

---

# JAVASCRIPT

## BACK-END(NODEJS, EXPRESS, API,CRUD-POSTMAN)

---

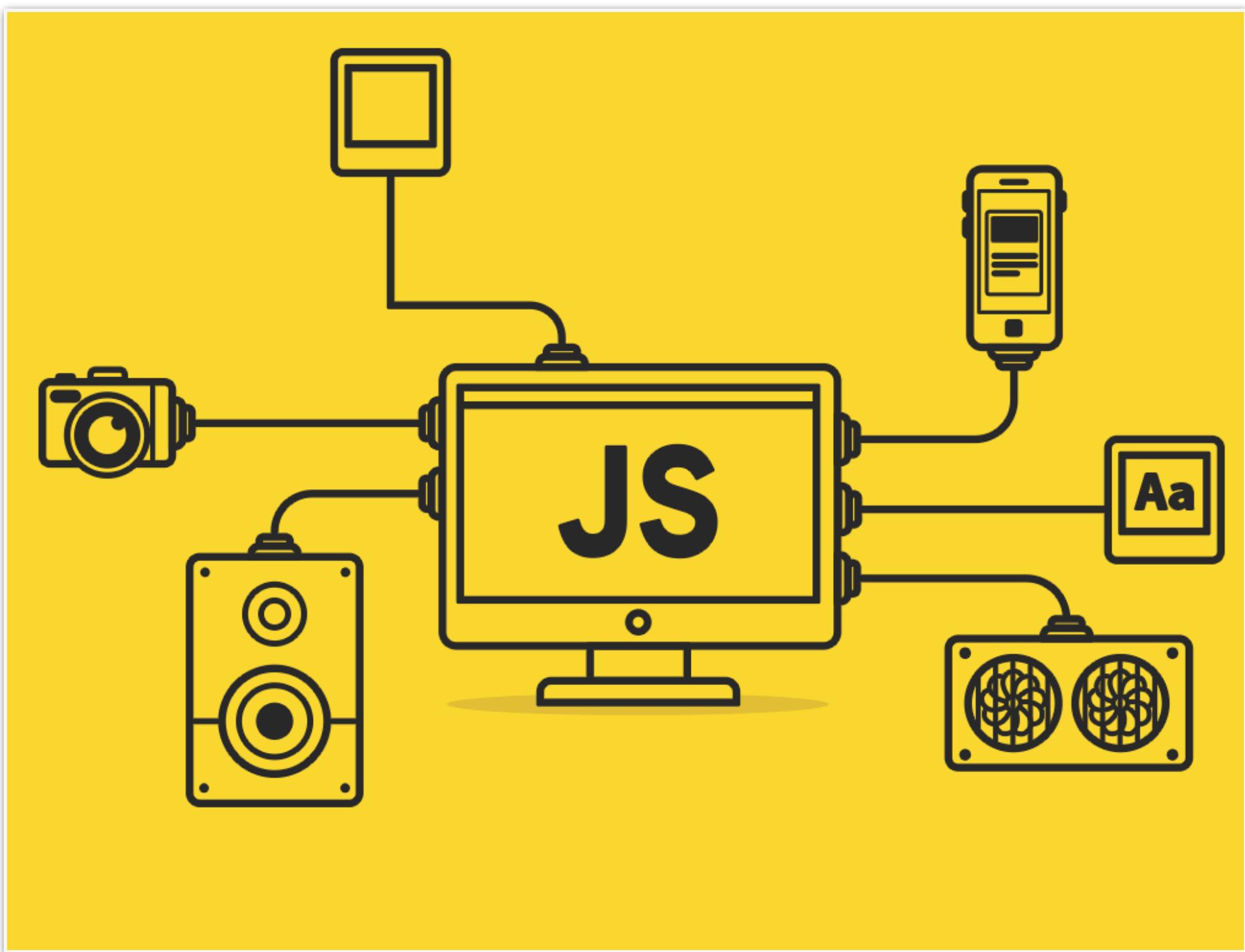
Anirach Mingkhwан

[Anirach.m@fitm.kmutnb.ac.th](mailto:Anirach.m@fitm.kmutnb.ac.th)



# OUTLINE

- 3-TIERS Concepts
- API Architect
- Postman
- CRUD with Postman

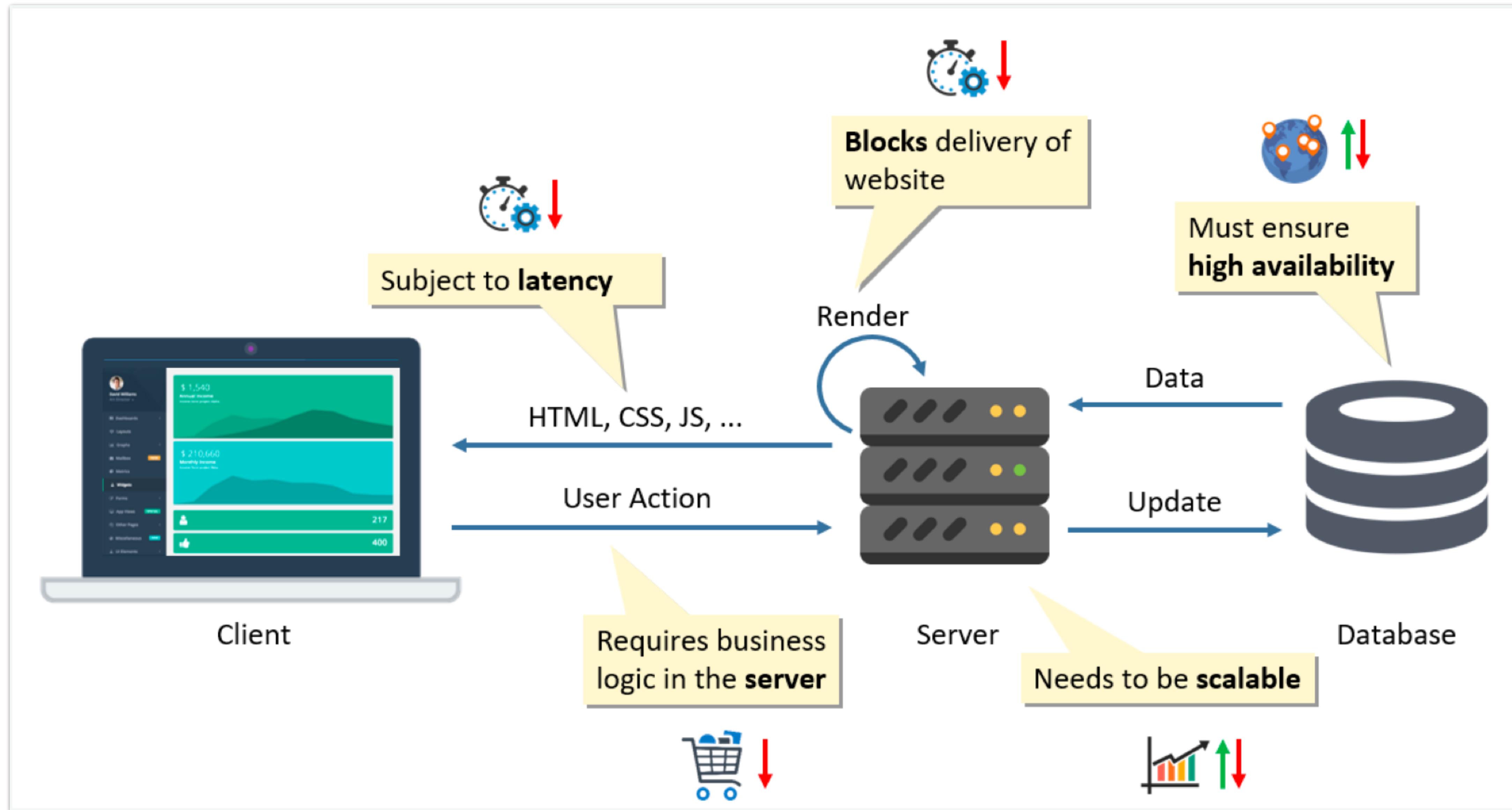


---

# 3-TIERS CONCEPT

---

# The Classic 3-Tier Architecture

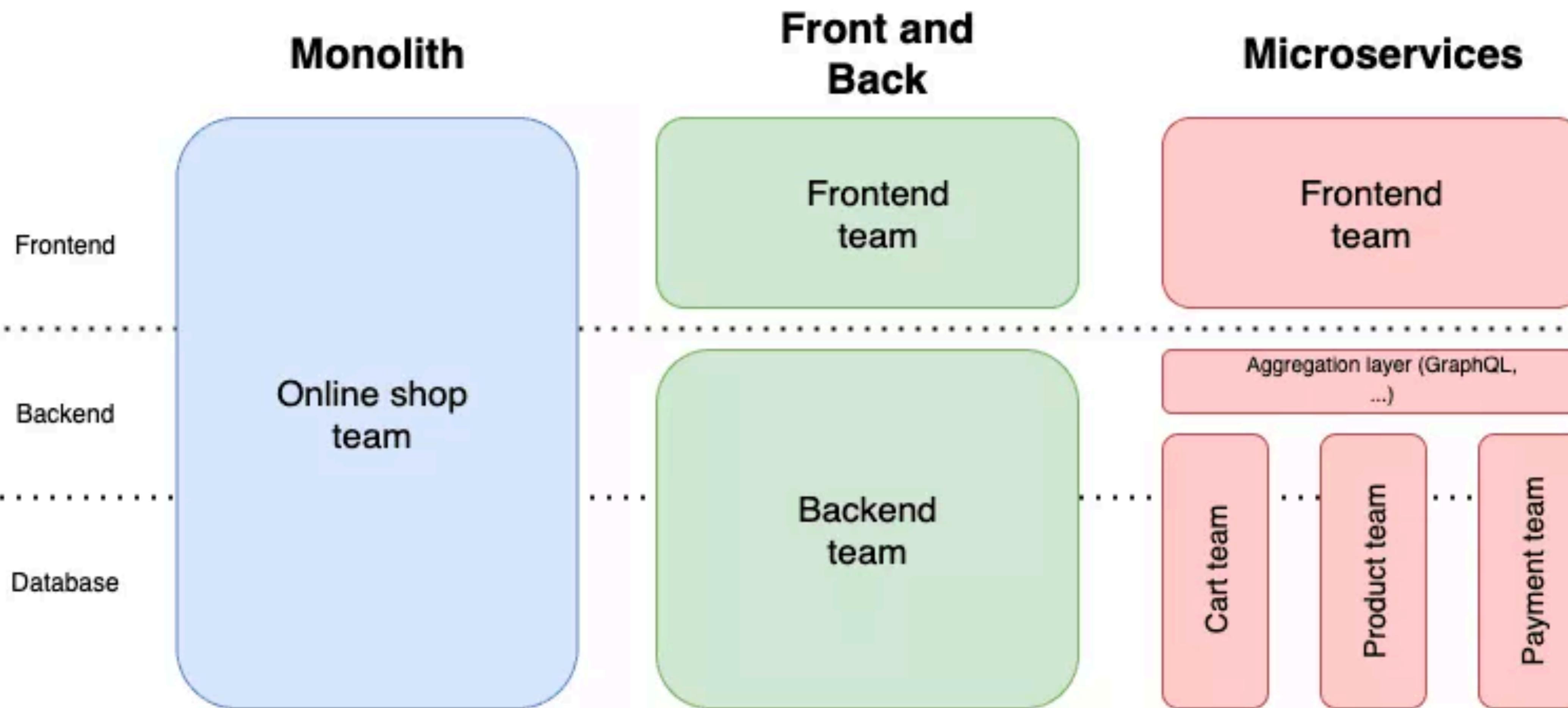


---

# API ARCHITECTURE

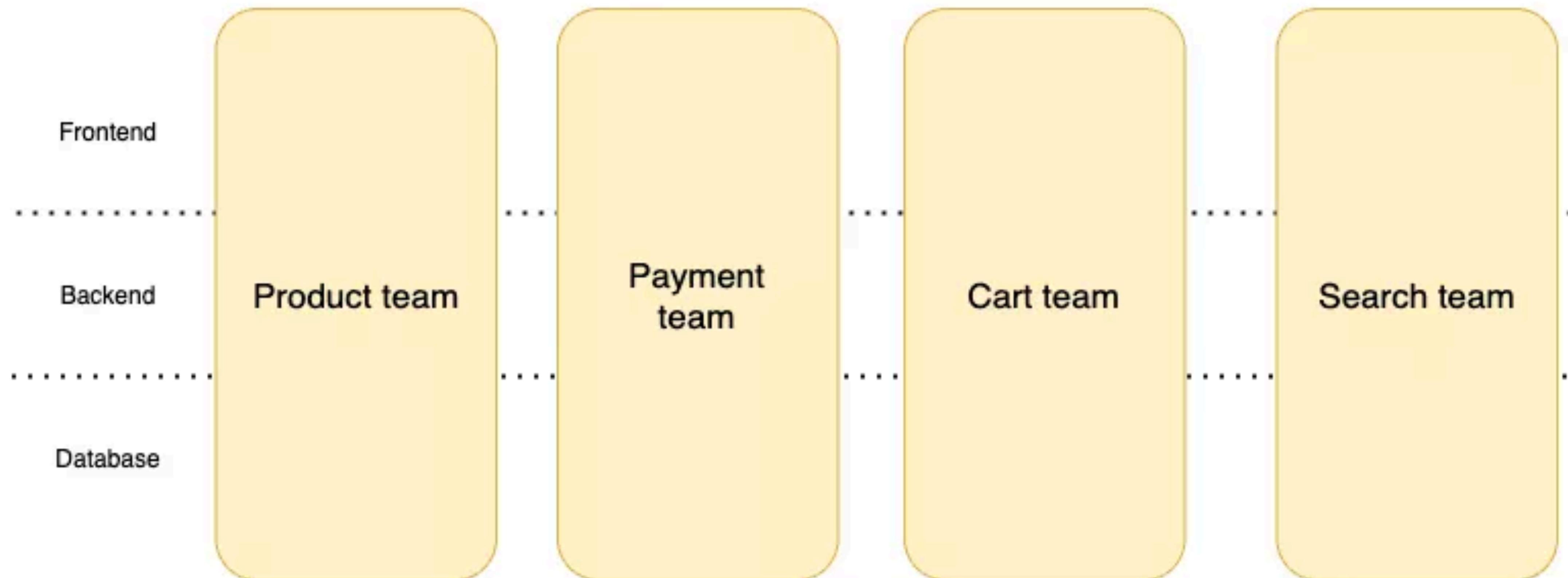
---

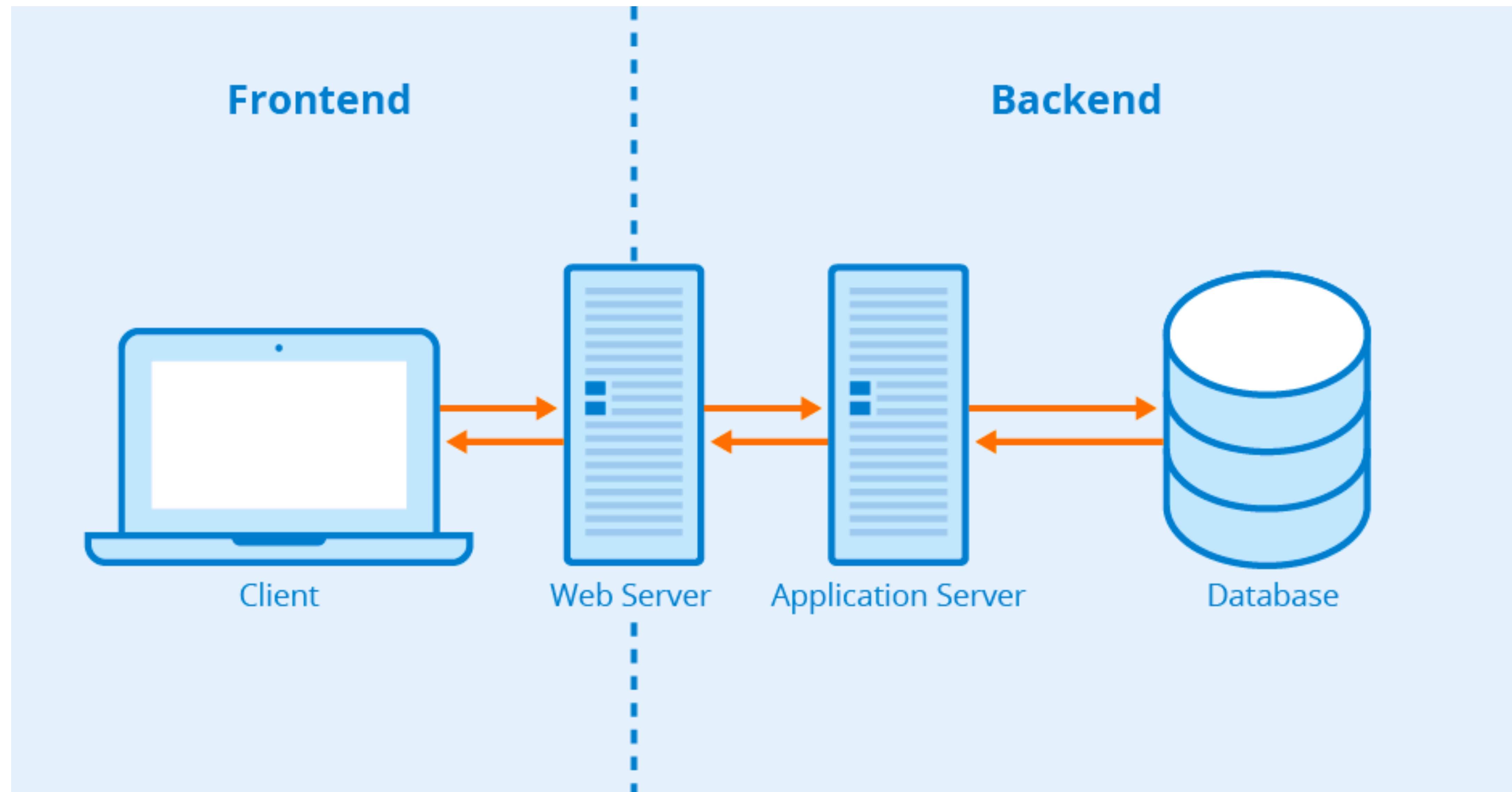
# API Platform Architecture

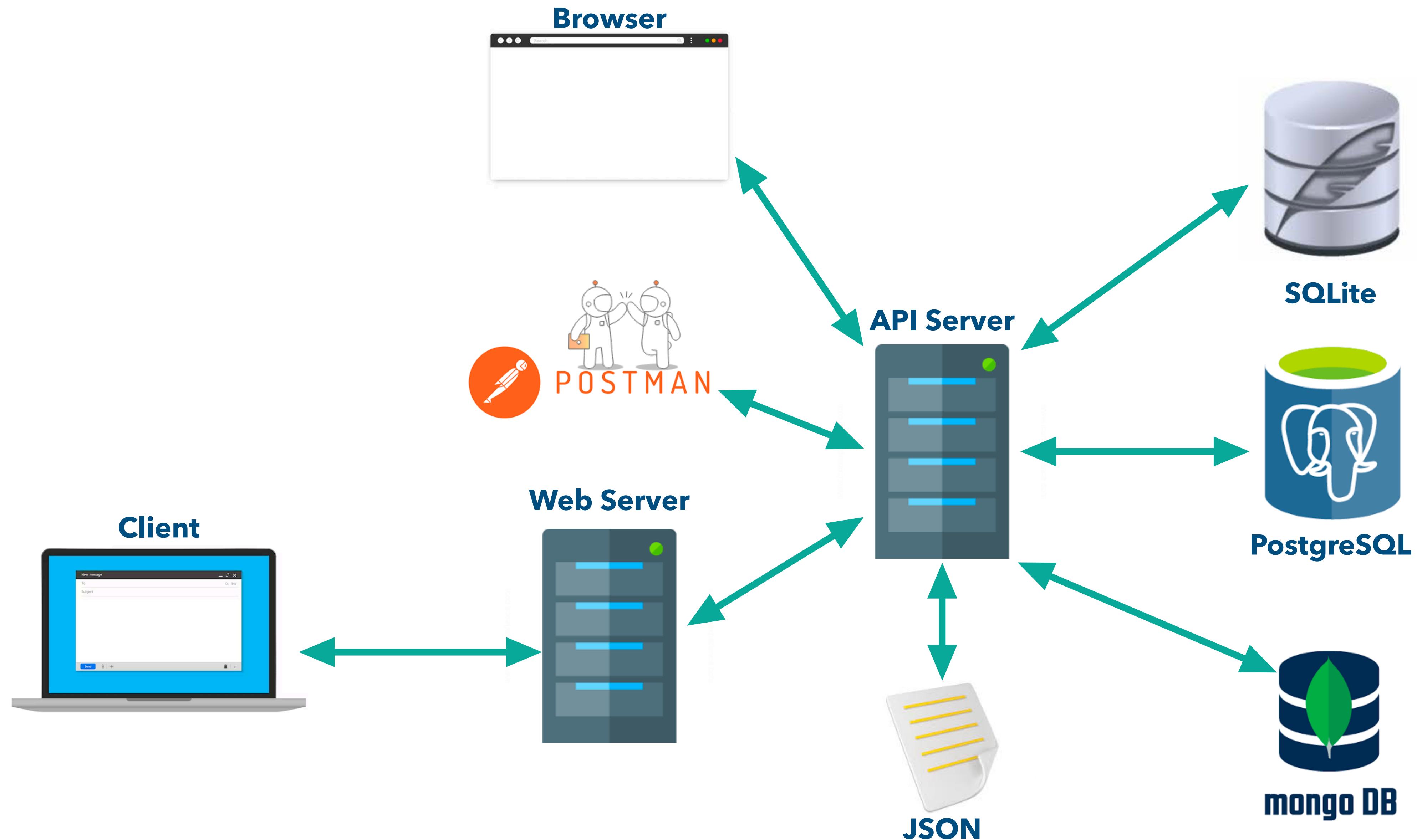


# API Platform Architecture

## Micro-frontends and microservices







---

# RECAP

---

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER View:** Shows the project structure. The root folder is "NODEREST". Inside it are "docs", "node\_modules", "NodeExpressRE...", and a "src" folder containing "CRUDBookNoD...", "index.js", ".env", ".gitignore", "package-lock.json", and "package.json". There are also "README.md" and ".git" icons.
- OPEN EDITORS View:** Shows the "package.json" file is open in the editor.
- Editor View:** Displays the content of the "package.json" file. The file is a JSON object with the following structure:

```
1 {  
2   "name": "noderest",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "format": "prettier --write *.js",  
8     "start": "node src/index.js",  
9     "dev": "nodemon src/index.js"  
10    },  
11    "repository": {  
12      "type": "git",  
13      "url": "git+https://github.com/Anirach/NodeREST.git"  
14    },  
15    "keywords": [],  
16    "author": "",  
17    "license": "ISC",  
18    "bugs": {  
19      "url": "https://github.com/Anirach/NodeREST/issues"  
20    },  
21    "homepage": "https://github.com/Anirach/NodeREST#readme",  
22    "devDependencies": {  
23      "nodemon": "^2.0.20",  
24      "prettier": "^2.8.3"  
25    },  
26    "dependencies": {  
27      "dotenv": "^16.0.3",  
28      "express": "^4.18.2"  
29    }  
30  }
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure. The **NODEREST** folder contains **docs**, **node\_modules**, **NodeExpressRE...**, **src** (which contains **CRUDBookNoD...** and **index.js**), **.env**, **.gitignore**, **package-lock.json**, **package.json** (marked with a **M** icon), and **README.md**. There are also icons for **4** files and **5** folders.
- Search Bar:** Contains the text "NodeREST".
- Editor (Top Right):** Displays the **index.js** file content. The code initializes an Express app, sets a port, and starts it, logging the message "Example app listening at http://localhost:\${port}" to the terminal.
- Terminal (Bottom Right):** Shows the command `> npm run start` and its output: `> noderest@1.0.0 start` and `> node src/index.js`. It then displays the message "Example app listening at http://localhost:3000".

# NODEMON

Never Patronize Mothers

Pro Teams Pricing Documentation

**npm** Search packages Search Sign Up Sign In

**nodemon** DT  
2.0.20 • Public • Published 4 months ago

[Readme](#) [Code](#) Beta [10 Dependencies](#) [4,219 Dependents](#) [241 Versions](#)



Install  
`npm i nodemon`

Repository  
[github.com/remy/nodemon](#)

Homepage  
[nodemon.io](#)

[Fund this package](#)

Weekly Downloads  
 5,247,014

Version License  
 2.0.20 MIT

Unpacked Size Total Files  
 ~~~ ~~~

**nodemon**

nodemon is a tool that helps develop Node.js based applications by automatically restarting the node application when file changes in the directory are detected.

nodemon does **not** require *any* additional changes to your code or method of development.

nodemon is a replacement wrapper for `node`. To use `nodemon`, replace the word `node` on the

**npm i -D nodemon**

The screenshot illustrates a Node.js application development environment. On the left is the Explorer sidebar showing files like package.json, index.js, .env, and NODEREST modules. The main area shows an open editor for index.js with the following code:

```
You, 3 days ago | 1 author (You)
1 require("dotenv").config();
2
3 const express = require("express");
4 const app = express();
5 const port = process.env.PORT || 3000;
6
7 app.get("/", (req, res) => {
8   res.send("Hello World!");
9 }
10
11 app.listen(port, () => {
12   console.log(`Example app listening at http://localhost:\${port}`);
13 }
14 )
```

The terminal below shows the command `node index.js` being run and the output "Example app listening at http://localhost:5500". To the right, a browser window displays the message "Hello World!" at `localhost:5500`.

# NODEMON EXAMPLE

The screenshot illustrates a Node.js development setup using a code editor (VS Code) and a terminal.

**Code Editor (VS Code):**

- EXPLORER:** Shows the project structure with files like package.json, index.js, package-lock.json, and .env.
- OPEN EDITORS:** Shows index.js is currently open.
- TERMINAL:** Shows the command `npm run dev` being run, which triggers nodemon to start the application.

```

    package.json M    index.js M X    package-lock.json    .env
    index.js > app.get("/") callback
    You, now | 1 author (You)
    1 require("dotenv").config();
    2 ...
    3 const express = require("express");
    4 const app = express();
    5 const port = process.env.PORT||3000;
    6
    7 app.get("/", (req, res) => {
    8   res.send("Hello World! Hello");      You, now • Uncommitted changes
    9 }
    10
    11 app.listen(port, () => {
    12   console.log(`Example app listening at http://localhost:${port}`);
    13 })
  
```

**Browser:** A Chrome browser window shows the URL `localhost:5500` and displays the text "Hello World! Hello".

**Terminal Output:**

```

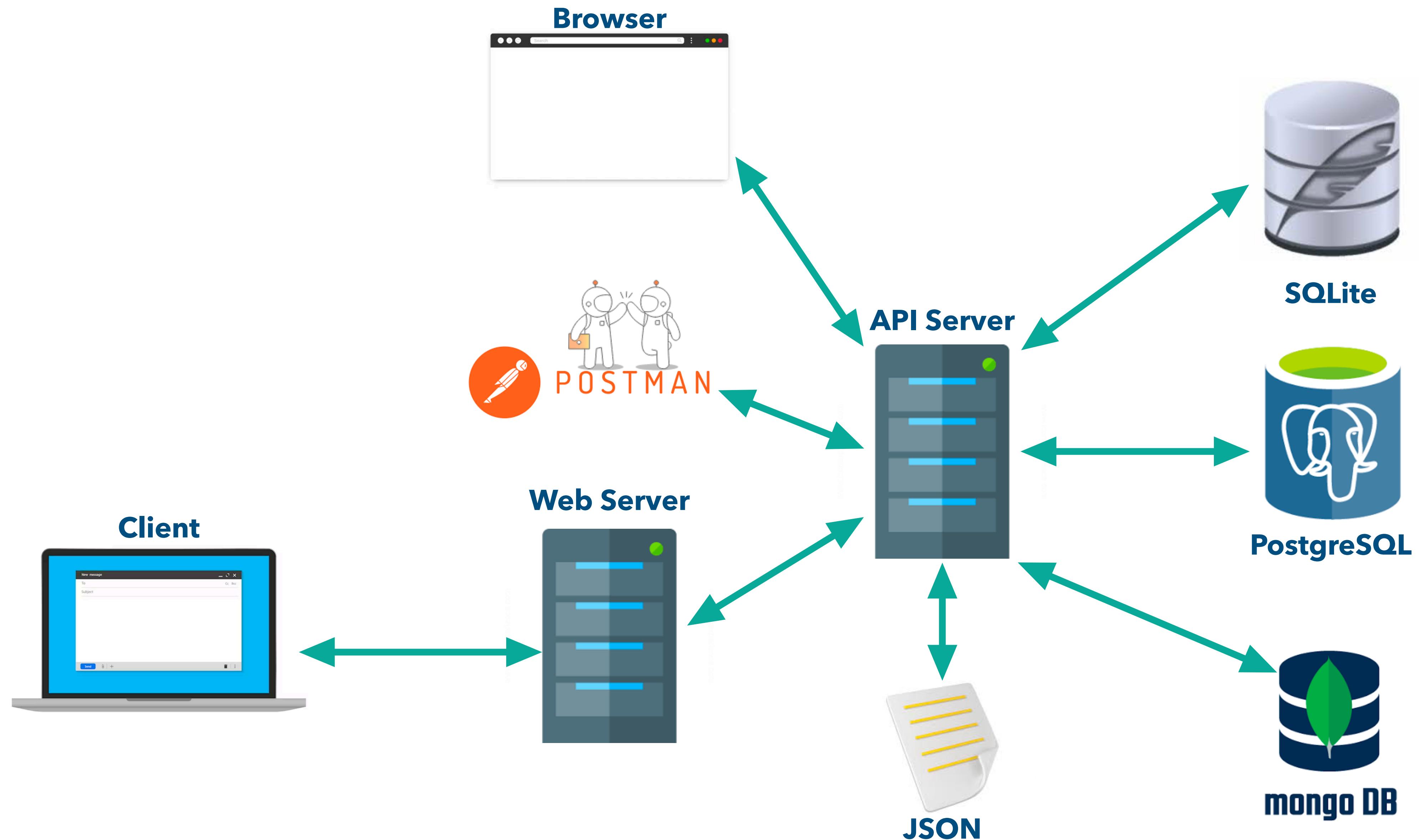
> npm run dev
> noderest@1.0.0 dev
> nodemon index.js

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Example app listening at http://localhost:5500
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Example app listening at http://localhost:5500
  
```

---

# BACK-END

---



The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** pane on the left:
  - OPEN EDITORS**: Shows a file named `CRUDBookNoDB.js`.
  - NODEREST**: Shows a folder structure with `docs`, `node_modules`, `NodeExpressRE...`, `.env`, `.gitignore`, and files `CRUDBookNoDB.js`, `index.js`, `package-lock.json`, `package.json`, and `README.md`. The `index.js` file is currently selected.
  - OUTLINE**
  - TIMELINE**
  - NPM SCRIPTS**: Shows scripts `format`, `start`, and `dev` defined in the `package.json` file.
- EDITOR** pane on the right:
  - File: `CRUDBookNoDB.js`
  - Content:

```
1 // Description: CRUD Book No DB
2 // npm install express
3 // Run this file with node CRUDBookNoDB.js
4 // Test with Postman
5 require("dotenv").config();
6 const express = require('express');
7 const app = express();

8
9 // parse incoming requests
10 app.use(express.json());

11
12 // sample data
13 let books = [
14   {
15     id: 1,
16     title: 'Book 1',
17     author: 'Author 1'
18   },
19   {
20     id: 2,
21     title: 'Book 2',
22     author: 'Author 2'
23   },
24   {
25     id: 3,
26     title: 'Book 3',
27     author: 'Author 3'
28   }
29 ];
30
```
- TERMINAL** pane at the bottom: Not visible in the screenshot.

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** pane on the left:
  - OPEN EDITORS**: Shows two files: `CRUDBookNoDB.js` (selected) and `index.js`.
  - NODEREST**: Shows project structure with `docs`, `node_modules`, and `NodeExpressRE...`. A blue circle with the number **4** is next to it.
  - `.env`
  - `.gitignore`
  - `CRUDBookNoDB.js` (selected)
  - `index.js`
  - `package-lock.json`
  - `package.json`
  - `README.md`
- EDITOR** pane in the center:
  - File: `CRUDBookNoDB.js`
  - Content:

```
30
31 // route to get all books
32 app.get('/books', (req, res) => {
33   res.json(books);
34 });
35
36 // route to get a book by id
37 app.get('/books/:id', (req, res) => {
38   const book = books.find(b => b.id === parseInt(req.params.id));
39   if (!book) res.status(404).send('Book not found');
40   res.json(book);
41 });
42
43 // route to create a new book
44 app.post('/books', (req, res) => {
45   const book = {
46     id: books.length + 1,
47     title: req.body.title,
48     author: req.body.author
49   };
50   books.push(book);
51   res.send(book);
52 });
53
```
- TERMINAL** pane at the bottom: Shows a single line of text: `PS D:\NodeJS\NodeExpressRESTAPI>`

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** pane on the left:
  - OPEN EDITORS**: Shows two files: `CRUDBookNoDB.js` (selected) and `index.js`.
  - NODEREST**: Shows directory structure with `docs`, `node_modules`, `NodeExpressRE...`, `.env`, and `.gitignore`.
  - OUTLINE**
  - TIMELINE**
  - NPM SCRIPTS**: Shows `package.json`.
  - format prettier --w...**
- EDITOR** pane in the center:
  - File: `CRUDBookNoDB.js`
  - Content:

```
53
54 // route to update a book
55 app.put('/books/:id', (req, res) => {
56   const book = books.find(b => b.id === parseInt(req.params.id));
57   if (!book) res.status(404).send('Book not found');
58   book.title = req.body.title;
59   book.author = req.body.author;
60   res.send(book);
61 });
62
63 // route to delete a book
64 app.delete('/books/:id', (req, res) => {
65   const book = books.find(b => b.id === parseInt(req.params.id));
66   if (!book) res.status(404).send('Book not found');
67   const index = books.indexOf(book);
68   books.splice(index, 1);
69   res.send(book);
70 });
71
72 const port = process.env.PORT || 3000;
73 app.listen(port, () => console.log(`Listening on port ${port}...`));
```
- TERMINAL** pane at the bottom: Not visible in the screenshot.

The screenshot shows a Node.js application structure in the Explorer sidebar. The project folder contains files like `docs`, `node_modules`, `NodeExpressRE...`, `.env`, `.gitignore`, `CRUDBookNoDB.js`, `index.js`, `package-lock.json`, `package.json`, and `README.md`. The `CRUDBookNoDB.js` file is open in the editor, displaying code to handle incoming requests. The terminal shows the command `node CRUDBookNoDB.js` being run and the message "Listening on port 3000...". A browser window at the bottom shows the URL `localhost:3000/books` and displays the JSON response: `[{"id":1,"title":"Book 1","author":"Author 1"}, {"id":2,"title":"Book 2","author":"Author 2"}, {"id":3,"title":"Book 3","author":"Author 3"}, {"id":4,"title":"Fire places","author":"Goya"}]`

```
// Description: CRUD Book No DB
// npm install express
// Run this file with node CRUDBookNoDB.js
// Test with Postman
require("dotenv").config();
const express = require('express');
const app = express();

// parse incoming requests
app.use(express.json());

// sample data
let books = [
{
  id: 1,
  title: 'Book 1',
  author: 'Author 1'
},
{
  id: 2,
  title: 'Book 2',
  author: 'Author 2'
},
{
  id: 3,
  title: 'Book 3',
  author: 'Author 3'
},
{
  id: 4,
  title: 'Fire places',
  author: 'Goya'
}]
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE

> node CRUDBookNoDB.js  
Listening on port 3000...

localhost:3000/books

```
[{"id":1,"title":"Book 1","author":"Author 1"}, {"id":2,"title":"Book 2","author":"Author 2"}, {"id":3,"title":"Book 3","author":"Author 3"}, {"id":4,"title":"Fire places","author":"Goya"}]
```

---

# POSTMAN

---

The screenshot shows the Postman download page at <https://www.postman.com/downloads/>. The page features a large "Download Postman" heading and a sub-section titled "The Postman app" with download links for Mac Intel Chip and Mac Apple Chip. A "Privacy Policy" and "Terms" link is also present. Below this, there's a note about OS compatibility and links to "Release Notes" and "Product Roadmap". On the right side, a modal dialog from the Postman app is displayed, asking for cookie acceptance. The dialog contains text about cookie storage for site navigation and marketing, with "Accept All Cookies" and "Cookies Settings" buttons.

Download Postman

Download the app to get started using the Postman API Platform today. Or, if you prefer a browser experience, you can try the web version of Postman.

## The Postman app

Download the app to get started with the Postman API Platform.

[Mac Intel Chip](#) [Mac Apple Chip](#)

By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

[Release Notes](#) · [Product Roadmap](#)

Not your OS? Download for Windows (x64) or Linux (x64)

By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts.

[Accept All Cookies](#) [Cookies Settings](#)

<https://www.postman.com/downloads/>

Home Workspaces API Network Explore

Search Postman

Invite Settings Notifications Profile Upgrade

No Environment

**My Workspace**

New Import

Collections

+

Books DB Node API

GET All Books  
GET Book By ID  
PUT Update Book By ID  
POST Create new book  
DEL Delete Book By ID

APIs Environments Mock Servers Monitors Flows History

Open Workspace Overview

Create a new request:

Online Find and Replace Console Cookies Capture requests Runner Trash

---

# CRUD

---



# HTTP Status Codes

## Level 200

200: OK  
201: Created  
202: Accepted  
203: Non-Authoritative Information  
204: No content

## Level 400

400: Bad Request  
401: Unauthorized  
403: Forbidden  
404: Not Found  
409: Conflict

## Level 500

500: Internal Server Error  
501: Not Implemented  
502: Bad Gateway  
503: Service Unavailable  
504: Gateway Timeout  
599: Network Timeout

---

**READ**

---

The screenshot shows a Node.js application structure in the Explorer sidebar and its corresponding code in the main editor area.

**Explorer Sidebar:**

- OPEN EDITORS: CRUDBookNoDB.js (selected)
- NODEREST:
  - docs
  - node\_modules
  - NodeExpressRE... (marked with a dot)
  - .env
  - .gitignore
  - CRUDBookNoDB.js (selected)
- index.js M
- package-lock.json
- package.json M
- README.md

**Main Editor Area:**

```
1 // Description: CRUD Book No DB
2 // npm install express
3 // Run this file with node CRUDBookNoDB.js
4 // Test with Postman
5 require("dotenv").config();
6 const express = require('express');
7 const app = express();
8
9 // parse incoming requests
10 app.use(express.json());
11
12 // sample data
13 let books = [
14   {
15     id: 1,
16     title: 'Book 1',
17     author: 'Author 1'
18   },
19   {
20     id: 2,
21     title: 'Book 2',
22     author: 'Author 2'
23   }
]
```

**Terminal:**

```
> node CRUDBookNoDB.js
Listening on port 3000...
```

**Browser:**

localhost:3000/books

```
[{"id":1,"title":"Book 1","author":"Author 1"}, {"id":2,"title":"Book 2","author":"Author 2"}, {"id":3,"title":"Book 3","author":"Author 3"}, {"id":4,"title":"Fire places","author":"Goya"}]
```

**My Workspace**

New Import

Collections APIs Environments Mock Servers Monitors Flows History

Books DB Node API / All Books

GET All Books

GET Book By ID

+

No Environment

Save ...

Send

GET All Books

GET Book By ID

PUT Update Book By ID

POST Create new book

DEL Delete Book By ID

localhost:3000/books

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (7) Test Results Security

200 OK 27 ms 375 B Save Response

Pretty Raw Preview Visualize JSON

```

1 [
2   {
3     "id": 1,
4     "title": "Book 1",
5     "author": "Author 1"
6   },
7   {
8     "id": 2,
9     "title": "Book 2",
10    "author": "Author 2"
11  },
12  {
13    "id": 3,
14    "title": "Book 3",
15    "author": "Author 3"
16  }
17 ]

```

EVENT TIME

- Prepare 5.89 ms
- Socket Initialization 1.45 ms
- DNS Lookup 0.26 ms
- TCP Handshake 0.51 ms
- Transfer Start 19.51 ms
- Download 4.4 ms
- Process 0.27 ms
- Total 32.26 ms

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** pane on the left:
  - OPEN EDITORS**: Shows two files: `CRUDBookNoDB.js` (selected) and `index.js`.
  - NODEREST**: Shows project structure including `docs`, `node_modules`, `NodeExpressRE...`, `.env`, `.gitignore`, `CRUDBookNoDB.js` (selected), `index.js`, `package-lock.json`, `package.json`, and `README.md`.
  - OUTLINE**
  - TIMELINE**
  - NPM SCRIPTS**: Shows `package.json` with scripts: `format` (using prettier), `start` (using node index.js).
- EDITOR** pane on the right:
  - File: `CRUDBookNoDB.js`
  - Content:

```
30
31 // route to get all books
32 app.get('/books', (req, res) => {
33   res.json(books);
34 });
35
36 // route to get a book by id
37 app.get('/books/:id', (req, res) => {
38   const book = books.find(b => b.id === parseInt(req.params.id));
39   if (!book) res.status(404).send('Book not found');
40   res.json(book);
41 });
42
43 // route to create a new book
44 app.post('/books', (req, res) => {
45   const book = {
46     id: books.length + 1,
47     title: req.body.title,
48     author: req.body.author
49   };
50   books.push(book);
51   res.send(book);
52 });
53
```
- TERMINAL** pane at the bottom: Not visible in the screenshot.

**My Workspace**

New Import

GET All Books    GET Book By ID

No Environment

Collections

Books DB Node API / Book By ID

GET localhost:3000/books/1

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (7) Test Results Security

Pretty Raw Preview Visualize JSON

EVENT TIME

Prepare 3.35 ms

Socket Initialization 1.34 ms

DNS Lookup 0.22 ms

TCP Handshake 0.74 ms

Transfer Start 1.48 ms

Download 1.2 ms

Process 0.07 ms

Total 8.38 ms

200 OK 5 ms 280 B Save Response

1 {  
2 "id": 1,  
3 "title": "Book 1",  
4 "author": "Author 1"  
5 }

APIs

Environments

Mock Servers

Monitors

Flows

History

The screenshot shows the Postman application interface. On the left, there's a sidebar with icons for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main area displays a collection named 'Books DB Node API'. Under this collection, there are several requests: 'GET All Books', 'GET Book By ID' (which is selected and highlighted with an orange border), 'PUT Update Book By ID', 'POST Create new book', and 'DEL Delete Book By ID'. The 'GET Book By ID' request has its URL set to 'localhost:3000/books/1'. Below the request list, there are tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, Settings, Cookies, Body, Cookies, Headers (7), Test Results, and Security. The 'Headers (7)' tab is currently active. The response section shows a status of 200 OK, a duration of 5 ms, and a body size of 280 B. The response body is displayed in JSON format: { "id": 1, "title": "Book 1", "author": "Author 1" }. To the right of the response, a timeline shows the execution steps: Prepare (3.35 ms), Socket Initialization (1.34 ms), DNS Lookup (0.22 ms), TCP Handshake (0.74 ms), Transfer Start (1.48 ms), Download (1.2 ms), Process (0.07 ms), and a total time of 8.38 ms.

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** pane on the left:
  - OPEN EDITORS**: Shows two files: `CRUDBookNoDB.js` (selected) and `index.js`.
  - NODEREST**: Shows project structure including `docs`, `node_modules`, `NodeExpressRE...`, `.env`, `.gitignore`, `CRUDBookNoDB.js` (selected), `index.js`, `package-lock.json`, `package.json`, and `README.md`.
  - OUTLINE**
  - TIMELINE**
  - NPM SCRIPTS**: Shows `package.json` with scripts: `format` (using prettier), `start` (using node index.js).
- EDITOR** pane in the center:
  - File: `CRUDBookNoDB.js`
  - Content:

```
30
31 // route to get all books
32 app.get('/books', (req, res) => {
33   res.json(books);
34 });
35
36 // route to get a book by id
37 app.get('/books/:id', (req, res) => {
38   const book = books.find(b => b.id === parseInt(req.params.id));
39   if (!book) res.status(404).send('Book not found');
40   res.json(book);
41 });
42
43 // route to create a new book
44 app.post('/books', (req, res) => {
45   const book = {
46     id: books.length + 1,
47     title: req.body.title,
48     author: req.body.author
49   };
50   books.push(book);
51   res.send(book);
52 });
53
```
- TERMINAL** pane at the bottom: Not visible in the screenshot.

---

**CREAT**

---

## Step 1

The screenshot shows the Postman interface with the following details:

- My Workspace** tab is active.
- Collections** sidebar is visible.
- Books DB Node API** collection is selected.
- POST Create new book** request is selected.
- Request URL:** `localhost:3000/books`
- Method:** `POST`
- Headers:** (9 items)
  - `Accept`: `/*`
  - `Accept-Encoding`: `gzip, deflate, br`
  - `Connection`: `keep-alive`
  - `Content-Type`: `application/json`
- Body** tab is selected.
- Body Content:** `{}  
{"title": "Fire places",  
"author": "Goya"}  
{}`

## Step 2

The screenshot shows the Postman interface with the following details:

- My Workspace** tab is active.
- Collections** sidebar is visible.
- Books DB Node API** collection is selected.
- POST Create new book** request is selected.
- Request URL:** `localhost:3000/books`
- Method:** `POST`
- Headers:** (9 items)
- Body** tab is selected.
- Body Type:** `JSON`
- Body Content:** `1 {  
2 ... "title": "Fire places",  
3 ... "author": "Goya"  
4 }`

**My Workspace**

New Import

GET All Books | GET Book By ID | POST Create new book | + ... | No Environment | ...

Collections | Books DB Node API | APIs | Environments | Mock Servers | Monitors | Flows | History

**Books DB Node API / Create new book**

**POST** localhost:3000/books

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies Beautify

Body (raw JSON)

```

1 {
2   "title": "Fire places",
3   "author": "Goya"
4 }
```

Body Cookies Headers (7) Test Results Security

200 OK 28 ms 281 B Save Response

EVENT TIME

| EVENT                 | TIME     |
|-----------------------|----------|
| Prepare               | 5.83 ms  |
| Socket Initialization | 1.71 ms  |
| DNS Lookup            | 0.26 ms  |
| TCP Handshake         | 0.51 ms  |
| Transfer Start        | 21.19 ms |
| Download              | 3.58 ms  |
| Process               | 0.21 ms  |
| Total                 | 33.26 ms |

Pretty Raw Preview Visualize JSON

```

1 {
2   "id": 4,
3   "title": "Fire places",
4   "author": "Goya"
5 }
```

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** pane on the left:
  - OPEN EDITORS**: Shows two files: `CRUDBookNoDB.js` (selected) and `index.js`.
  - NODEREST**: Shows directory structure with `docs`, `node_modules`, and `NodeExpressRE...`. A blue circle with the number **4** is next to it.
  - `.env`
  - `.gitignore`
  - `CRUDBookNoDB.js` (selected)
  - `index.js`
  - `package-lock.json`
  - `package.json`
  - `README.md`
- EDITOR** pane in the center:
  - File: `CRUDBookNoDB.js`
  - Content:

```
30
31 // route to get all books
32 app.get('/books', (req, res) => {
33   res.json(books);
34 });
35
36 // route to get a book by id
37 app.get('/books/:id', (req, res) => {
38   const book = books.find(b => b.id === parseInt(req.params.id));
39   if (!book) res.status(404).send('Book not found');
40   res.json(book);
41 });
42
43 // route to create a new book
44 app.post('/books', (req, res) => {
45   const book = {
46     id: books.length + 1,
47     title: req.body.title,
48     author: req.body.author
49   };
50   books.push(book);
51   res.send(book);
52 });
53
```
- TERMINAL** pane at the bottom: Shows a single line of text: `PS D:\Work\NodeJS\NodeExpressRESTAPI>`

---

# UPDATE

---

Step 1

The screenshot shows the Postman interface with the 'Books DB Node API / Update Book By ID' collection selected. The 'PUT Update Book By ID' request is highlighted. The 'Headers' tab is active, displaying the following configuration:

| Key             | Value             | Description |
|-----------------|-------------------|-------------|
| Accept          | /*                |             |
| Accept-Encoding | gzip, deflate, br |             |
| Connection      | keep-alive        |             |
| Content-Type    | application/json  |             |

Step 2

The screenshot shows the Postman interface with the 'Books DB Node API / Update Book By ID' collection selected. The 'PUT Update Book By ID' request is highlighted. The 'Body' tab is active, showing the raw JSON payload:

```

1  ....{
2  ....  "title": "7000 Stars",
3  ....  "author": "Anirach"
4  ....}

```

Below the body, there are options for selecting the data type: none, form-data, x-www-form-urlencoded, raw (selected), binary, GraphQL, and JSON.

**My Workspace**

New Import

GET All Books | GET Book By ID | POST Create new book | PUT Update Book By ID | + ... | No Environment | ▾

Collections | APIs | Environments | Mock Servers | Monitors | Flows | History

**Books DB Node API / Update Book By ID**

PUT localhost:3000/books/2 | Save | ▾ | Send | ▾

Params | Authorization | Headers (9) | **Body** | Pre-request Script | Tests | Settings | Cookies | Beautify

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | **JSON** | ▾

```

1 ...
2 ... "title": "7000 Stars",
3 ... "author": "Anirach"
4 ...

```

Body | Cookies | Headers (7) | Test Results | Security | 200 OK | 5 ms | 283 B | Save Response | ▾

Pretty | Raw | Preview | Visualize | JSON | ▾

```

1 {
2   "id": 2,
3   "title": "7000 Stars",
4   "author": "Anirach"
5 }

```

EVENT | TIME

|                       |                |
|-----------------------|----------------|
| Prepare               | 3.59 ms        |
| Socket Initialization | 0.84 ms        |
| DNS Lookup            | 0.32 ms        |
| TCP Handshake         | 0.42 ms        |
| Transfer Start        | 2.2 ms         |
| Download              | 1.07 ms        |
| Process               | 0.07 ms        |
| <b>Total</b>          | <b>8.48 ms</b> |

**My Workspace**

New Import

GET All Books    GET Book By ID    POST Create new book    PUT Update Book By I ●    +    ...    No Environment

Collections    +    ⚙️    ...

Books DB Node API / All Books

Save    ⚙️    ...

Send

GET localhost:3000/books

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (7) Test Results Security

200 OK 8 ms 425 B Save Response

Pretty Raw Preview Visualize JSON

```

1 [
2   {
3     "id": 1,
4     "title": "Book 1",
5     "author": "Author 1"
6   },
7   {
8     "id": 2,
9     "title": "7000 Stars",
10    "author": "Anirach"
11  },
12  {
13    "id": 3,
14    "title": "Book 3",
15    "author": "Author 3"
16  },
17  {
18    "id": 4,
19    "title": "Fire places",
20    "author": "Goya"
21  }
22 ]

```

EVENT TIME

| Event                 | Time (ms) |
|-----------------------|-----------|
| Prepare               | 7.58 ms   |
| Socket Initialization | 1.21 ms   |
| DNS Lookup            | 0.24 ms   |
| TCP Handshake         | 0.94 ms   |
| Transfer Start        | 1.75 ms   |
| Download              | 3.45 ms   |
| Process               | 0.21 ms   |
| Total                 | 15.36 ms  |

---

# DELETE

---

**My Workspace**

New Import

GET All Books | GET Book By ID | POST Create new b | PUT Update Boo | DEL Delete Book

Books DB Node API / Delete Book By ID

DELETE localhost:3000/books/3

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (7) Test Results Security

200 OK 4 ms 280 B Save Response

EVENT TIME

- Prepare 2.83 ms
- Socket Initialization 0.57 ms
- DNS Lookup 0.25 ms
- TCP Handshake 0.33 ms
- Transfer Start 1.16 ms
- Download 1.05 ms
- Process 0.05 ms

Total 6.21 ms

Collections APIs Environments Mock Servers Monitors Flows History

GET All Books | GET Book By ID | POST Create new book | PUT Update Book By ID | DELETE Delete Book By ID

The screenshot shows the Postman application interface. On the left sidebar, there are several sections: Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main area displays a collection named "Books DB Node API". Inside this collection, there are five endpoints: "GET All Books", "POST Create new book", "PUT Update Book By ID", "GET Book By ID", and "DEL Delete Book By ID". The "GET All Books" endpoint is currently selected. The request details show a GET method for "localhost:3000/books". The response tab shows a status of 200 OK with a response time of 4 ms and a body size of 379 B. The response body is a JSON array of three books:

```
1  [
2    {
3      "id": 1,
4      "title": "Book 1",
5      "author": "Author 1"
6    },
7    {
8      "id": 2,
9      "title": "7000 Stars",
10     "author": "Anirach"
11   },
12   {
13     "id": 4,
14     "title": "Fire places",
15     "author": "Goya"
16   }
17 ]
```

The timeline on the right side of the response details shows the execution steps: Prepare (2.88 ms), Socket Initialization (0.62 ms), DNS Lookup (0.92 ms), TCP Handshake (0.37 ms), Transfer Start (1.05 ms), Download (0.73 ms), Process (0.08 ms), and a total time of 6.62 ms.

# THANK YOU

---

- 3-TIERS Concepts
- API Architect
- Postman
- CRUD with Postman