

CRUD on API

Asst.Prof.Drusawin Vongpramate

Department of Information Technology

Faculty of Science, BRU

Hint



Main Topic

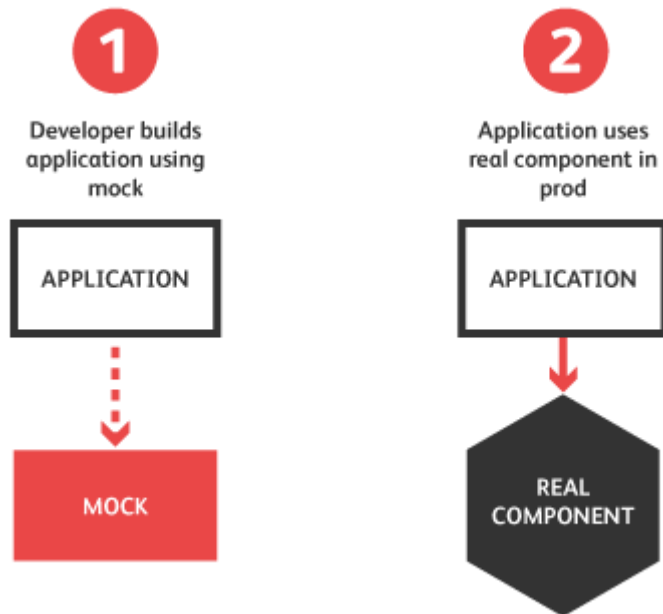


Sub Topic

Mock API

The easiest way to mock REST APIs

Quickly setup endpoints, generate custom data, and perform operations on it using RESTful interface



API endpoint

https://66b9a69

New resource

user

10

EDIT RESOURCE - USER

Schema

Define Resource schema, it will be used to generate mock data.

id	Object ID	
createdAt	Faker.js	date.recent
name	Faker.js	name.fullName
avatar	Faker.js	image.avatar
password	Faker.js	random.word
experience	Faker.js	random.numeric
url	Faker.js	internet.url
email	Faker.js	internet.exampleE

+

Object template

Requires subscrip

To define more complex structure for your data, use JSON template. You can use Faker.js function using `$function_name`.

```
{
  "username": "$internet.userName",
  "knownIps": ["$internet.ip", "$internet.ipv6"],
  "profile": {
    "firstName": "$name.firstName",
    "lastName": "$name.lastName",
    "staticData": [100, 200, 300]
  }
}
```

```
[
  {
    "createdAt": "2024-08-11T20:54:58.462Z",
    "name": "Albert Conn",
    "avatar": "https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhQHwNuwDibDjuhF96VLCBmG90QKSg97Dh4dAU6U",
    "password": "Squares",
    "experience": "5",
    "url": "https://oddball-sausage.info",
    "email": "Remington.Upton@example.com",
    "id": "1"
  },
  {
    "createdAt": "2024-08-11T08:05:27.647Z",
    "name": "Mr. Jan Gerlach",
    "avatar": "https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhQHwNuwDibDjuhF96VLCBmG90QKSg97Dh4dAU6U",
    "password": "male",
    "experience": "8",
    "url": "http://tender-plan.org",
    "email": "Douglas.Reichert@example.net",
    "id": "2"
  },
  {
    "createdAt": "2024-08-11T18:36:17.595Z",
    "name": "Yolanda Sawayn",
    "avatar": "https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhQHwNuwDibDjuhF96VLCBmG90QKSg97Dh4dAU6U",
    "password": "Dunwoody",
    "experience": "6",
    "url": "http://easy-going-plaintiff.net",
    "email": "Domenica73@example.org",
    "id": "3"
  },
  {
    "createdAt": "2024-08-12T02:05:48.195Z",
    "name": "Ray Daugherty",
    "avatar": "https://cloudflare-ipfs.com/ipfs/Qmd3W5DuhQHwNuwDibDjuhF96VLCBmG90QKSg97Dh4dAU6U",
    "password": "drive",
    "experience": "3",
    "url": "http://far-off-shawl.com",
    "email": "Ladarius_Douglas@example.net",
    "id": "4"
  }
]
```

Test on Postman

The screenshot displays the Postman interface for a REST client. At the top, the URL `https://66b9a695fa763ff550f8fa56.mockapi.io/api/v1/user` is entered. The request method is set to **POST**. The request body is formatted as **JSON** and contains the following content:

```
1 {
2   ... "name": "dadwin vong"
3 }
```

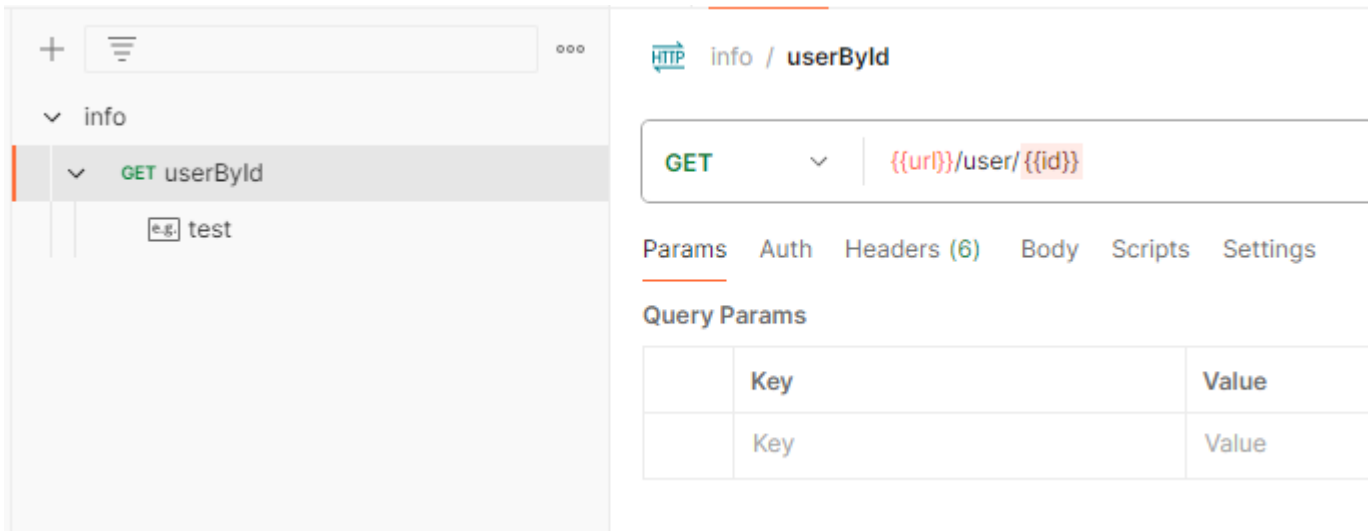
A red arrow points to the value `"dadwin vong"` in the request body. Below the request editor, the response is shown in the **Body** tab, formatted as **JSON**. The response content is:

```
1 {
2   "createdAt": "2024-08-11T11:59:39.368Z",
3   "name": "dadwin vong",
4   "avatar": "https://cloudflare-ipfs.com/ipfs/
5     Qmd3W5DuhgHirLHGVixi6V76LhCkZUZ6pnFt5AJBiyvHye/avatar/951.jpg",
6   "password": "Dong",
7   "experience": "5",
8   "url": "https://lined-painter.com",
9   "email": "Joany46@example.com",
10  "id": "13"
}
```

A red arrow points to the value `"dadwin vong"` in the response body. The interface also shows a status bar at the bottom with details: `201 Created`, `276 ms`, `1.12 KB`, and a `Save as example` button.

Mock API on Postman

Create “Info” Collection then add GET request
“userById” is `{{url}}/user/{{id}}`



The screenshot displays the Postman interface for configuring a GET request. On the left, a collection named 'info' is expanded to show a request named 'GET userById'. The main area shows the request configuration for 'info / userById' with the method 'GET' and the URL template `{{url}}/user/{{id}}`. Below the URL, there are tabs for 'Params', 'Auth', 'Headers (6)', 'Body', 'Scripts', and 'Settings'. The 'Query Params' section is currently active, showing a table with two columns: 'Key' and 'Value'.

Key	Value
Key	Value

Mock API on Postman

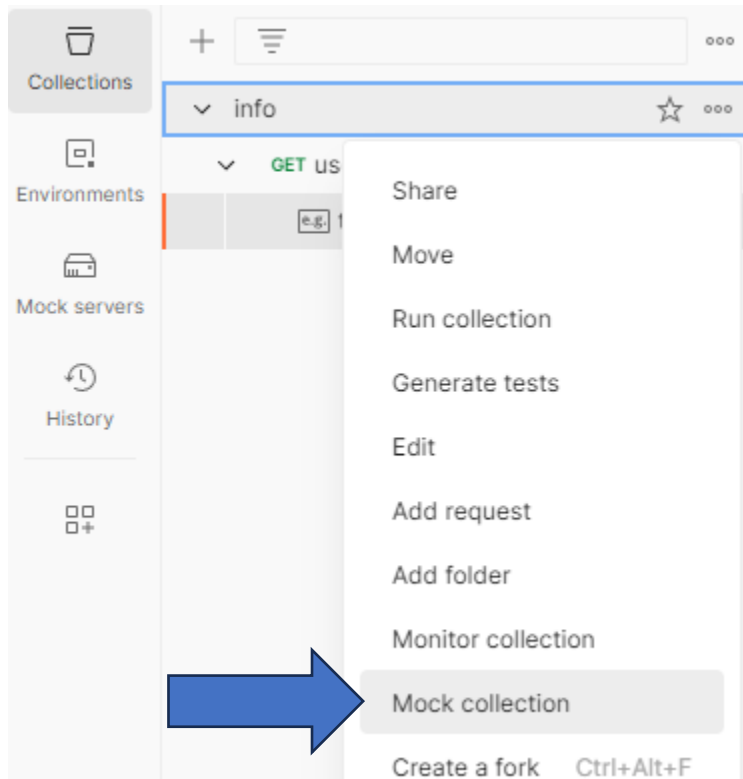
Create example (eg.) “test” and change url to `{{url}}/user/1` then add response body as picture

The screenshot shows the Postman interface for a mock API. The URL is `{{url}}/user/1` and the method is `GET`. The response body is a JSON object: `{ "user": "win", "age": 12 }`. The response status is `200 OK`. The response is displayed in the `JSON` view. Blue arrows point to the `Content-Type: application/json` dropdown, the JSON response body, and the `JSON` view toggle.

REST Verb	Action	Success	Failure
GET	Fetches a record or set of resources from the server	200	404
OPTIONS	Fetches all available REST operations	200	-
POST	Creates a new set of resources or a resource	201	404, 409
PUT	Updates or replaces the given record	200, 204	404
PATCH	Modifies the given record	200, 204	404
DELETE	Deletes the given resource	200	404

Mock API on Postman

Create Mock server by Mock collection menu



Create a mock server

Select collection to mock 2. Configuration

Mock Server Name

Collection

Environment

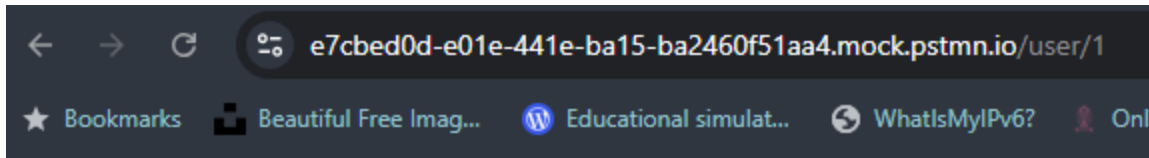
An environment is a group of variables useful for storing and reusing
[Learn more](#)

Save the mock server URL as a new environment variable
This will create a new environment containing URL.

Simulate a fixed network delay

Mock API on Postman

Test on Postman or Browser



```
{ "user": "win", "age": 12 }
```


Axios

Axios is a simple **promise** based HTTP client for the browser and node.js. Axios provides a simple to use library in a small package with a very extensible interface.

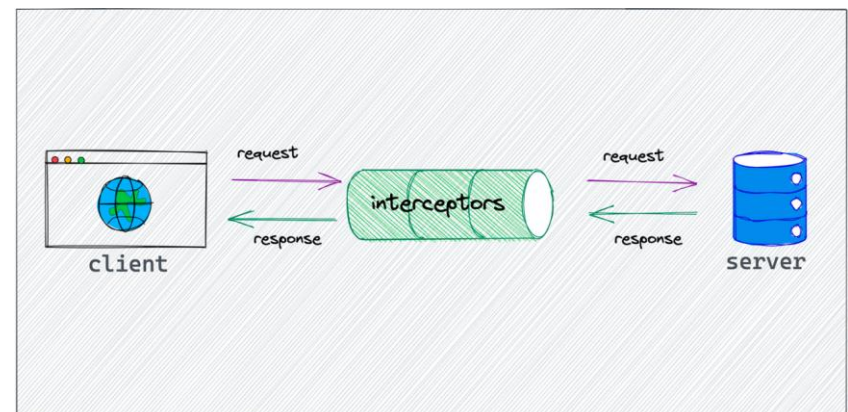
Install Package

```
> npm install axios
```

Coding

```
import axios from 'axios';
```

A X I O S



Create file for call service

/lib/userService.ts

```
1 import axios from 'axios';
2
3 const API_URL = 'https://66b9a695fa763ff550f8fa56.mockapi.io/api/v1/user';
4
5 export interface User {
6   id: string;
7   name: string;
8   email: string;
9   avatar?: string;
10  experience?: string;
11  password?: string;
12  createdAt?: string;
13 }
14
15 export const getUsers = async (): Promise<User[]> => {
16   const response = await axios.get<User[]>(API_URL);
17   return response.data;
18 };
19
20 export const getUserById = async (id: string): Promise<User> => {
21   const response = await axios.get<User>(`${API_URL}/${id}`);
22   return response.data;
23 };
```

Create file for call service

/lib/userService.ts

```
25 export const createUser = async (userData: Omit<User, 'id'>): Promise<User> => {
26     const response = await axios.post<User>(API_URL, userData);
27     return response.data;
28 };
29
30 export const updateUser = async (id: string, userData: Omit<User, 'id'>): Promise<User> => {
31     const response = await axios.put<User>(`${API_URL}/${id}`, userData);
32     return response.data;
33 };
34
35 export const deleteUser = async (id: string): Promise<void> => {
36     await axios.delete(`${API_URL}/${id}`);
37 };
38
```

Create route /user

/app/user/layout.tsx

```
1 import { ReactNode } from 'react';
2
3 export default function Layout({ children }: { children: ReactNode }) {
4   return (
5     <div>
6       <header>
7         <h1>My CRUD App</h1>
8       </header>
9       <main>{children}</main>
10      <footer>
11        <p>© 2024 My CRUD App</p>
12      </footer>
13    </div>
14  );
15 }
16
```

Create route /user

/app/user/
page.tsx

```
final > app > user > page.tsx > ...
1  'use client';
2
3  import Link from 'next/link';
4  import { useEffect, useState } from 'react';
5  import { getUsers, deleteUser, User } from '../lib/userService';
6
7  export default function UsersPage() {
8    const [users, setUsers] = useState<User[]>([]);
9
10   useEffect(() => {
11     const fetchUsers = async () => {
12       const data = await getUsers();
13       setUsers(data);
14     };
15     fetchUsers();
16   }, []);
17
18   const handleDelete = async (id: string) => {
19     await deleteUser(id);
20     setUsers(users.filter(user => user.id !== id));
21   };
22
23   return (
24     <div>
25       <h1>Users</h1>
26       <Link href="/user/create">Create New User</Link>
27       <ul>
28         {users.map(user => (
29           <li key={user.id}>
30             {user.name} - {user.email}
31             <Link href={`/user/edit/${user.id}`}>Edit</Link>
32             <button onClick={() => handleDelete(user.id)}>Delete</button>
33           </li>
34         ))}
35       </ul>
36     </div>
37   );
38 }
```

Create route /user

/app/user/create/page.tsx

```
final > app > user > create > page.tsx > CreateUser > handleSubmit > userData
1  'use client';
2
3  import { useState } from 'react';
4  import { useRouter } from 'next/navigation';
5  import { createUser, User } from '../../lib/userService';
6
7  export default function CreateUser() {
8    const [name, setName] = useState<string>('');
9    const [email, setEmail] = useState<string>('');
10   const [avatar, setAvatar] = useState<string>('');
11   const [experience, setExperience] = useState<string>('');
12   const router = useRouter();
13
14   const handleSubmit = async (e: React.FormEvent) => {
15     e.preventDefault();
16     const userData: Omit<User, 'id'> = {
17       name,
18       email,
19       avatar,
20       experience,
21       password: 'defaultpassword',
22     };
23     await createUser(userData);
24     router.push('/user');
25   };
26
```

Create route /user

/app/user
/create/
page.tsx

```
27   return (  
28     <div>  
29       <h1>Create User</h1>  
30       <form onSubmit={handleSubmit}>  
31         <input  
32           type="text"  
33           placeholder="Name"  
34           value={name}  
35           onChange={(e) => setName(e.target.value)}  
36           required  
37         />  
38         <input  
39           type="email"  
40           placeholder="Email"  
41           value={email}  
42           onChange={(e) => setEmail(e.target.value)}  
43           required  
44         />  
45         <input  
46           type="text"  
47           placeholder="Avatar URL"  
48           value={avatar}  
49           onChange={(e) => setAvatar(e.target.value)}  
50         />  
51         <input  
52           type="text"  
53           placeholder="Experience"  
54           value={experience}  
55           onChange={(e) => setExperience(e.target.value)}  
56         />  
57         <button type="submit">Create</button>  
58       </form>  
59     </div>  
60   );  
61 }
```

Create route /user

/app/user/edit/[id]/page.tsx

```
1  'use client';
2
3  import { useState, useEffect } from 'react';
4  import { useRouter, useParams } from 'next/navigation';
5  import { getUserById, updateUser, User } from '../../lib/userService';
6
7  export default function EditUser() {
8      const [name, setName] = useState<string>('');
9      const [email, setEmail] = useState<string>('');
10     const [avatar, setAvatar] = useState<string>('');
11     const [experience, setExperience] = useState<string>('');
12     const router = useRouter();
13     const { id } = useParams() as { id: string };
14
15     useEffect(() => {
16         const fetchUser = async () => {
17             const data = await getUserById(id);
18             setName(data.name);
19             setEmail(data.email);
20             setAvatar(data.avatar || '');
21             setExperience(data.experience || '');
22         };
23         fetchUser();
24     }, [id]);
25 }
```


Create route /user

/app/user/edit/[id]/page.tsx

```
26     const handleSubmit = async (e: React.FormEvent) => {
27         e.preventDefault();
28         const userData: Omit<User, 'id'> = {
29             name,
30             email,
31             avatar,
32             experience,
33         };
34         await updateUser(id, userData);
35         router.push('/user');
36     };
```

Create route /user

/app/user/edit/[id]/page.tsx

```
26     const handleSubmit = async (e: React.FormEvent) => {
27         e.preventDefault();
28         const userData: Omit<User, 'id'> = {
29             name,
30             email,
31             avatar,
32             experience,
33         };
34         await updateUser(id, userData);
35         router.push('/user');
36     };
```

Create route /user

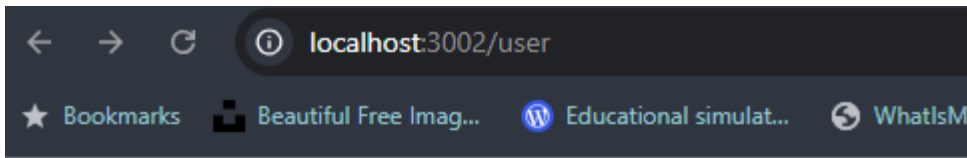
/app/user/edit/[id]/page.tsx

```
26     const handleSubmit = async (e: React.FormEvent) => {
27         e.preventDefault();
28         const userData: Omit<User, 'id'> = {
29             name,
30             email,
31             avatar,
32             experience,
33         };
34         await updateUser(id, userData);
35         router.push('/user');
36     };
```

Create route /user

/app/user
/edit/[id]/
page.tsx

```
38     return (  
39         <div>  
40             <h1>Edit User</h1>  
41             <form onSubmit={handleSubmit}>  
42                 <input  
43                     type="text"  
44                     placeholder="Name"  
45                     value={name}  
46                     onChange={(e) => setName(e.target.value)}  
47                     required  
48                 />  
49                 <input  
50                     type="email"  
51                     placeholder="Email"  
52                     value={email}  
53                     onChange={(e) => setEmail(e.target.value)}  
54                     required  
55                 />  
56                 <input  
57                     type="text"  
58                     placeholder="Avatar URL"  
59                     value={avatar}  
60                     onChange={(e) => setAvatar(e.target.value)}  
61                 />  
62                 <input  
63                     type="text"  
64                     placeholder="Experience"  
65                     value={experience}  
66                     onChange={(e) => setExperience(e.target.value)}  
67                 />  
68                 <button type="submit">Update</button>  
69             </form>  
70         </div>  
71     );  
72 }  
73
```



My CRUD App

Users

[Create New User](#)

- Albert Conn - Remington.Upton@example.com [Edit](#) [Delete](#)
- Ray Daugherty - Ladarius_Douglas@example.net [Edit](#) [Delete](#)
- Alfredo Parker - Sanford27@example.org [Edit](#) [Delete](#)
- Mrs. Alma Moore - Giuseppe_Bogan88@example.net [Edit](#) [Delete](#)
- Ethel Zulauf - Beulah.Schultz10@example.net [Edit](#) [Delete](#)
- Pearl Beier - Hortense_Hayes70@example.net [Edit](#) [Delete](#)
- Julia Langworth - Khalil.Stanton@example.com [Edit](#) [Delete](#)
- Horace Kirlin - Summer.Hane96@example.net [Edit](#) [Delete](#)
- Mrs. Troy Mante IV - Vergie.Stroman42@example.com [Edit](#) [Delete](#)
- Hector Nienow - Tressa_Little80@example.org [Edit](#) [Delete](#)

© 2024 My CRUD App

My CRUD App

Edit User

Albert Conn	Remington.Upton@example	https://cloudflare-ipfs.com/ipf	5	Update
-------------	-------------------------	---------------------------------	---	--------

© 2024 My CRUD App



Create CRUD



Let's Try

Guideline

1. Which API from the service?
2. Which UI for control?

Q & A