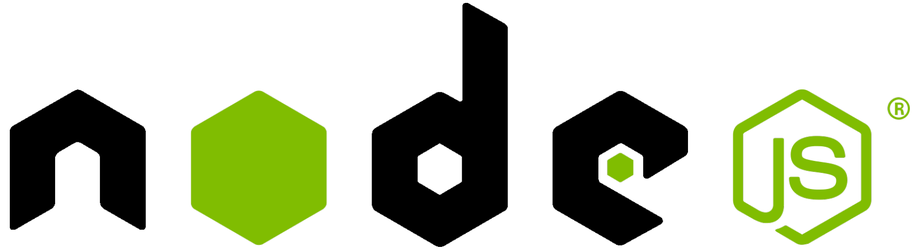
JAVASCRIPT

BACK-END(API WITH DATABASE)

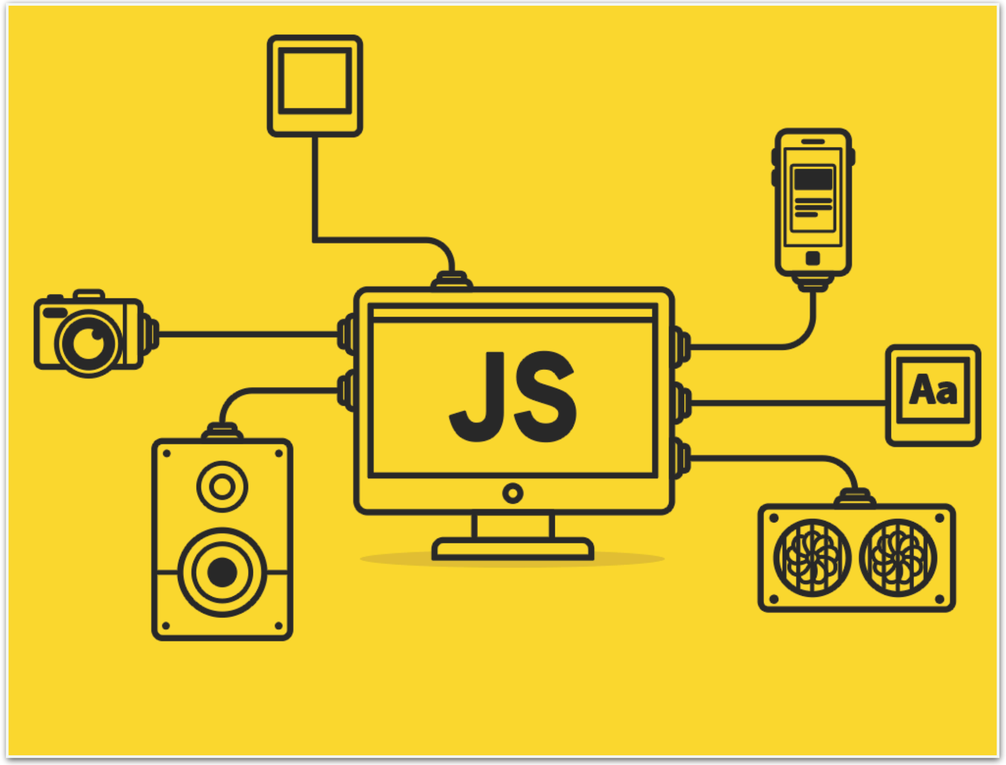
Anirach Mingkhwan

Anirach.m@fitm.kmutnb.ac.th

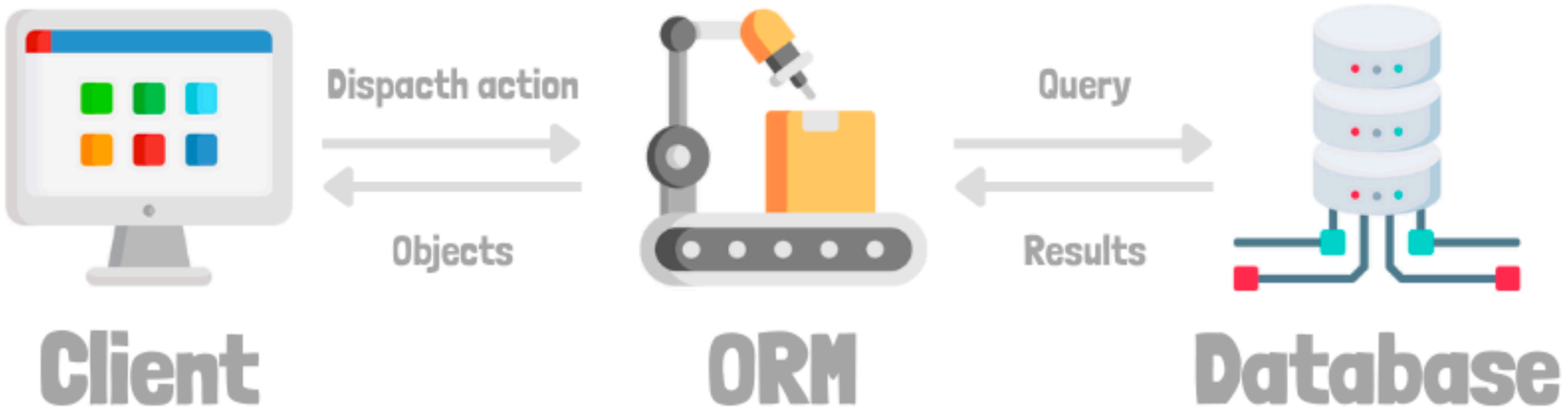


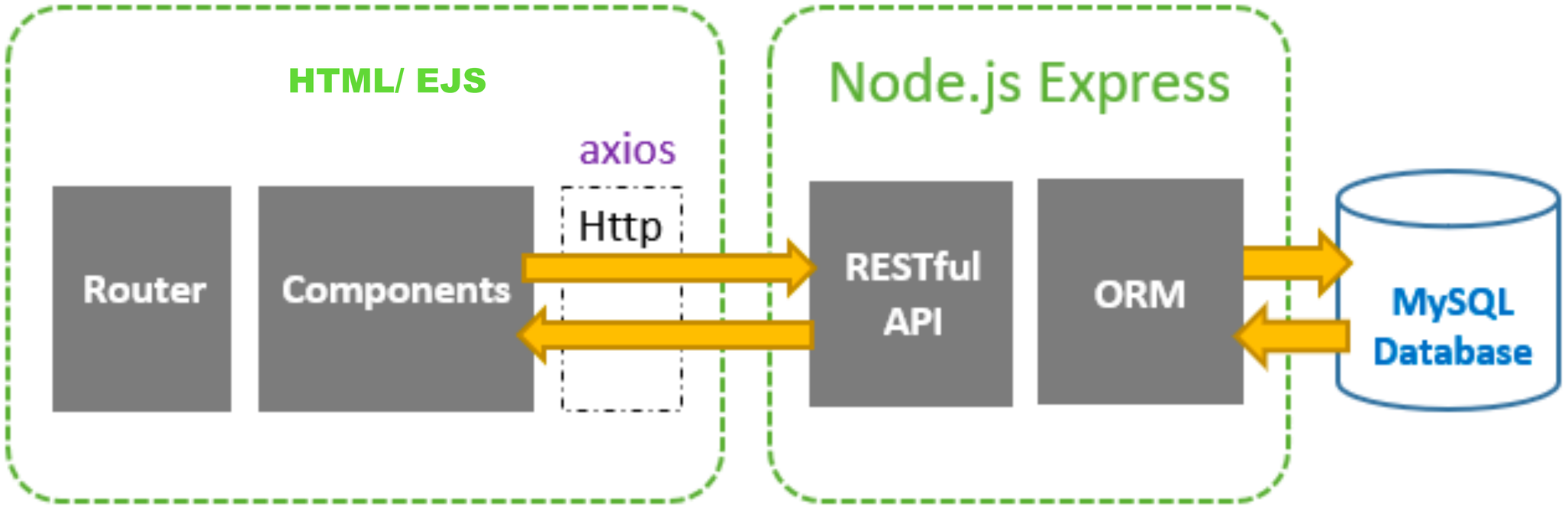
OUTLINE

- ORM
- SQUELize
- Axios
- EJS
- HTML Front-End



ORM-SQUELIZE


Dispacth action
Objects
Query
Results
Client
ORM
Database


HTML/ EJS
Node.js Express
axios
Router
Components
Http
RESTful API
ORM
MySQL Database

Relational database (such as PostgreSQL or MySQL)

ID	FIRST_NAME	LAST_NAME	PHONE
1	John	Connor	+16105551234
2	Matt	Makai	+12025555689
3	Sarah	Smith	+19735554512
...	...	...	...

Python objects

```
class Person:
  first_name = "John"
  last_name = "Connor"
  phone_number = "+16105551234"
```

```
class Person:
  first_name = "Matt"
  last_name = "Makai"
  phone_number = "+12025555689"
```

```
class Person:
  first_name = "Sarah"
  last_name = "Smith"
  phone_number = "+19735554512"
```

ORMs provide a bridge between **relational database tables, relationships and fields** and **Python objects**

Sequelize v6 - stable API References Slack GitHub Security Search ⌘ K

Sequelize

Sequelize is a modern TypeScript and Node.js ORM for Oracle, Postgres, MySQL, MariaDB, SQLite and SQL Server, and more. Featuring solid transaction support, relations, eager and lazy loading, read replication and more.

Getting Started | API Reference

Upgrade to v6 | Support us

https://sequelize.org/

sequelize

6.28.0 • Public • Published 2 months ago

Readme | Code Beta | 16 Dependencies | 5,553 Dependents | 601 Versions

Sequelize

npm v6.28.0 | CI failing | downloads 6.7M/month | contributors 381 | backers 16 | sponsors 28 | merged PRs 3.8K | semantic-release | License MIT

Sequelize is an easy-to-use and promise-based Node.js ORM tool for Postgres, MySQL, MariaDB, SQLite, DB2, Microsoft SQL Server, and Snowflake. It features solid transaction support, relations, eager and lazy loading, read replication and more.

Would you like to contribute? Read our contribution guidelines to know more. There are many ways to help! 😃

💻 Getting Started

Ready to start using Sequelize? Head to sequelize.org to begin!

Install

```
> npm i sequelize
```

Repository

github.com/sequelize/sequelize

Homepage

sequelize.org/

❤Fund this package

Weekly Downloads

1,600,147

Version	License
6.28.0	MIT

Unpacked Size	Total Files
2.88 MB	300

API-ORM-SQLITE

src > JS SequlizeSQLiteCRUDBook.js > ...

You, 25 minutes ago | 1 author (You)

```
1   // Description: Node Express REST API with Sequelize and SQLite CRUD Book
2   // npm install express sequelize sqlite3
3   // Run this file with node SequlizeSQLiteCRUDBook.js
4   // Test with Postman
5
6   const express = require('express');
7   const Sequelize = require('sequelize');
8   const app = express();
9
10  // parse incoming requests
11  app.use(express.json());
12
13  // create a connection to the database
14  const sequelize = new Sequelize('database', 'username', 'password', {
15    host: 'localhost',
16    dialect: 'sqlite',
17    storage: './Database/SQBooks.sqlite'
18  });
19
```

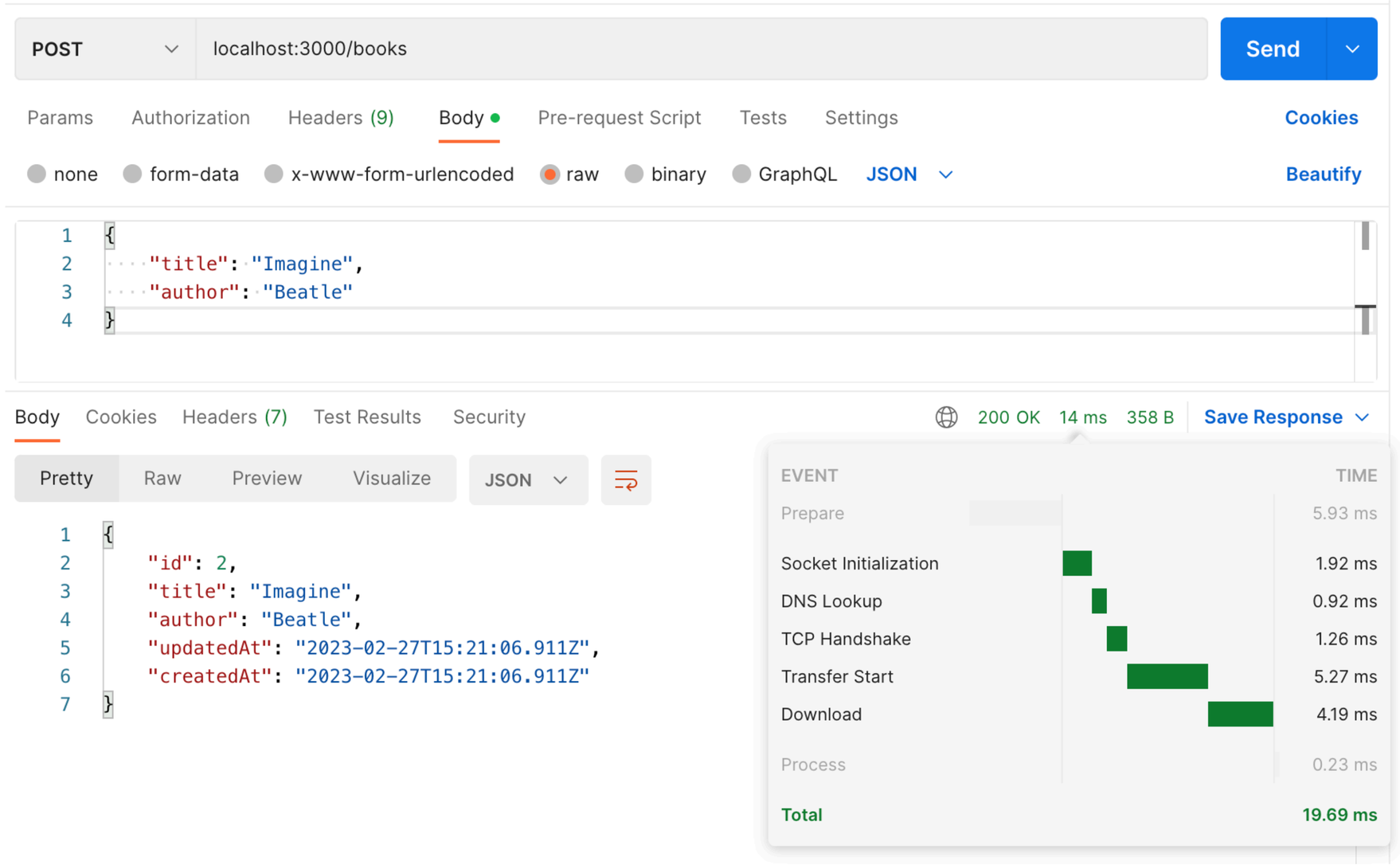
```
19
20  // define the Book model
21  const Book = sequelize.define('book', {
22    id: {
23      type: Sequelize.INTEGER,
24      autoIncrement: true,
25      primaryKey: true
26    },
27    title: {
28      type: Sequelize.STRING,
29      allowNull: false
30    },
31    author: {
32      type: Sequelize.STRING,
33      allowNull: false
34    }
35  });
36
37  // create the books table if it doesn't exist
38  sequelize.sync();
39
```

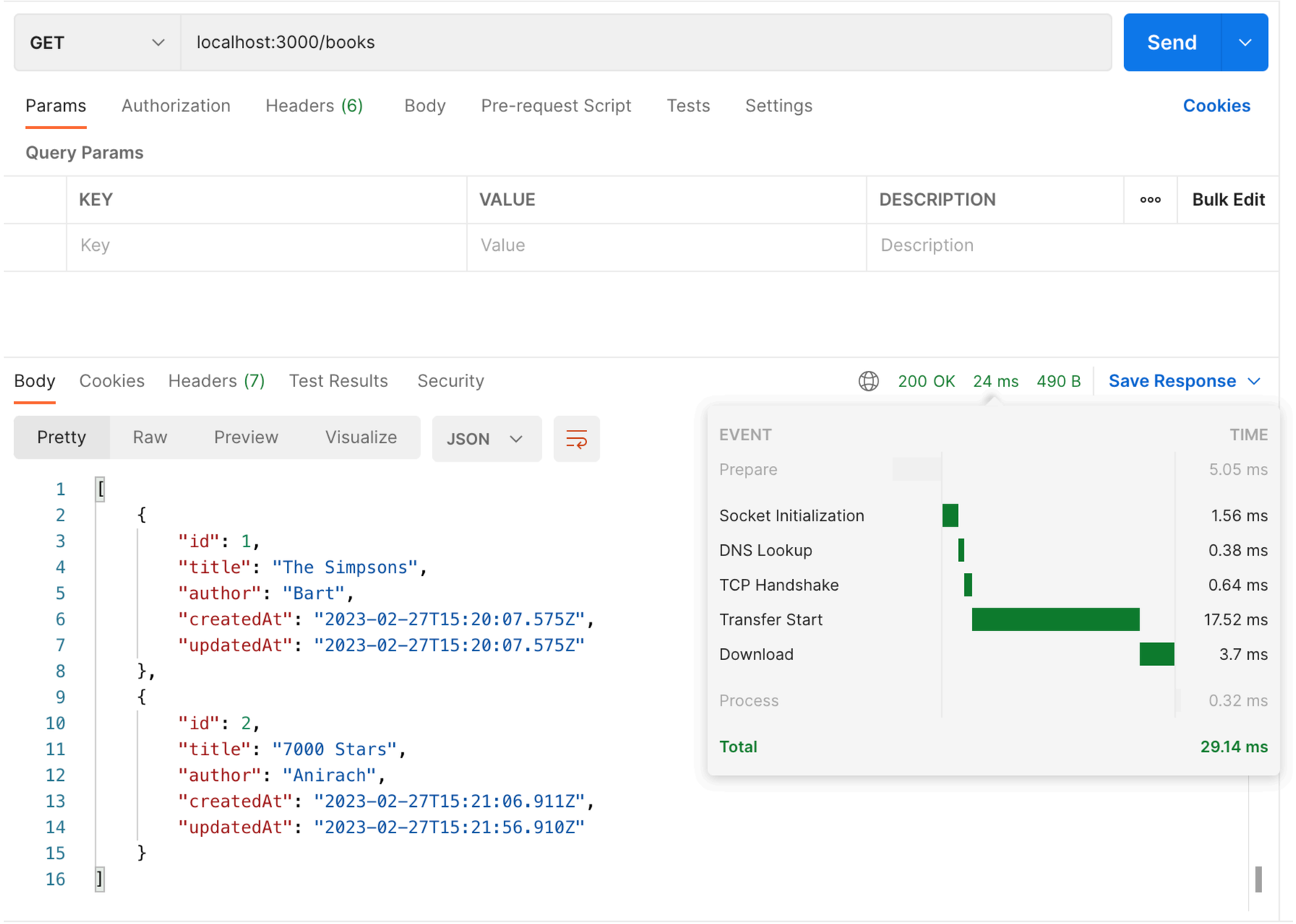
```
39
40  // route to get all books
41  app.get('/books', (req, res) => {
42    Book.findAll().then(books => {
43      res.json(books);
44    }).catch(err => {
45      res.status(500).send(err);
46    });
47  });
48
49  // route to get a book by id
50  app.get('/books/:id', (req, res) => {
51    Book.findByPk(req.params.id).then(book => {
52      if (!book) {
53        res.status(404).send('Book not found');
54      } else {
55        res.json(book);
56      }
57    }).catch(err => {
58      res.status(500).send(err);
59    });
60  });
61
```

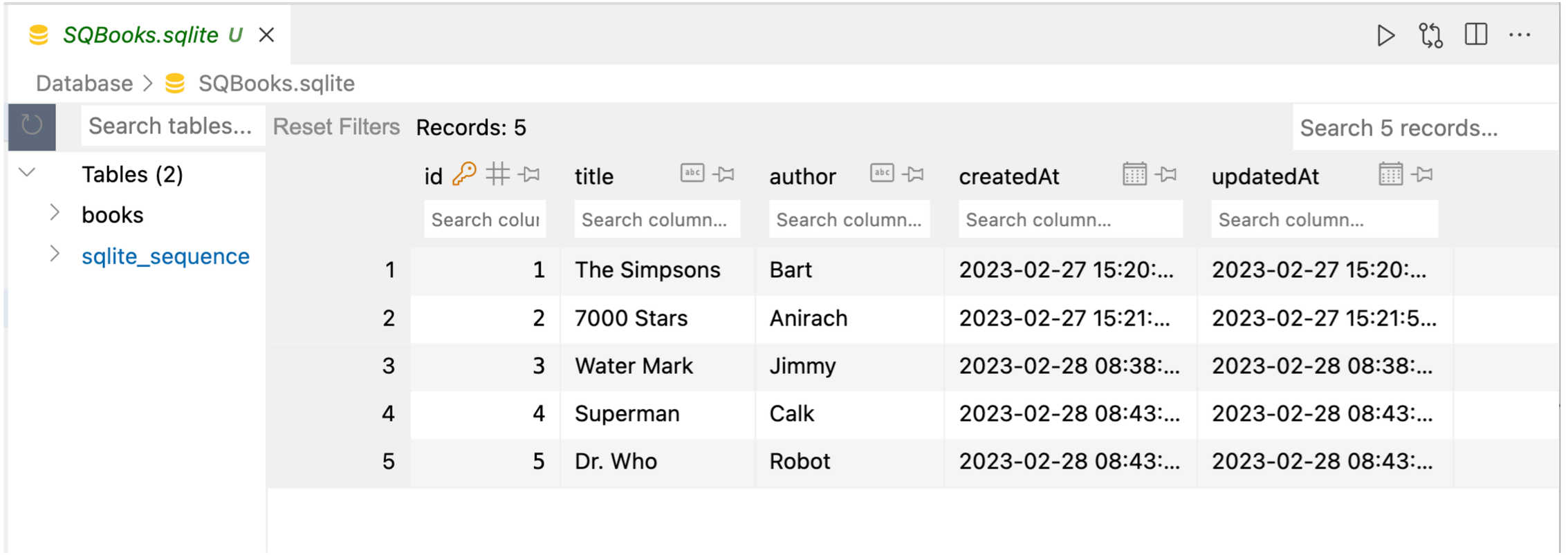
```
61
62  // route to create a new book
63  app.post('/books', (req, res) => {
64    Book.create(req.body).then(book => {
65      res.send(book);
66    }).catch(err => {
67      res.status(500).send(err);
68    });
69  });
70
71  // route to update a book
72  app.put('/books/:id', (req, res) => {
73    Book.findByPk(req.params.id).then(book => {
74      if (!book) {
75        res.status(404).send('Book not found');
76      } else {
77        book.update(req.body).then(() => {
78          res.send(book);
79        }).catch(err => {
80          res.status(500).send(err);
81        });
82      }
83    }).catch(err => {
84      res.status(500).send(err);
85    });
86  });
```

```
87
88  // route to delete a book
89  app.delete('/books/:id', (req, res) => {
90    Book.findByPk(req.params.id).then(book => {
91      if (!book) {
92        res.status(404).send('Book not found');
93      } else {
94        book.destroy().then(() => {
95          res.send({});
96        }).catch(err => {
97          res.status(500).send(err);
98          });
99      }
100 }).catch(err => {
101   res.status(500).send(err);
102 });
103 });
104
105 // start the server
106 const port = process.env.PORT || 3000;
107 app.listen(port, () => console.log(`Listening on port ${port}...`));
108
```

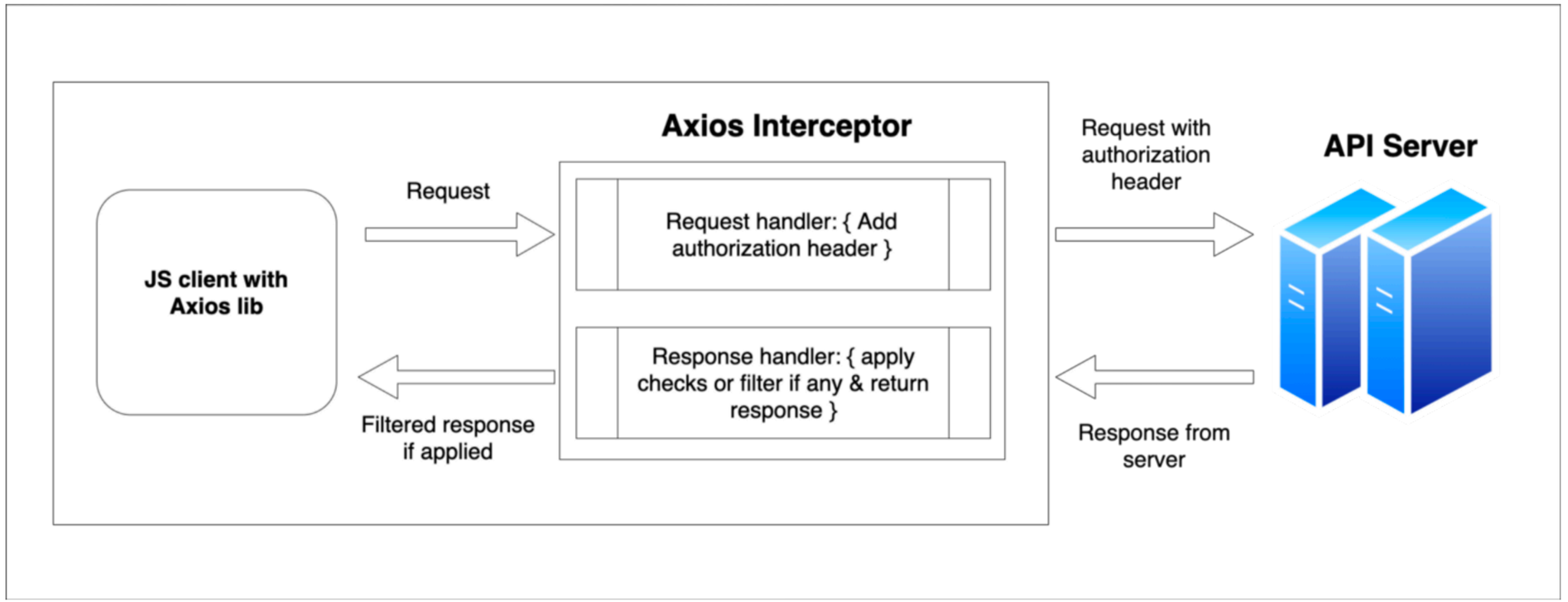
TEST WITH POSTMAN

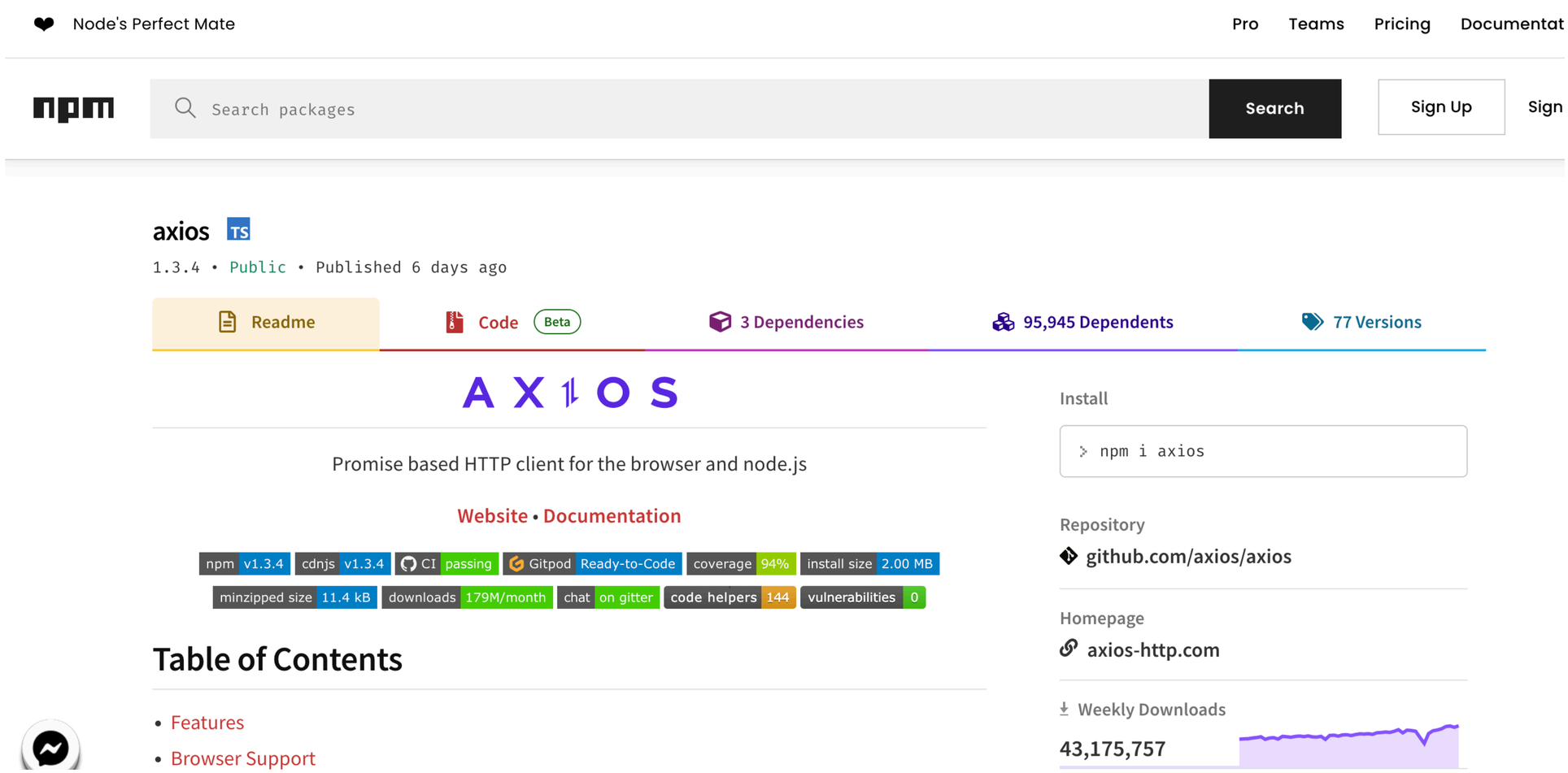

POST
localhost:3000/books
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"title": "Imagine",
"author": "Beatle"
}
Body
Cookies
Headers (7)
Test Results
Security
200 OK
14 ms
358 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": 2,
"title": "Imagine",
"author": "Beatle",
"updatedAt": "2023-02-27T15:21:06.911Z",
"createdAt": "2023-02-27T15:21:06.911Z"
}
EVENT
TIME
Prepare
5.93 ms
Socket Initialization
1.92 ms
DNS Lookup
0.92 ms
TCP Handshake
1.26 ms
Transfer Start
5.27 ms
Download
4.19 ms
Process
0.23 ms
Total
19.69 ms


GET
localhost:3000/books
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (7)
Test Results
Security
200 OK
24 ms
490 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 1,
"title": "The Simpsons",
"author": "Bart",
"createdAt": "2023-02-27T15:20:07.575Z",
"updatedAt": "2023-02-27T15:20:07.575Z"
},
{
"id": 2,
"title": "7000 Stars",
"author": "Anirach",
"createdAt": "2023-02-27T15:21:06.911Z",
"updatedAt": "2023-02-27T15:21:56.910Z"
}
]
EVENT
TIME
Prepare
5.05 ms
Socket Initialization
1.56 ms
DNS Lookup
0.38 ms
TCP Handshake
0.64 ms
Transfer Start
17.52 ms
Download
3.7 ms
Process
0.32 ms
Total
29.14 ms


SQBooks.sqlite U
Database > SQBooks.sqlite
Search tables...
Reset Filters
Records: 5
Search 5 records...
Tables (2)
books
sqlite_sequence
id
title
author
createdAt
updatedAt
Search colu
Search column...
Search column...
Search column...
Search column...
1 1 The Simpsons Bart 2023-02-27 15:20:... 2023-02-27 15:20:...
2 2 7000 Stars Anirach 2023-02-27 15:21:... 2023-02-27 15:21:5...
3 3 Water Mark Jimmy 2023-02-28 08:38:... 2023-02-28 08:38:...
4 4 Superman Calk 2023-02-28 08:43:... 2023-02-28 08:43:...
5 5 Dr. Who Robot 2023-02-28 08:43:... 2023-02-28 08:43:...

AXIOS


Axios Interceptor
JS client with Axios lib
Request
Request handler: { Add authorization header }
Response handler: { apply checks or filter if any & return response }
Filtered response if applied
Request with authorization header
API Server
Response from server



https://www.npmjs.com/package/axios

AXIOS

English

Getting Started

Promise based HTTP client for the browser and node.js

What is Axios?

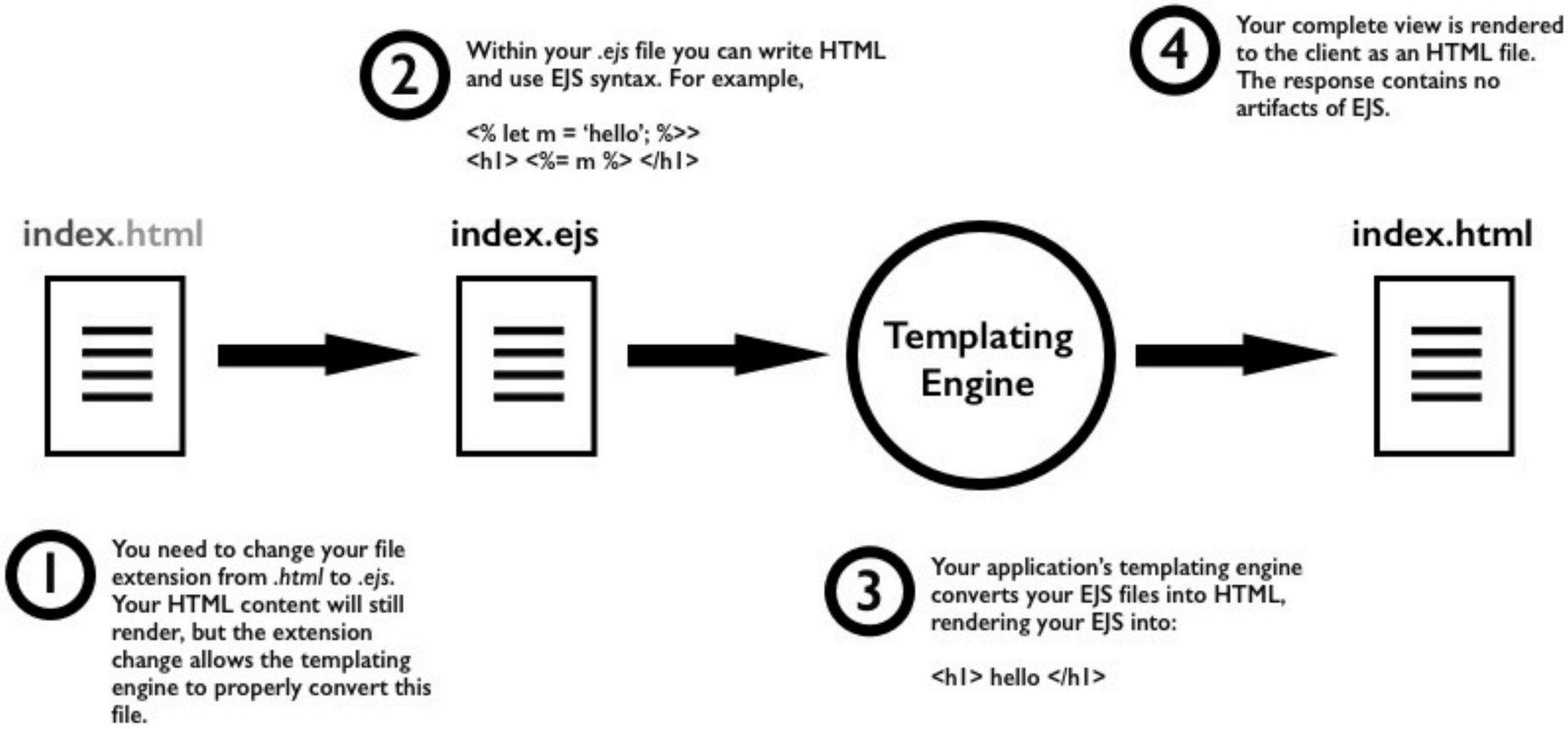
Axios is a *promise-based* HTTP Client for `node.js` and the browser. It is *isomorphic* (= it can run in the browser and nodejs with the same codebase). On the server-side it uses the native node.js `http` module, while on the client (browser) it uses XMLHttpRequests.

Features

- Make XMLHttpRequests from the browser
- Make http requests from node.js
- Supports the Promise API
- Intercept request and response
- Transform request and response data
- Cancel requests
- Timeouts
- Query parameters serialization with support for nested entries
- Automatic request body serialization to:

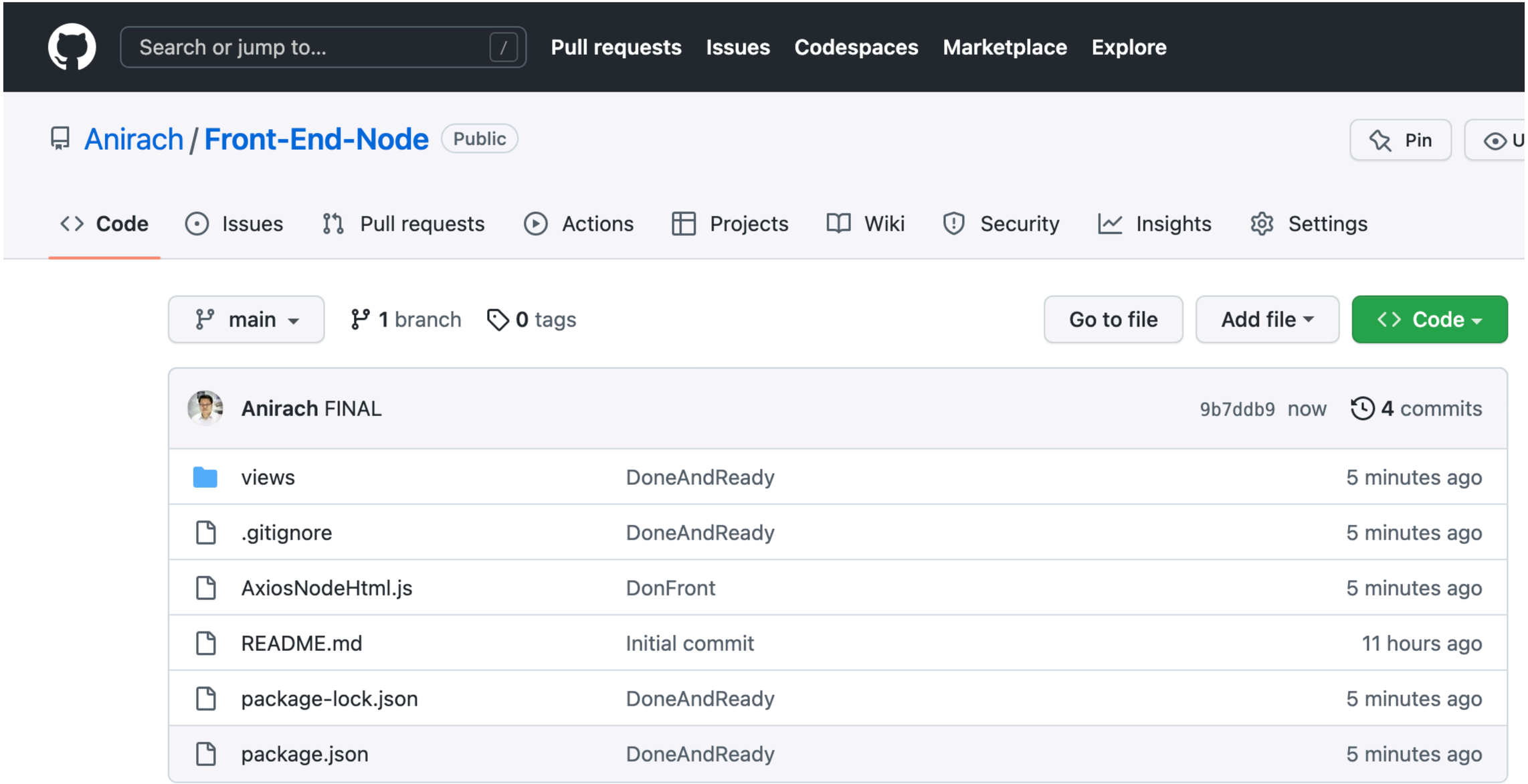
https://axios-http.com/docs/intro

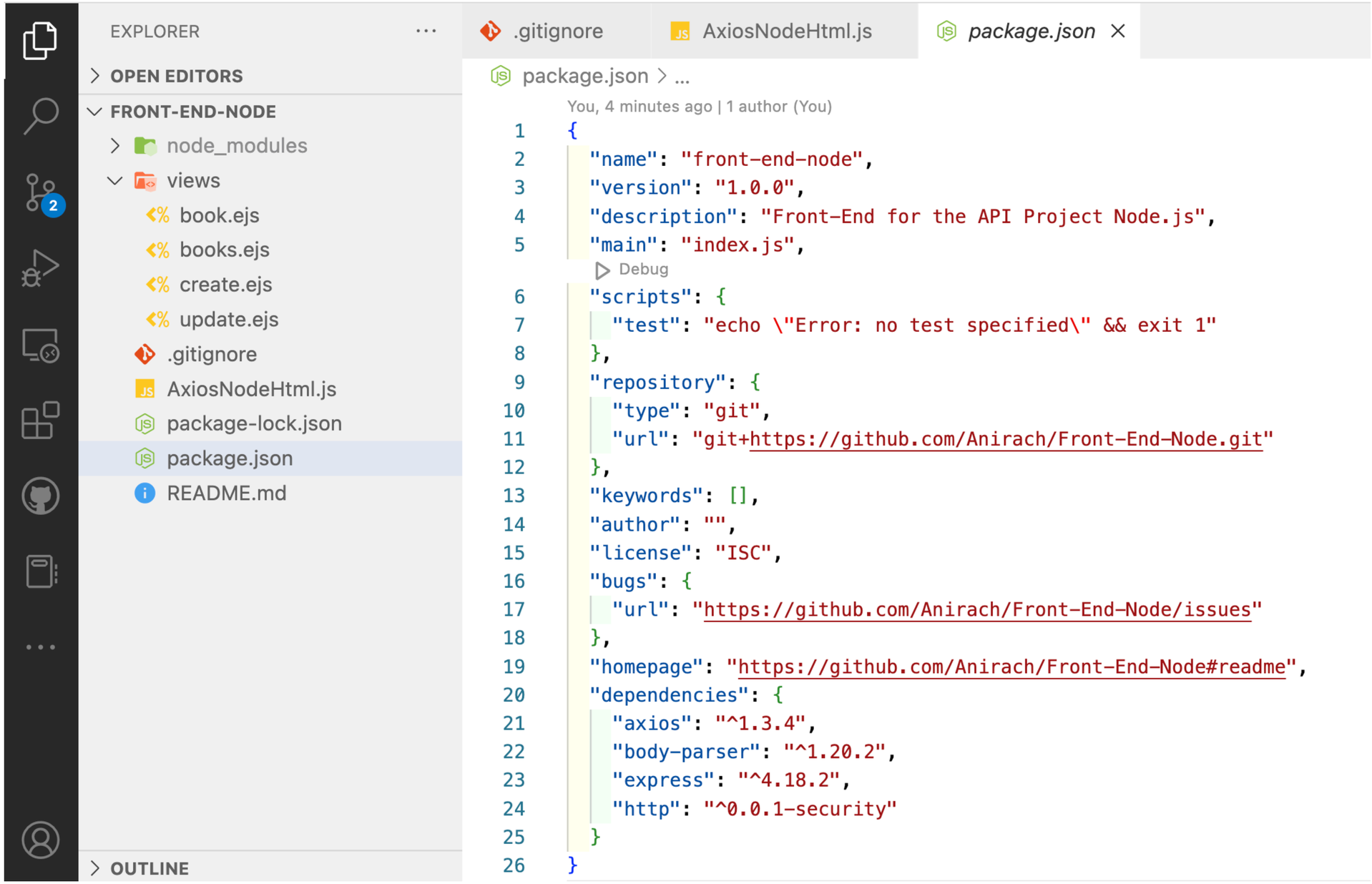
EJS



https://blog.logrocket.com/how-to-use-ejs-template-node-js-application/

HTML AXIOS FRONT-END


Search or jump to... /
Pull requests
Issues
Codespaces
Marketplace
Explore
Anirach / Front-End-Node
Public
Pin
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
Anirach FINAL
9b7ddb9 now
4 commits
views DoneAndReady 5 minutes ago
.gitignore DoneAndReady 5 minutes ago
AxiosNodeHtml.js DonFront 5 minutes ago
README.md Initial commit 11 hours ago
package-lock.json DoneAndReady 5 minutes ago
package.json DoneAndReady 5 minutes ago


EXPLORER
OPEN EDITORS
FRONT-END-NODE
node_modules
views
book.ejs
books.ejs
create.ejs
update.ejs
.gitignore
AxiosNodeHtml.js
package-lock.json
package.json
README.md
OUTLINE
.gitignore
AxiosNodeHtml.js
package.json
package.json > ...
You, 4 minutes ago | 1 author (You)
Debug
{
"name": "front-end-node",
"version": "1.0.0",
"description": "Front-End for the API Project Node.js",
"main": "index.js",
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1"
},
"repository": {
"type": "git",
"url": "git+https://github.com/Anirach/Front-End-Node.git"
},
"keywords": [],
"author": "",
"license": "ISC",
"bugs": {
"url": "https://github.com/Anirach/Front-End-Node/issues"
},
"homepage": "https://github.com/Anirach/Front-End-Node#readme",
"dependencies": {
"axios": "^1.3.4",
"body-parser": "^1.20.2",
"express": "^4.18.2",
"http": "^0.0.1-security"
}
}

AxiosNodeHtml.js > ...

```
1   // Description: Node.js HTML client
2   // requires: npm install express ejs axios body-parser
3
4   const express = require('express');
5   const axios = require('axios');
6   const app = express();
7   var bodyParser = require('body-parser');
8
9   // Base URL for the API
10  //const base_url = "https://api.example.com";
11  const base_url = "http://localhost:3000";
12
13  // Set the template engine
14  app.set('view engine', 'ejs');
15  app.use(bodyParser.json());
16  app.use(bodyParser.urlencoded({ extended: false }));
17
```

```
17
18  // Serve static files
19  app.use(express.static(__dirname + '/public'));
20
21  app.get("/", async (req, res) => {
22      try {
23          const response = await axios.get(base_url + '/books');
24          res.render("books", { books: response.data });
25      } catch (err) {
26          console.error(err);
27          res.status(500).send('Error');
28      }
29  });
30
31  app.get("/book/:id", async (req, res) => {
32      try {
33          const response = await axios.get(base_url + '/books/' + req.params.id);
34          res.render("book", { book: response.data });
35      } catch (err) {
36          console.error(err);
37          res.status(500).send('Error');
38      }
39  });
40
```

```
40
41  app.get("/create", (req, res) => {
42      res.render("create");
43  });
44
45  app.post("/create", async (req, res) => {
46      try {
47          const data = { title: req.body.title, author: req.body.author };
48          await axios.post(base_url + '/books', data);
49          res.redirect("/");
50      } catch (err) {
51          console.error(err);
52          res.status(500).send('Error');
53      }
54  });
55
56  app.get("/update/:id", async (req, res) => {
57      try {
58          const response = await axios.get(
59          base_url + '/books/' + req.params.id);
60          res.render("update", { book: response.data });
61      } catch (err) {
62          console.error(err);
63          res.status(500).send('Error');
64      }
65  });
66
```

```
66
67  app.post("/update/:id", async (req, res) => {
68      try {
69          const data = { title: req.body.title, author: req.body.author };
70          await axios.put(base_url + '/books/' + req.params.id, data);
71          res.redirect("/");
72      } catch (err) {
73          console.error(err);
74          res.status(500).send('Error');
75      }
76  });
77
78  app.get("/delete/:id", async (req, res) => {
79      try {
80          await axios.delete(base_url + '/books/' + req.params.id);
81              res.redirect("/");
82      } catch (err) {
83          console.error(err);
84          res.status(500).send('Error');
85      }
86  });
87
88  app.listen(5500, () => {
89              console.log('Server started on port 5500');
90              });
91
```

views > book.ejs > ...

```
1   <!-- book.ejs -->
2   <!DOCTYPE html>
3   <html>
4   <head>
5     <title>Book</title>
6   </head>
7   <body>
8     <h1><%= book.title %></h1>
9     <p>Author: <%= book.author %></p>
10    <a href="/update/<%= book.id %>">Edit</a>
11    <a href="/delete/<%= book.id %>">Delete</a>
12    <a href="/">Back to List</a>
13  </body>
14  </html>
15
```

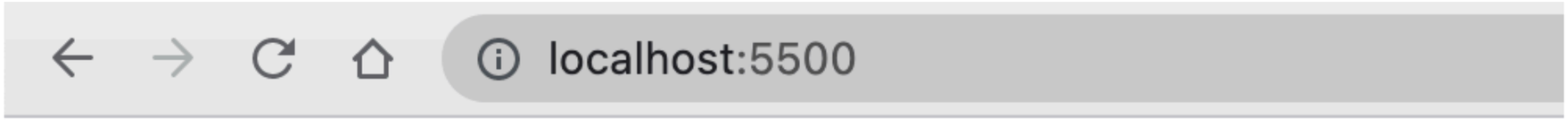
views > <% books.ejs > ...

```ejs
<!-- books.ejs -->
<!DOCTYPE html>
<html>
<head>
  <title>Books</title>
</head>
<body>
  <h1>Books</h1>
  <table>
    <tr>
      <th>Title</th>
      <th>Author</th>
      <th>Actions</th>
    </tr>
    <% for(var i=0; i < books.length; i++) { %>
    <tr>
      <td><%= books[i].title %></td>
      <td><%= books[i].author %></td>
      <td>
        <a href="/book/<%= books[i].id %>">View</a>
        <a href="/update/<%= books[i].id %>">Edit</a>
        <a href="/delete/<%= books[i].id %>">Delete</a>
      </td>
    </tr>
    <% } %>
  </table>
  <a href="/create">Create New Book</a>
</body>
</html>
```

```
<!-- create.ejs -->
<!DOCTYPE html>
<html>
<head>
  <title>Create Book</title>
</head>
<body>
  <h1>Create Book</h1>
  <form method="post" action="/create">
    <label for="title">Title:</label>
    <input type="text" id="title" name="title">
    <br>
    <label for="author">Author:</label>
    <input type="text" id="author" name="author">
    <br>
    <input type="submit" value="create">
  </form>
  <a href="/">Back to List</a>
</body>
</html>
```

views > <% update.ejs > ...

```
1  <!-- update.ejs -->
2  <!DOCTYPE html>
3  <html>
4  <head>
5    <title>Update Book</title>
6  </head>
7  <body>
8    <h1>Update Book</h1>
9    <form method="post" action="/update/<%= book.id %>">
10     <label for="title">Title:</label>
11     <input type="text" id="title" name="title" value="<%= book.title %>">
12     <br>
13     <label for="author">Author:</label>
14     <input type="text" id="author" name="author" value="<%= book.author %>">
15     <br>
16     <input type="submit" value="Update">
17   </form>
18 <a href="/book/<%= book.id %>">View</a>
19 <a href="/delete/<%= book.id %>">Delete</a>
20 <a href="/">Back to List</a>
21 </body>
22 </html>
```



Books

Title	Author	Actions
The Simpsons	Bart Simson	View Edit Delete
7000 Stars	Anirach	View Edit Delete
Water Mark	Jimmy	View Edit Delete
Dr. Who	Robot	View Edit Delete
Runway	plan	View Edit Delete

Create New Book

THANK YOU

- ORM
- SQUELize
- Axios
- EJS
- HTML Front-End