

Now I will show you why we used *JSON.stringify(dataset)* and not *dataset.toString()*

```
// you will get an array of objetcs
console.log(dataset);
// you will get a string version of that array of objs
console.log(JSON.strigify(dataset));
// but if you did this
console.log(dataset.toString();
// this will give this output:
// [object Object],[object Object],[object Object],[object Object],[object Object],[object Object],[o
```

Let's get this data in the NodeJS server. We create a file named **"server.js"**.

```
// ******* backend/server.js *******

const http = require('http');
const database = require('./database');

http.createServer(async (req,res)=> {
    // THIS IS FOR CORS ERRORS
    res.setHeader('Access-Control-Allow-Origin', '*'); /* @dev First, read about security */
    res.setHeader('Access-Control-Allow-Methods', 'OPTIONS, GET');
    res.setHeader('Access-Control-Max-Age', 2592000); // 30 days
    if(req.url == '/api/shoes'){
        try {
            res.writeHead(200, {'Content-Type':'application/json'});
            const dataset = await database.shoes(); // here we get the string json
            res.write(dataset); // whoever requests will get the string json as response
        }
        finally {
            res.end(); // must end response. DONT FOGET
        }
    }
}).listen(4000);
// the server will listen to all request in port 400 because
// reactjs is run on port 3000. react will request for data from port
// 3000 to our nodejs in port 4000
```

Our code for the backend is done.

Frontend Setup

As you saw in the folder structure, our react project is another folder. So simply go to your root folder and then install ReactJS using

```
npx create-react-app reactfrontend
```

Install **'Axios'** too.

```
npm install axios
```

now delete all the unnecessary files according to the picture of the files and folder structure. Here is the App component in **'App.js'** file:

```
// ******* reactfrontend/src/App.js *****

import React, { useEffect } from "react";
import Axios from 'axios';

export default function App() {
    // will update list as database updates on refreshing the site
    const [list, setList] = React.useState([]);

    // will be run once
    useEffect(()=> {
        // here we get the data by requesting data from this link
        // to our nodejs server
        Axios.get('http://localhost:4000/api/shoes')
        .then((res)=> setList(res.data));
    }, []);

    // creating list of shoes
    let val = list.map((item)=>{
        return <li key={item.id}>{item.name}</li>
    });

    return (
        <div>
            <h1>hello world</h1>
            <p>i live in this world</p>
            <ol>
                {val}
            </ol>
        </div>
    )
}
```

Here is the **'index.js'** file:

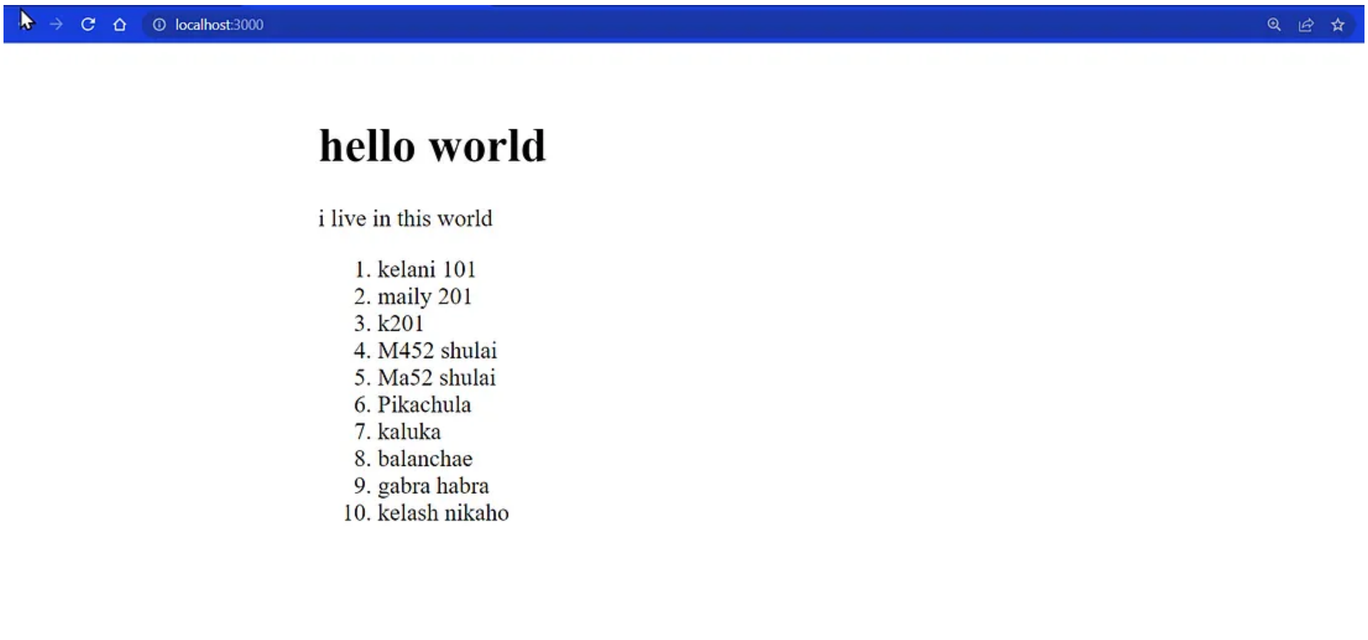
```
// ******* reactfrontend/src/index.js *******

import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Now to run this whole application, go to your 'backend' directory in the terminal and run: ***node server.js***

After running the server go to your 'reactfrontend' folder directory in the terminal type: ***npx start***

Now on your browser type: localhost:3000

You should be able to see your database data in a list like mine.



hello world

i live in this world

```
 1. kelani 101
 2. maily 201
 3. k201
 4. M452 shulai
 5. Ma52 shulai
 6. Pikachula
 7. kaluka
 8. balanchae
 9. gabra habra
10. kelash nikaho
```

Hope this helped get the idea. Do follow me on medium to get more of react and Nodejs.

Reactjs Nodejs Axios

Follow

Written by Maisha Maliha

11 Followers · 4 Following

Hello, I am Maisha. I am from Bangladesh. I am studying CSE. I like learning new spoken languages and coding.

More from Maisha Maliha

In Dev Genius by Maisha Maliha

Quick OpenGL setup on CodeBlocks with freeglut

A quick start to setting up OpenGL for CodeBlocks. First, we will download everything and later we will setup our first OpenGL project on...

Feb 5, 2023 60

Maisha Maliha

My struggles in setting up OpenGL for the first time

I sat down on 2nd February 2023 to setup OpenGL on my Laptop. I couldn’t. I tried again on Friday(the next day)but I still couldn’t...

Feb 4, 2023 1

In Towards Dev by Maisha Maliha

Download latest version of GCC/G++(13.0.1) for C++23

Recently, I have been concerned about being up to date with C++. When I was learning Java, it was pretty much easy to determine and…

Feb 6, 2023 3

Maisha Maliha

C++ Syllabus/Topics/Roadmap for Beginners

C++ is a compiled language so you must learn about the compiler you are using because not all compilers will give the same result. C++…

Mar 25, 2023 6

See all from Maisha Maliha

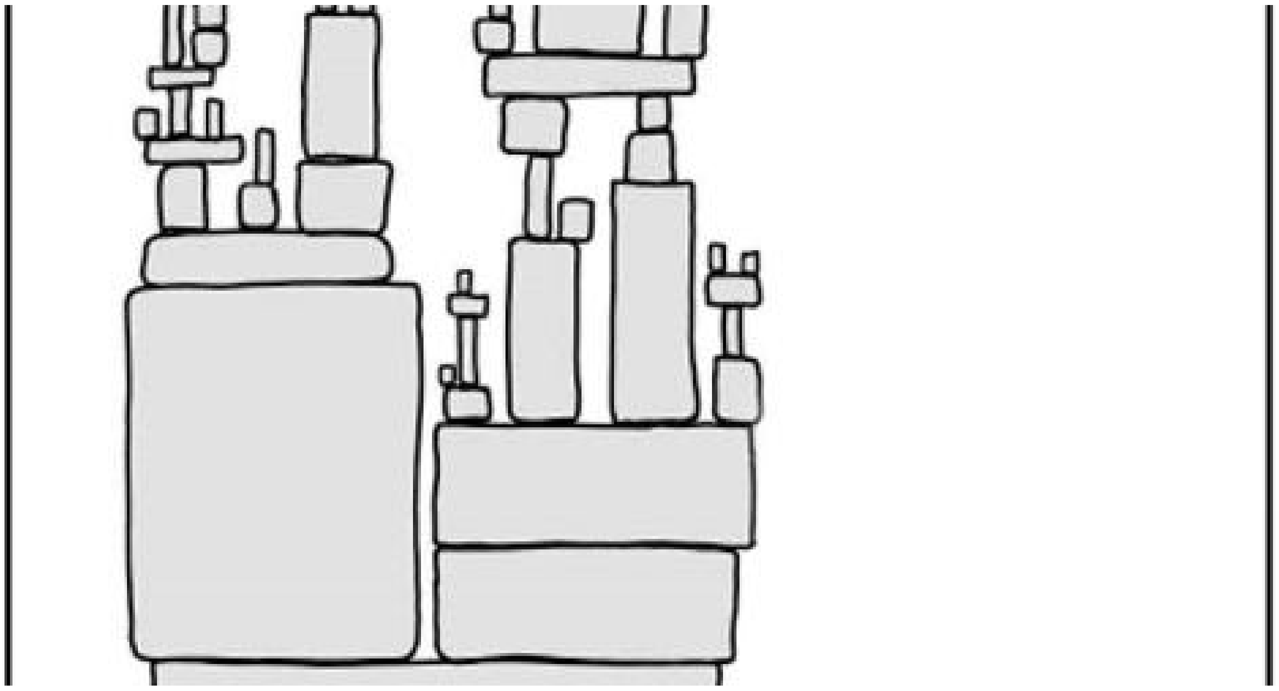
Recommended from Medium

Imila Maheshan

Introduction to the MERN Stack: Building a Full-Stack CRUD Project

The MERN stack, a combination of MongoDB, Express, React, and Node.js, has quickly become a preferred choice for full-stack developers...

6d ago



Louis Trinh

Mongoose Schema Modeling for E-commerce Products: Best Practices and Considerations

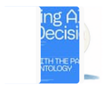
Mongoose Schema Model for Products in E-commerce

Jun 8 4

Lists

Stories to Help You Grow as a Software Developer

19 stories · 1476 saves



data science and AI

40 stories · 285 saves

Himanshu Yadav

Axios in React

Axios is a popular JavaScript library used for making HTTP requests from both browsers and Node.js applications. It provides an easy-to-use...

Jul 7 3

habtesoft

Getting Started with Sequelize in Node.js: A Step-by-Step Tutorial

Sequelize is a powerful ORM (Object-Relational Mapping) for Node.js that makes it easy to work with SQL databases like MySQL, PostgreSQL...

Sep 4 24

Zihan Lin

Master JSON Web Tokens (JWT): How to Use JWT for Secure Authentication and Beyond

What is a JWT?

Aug 26 101

Arunangshu Das

Top 10 Advanced Pagination Techniques with Mongoose

Pagination is a crucial part of any application that handles large datasets, ensuring that data is presented in manageable chunks and…

Jul 2 2

See more recommendations