CRUD on API

Asst.Prof.Drusawin Vongpramate
Department of Information Technology
Faculty of Science, BRU

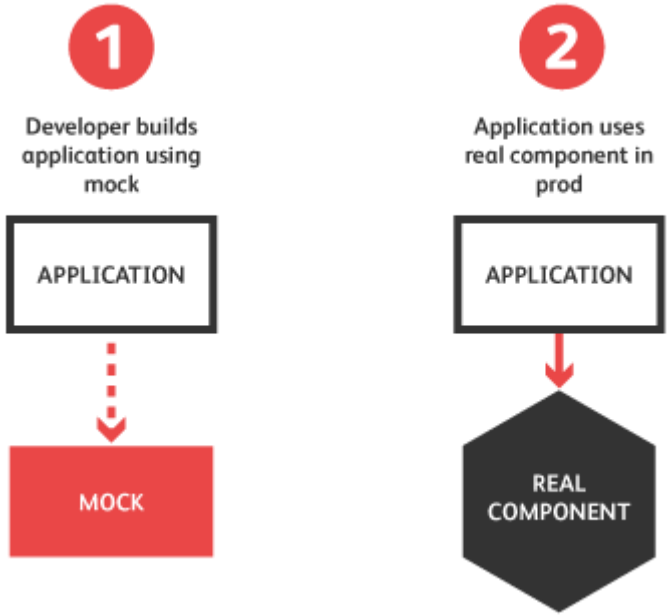
Hint

Main Topic Sub Topic

Mock API

The easiest way to mock REST APIs

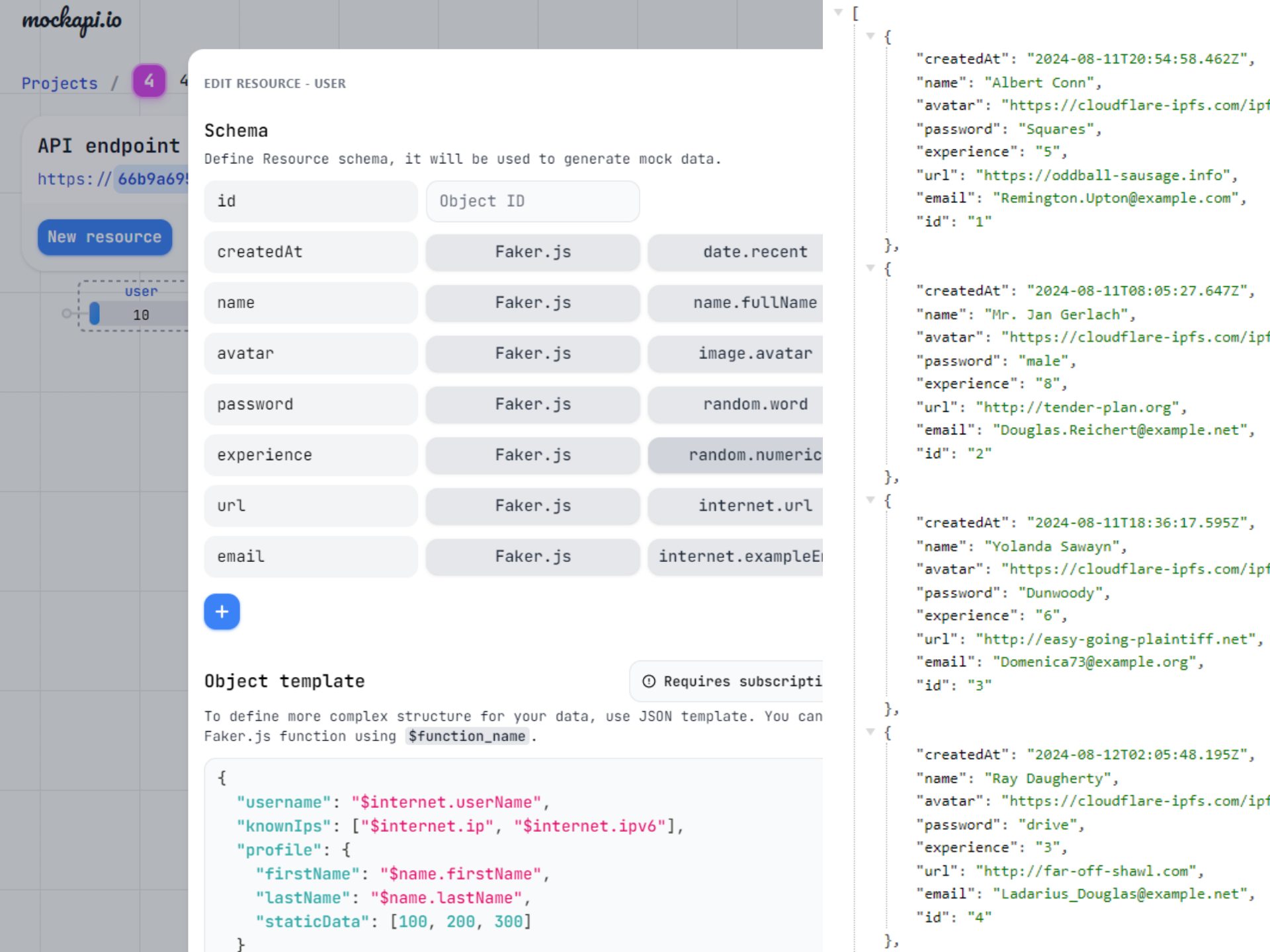
Quickly setup endpoints, generate custom data, and perform operations on it using RESTful interface



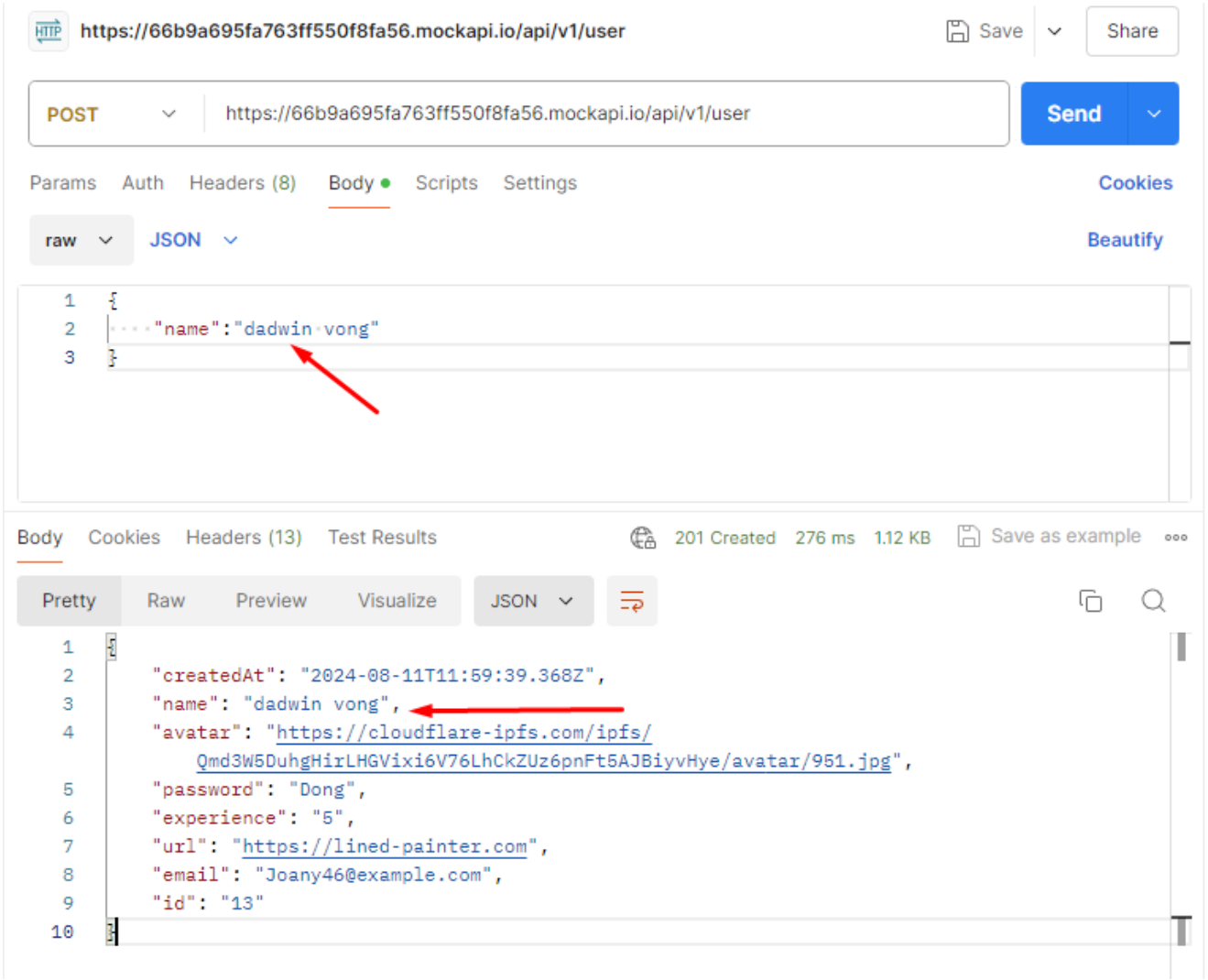
mockapi.io

MOCKANAPI

mockoon

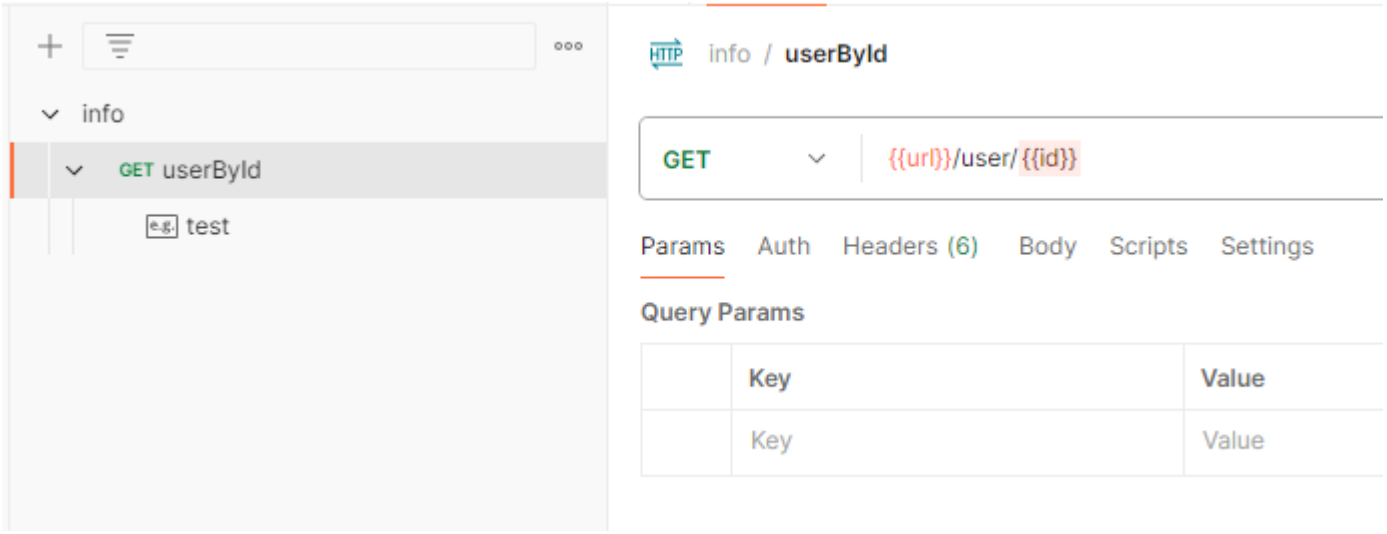

mockapi.io
Projects / 4 4
API endpoint
https://66b9a69
New resource
user
10
EDIT RESOURCE - USER
Schema
Define Resource schema, it will be used to generate mock data.
id
Object ID
createdAt
Faker.js
date.recent
name
Faker.js
name.fullName
avatar
Faker.js
image.avatar
password
Faker.js
random.word
experience
Faker.js
random.numeric
url
Faker.js
internet.url
email
Faker.js
internet.exampleE
Object template
Requires subscripti
To define more complex structure for your data, use JSON template. You can
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/ipf
"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/ipf
"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/ipf
"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/ipf
"password": "drive",
"experience": "3",
"url": "http://far-off-shawl.com",
"email": "Ladarius_Douglas@example.net",
"id": "4"
},

Test on Postman



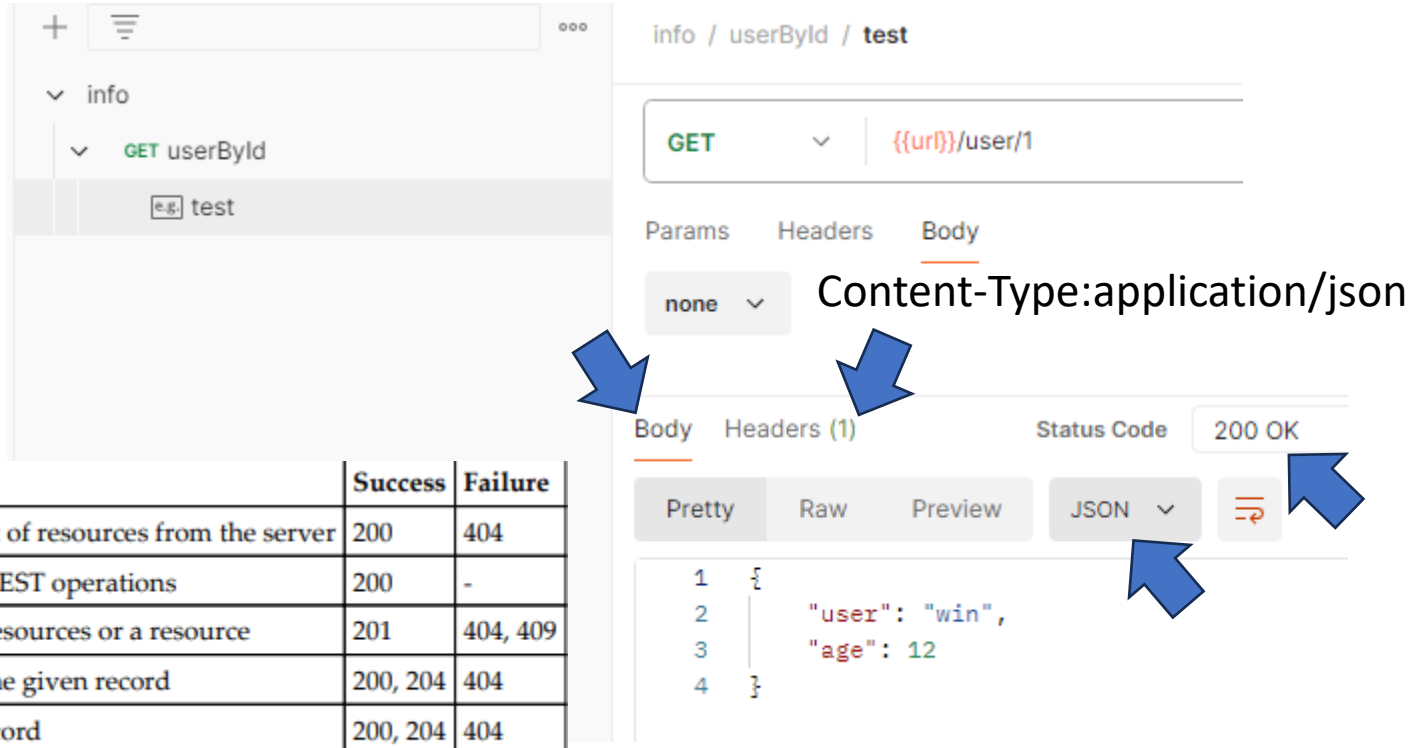
Mock API on Postman

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



Mock API on Postman

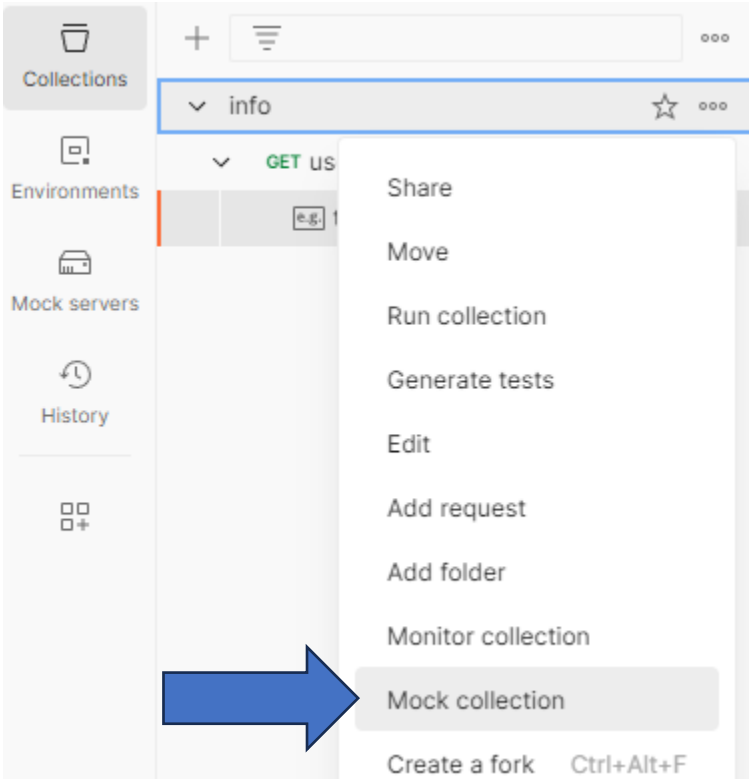
Create example (eg.) “test” and change url to {{url}}/user/1 then add response body as picture

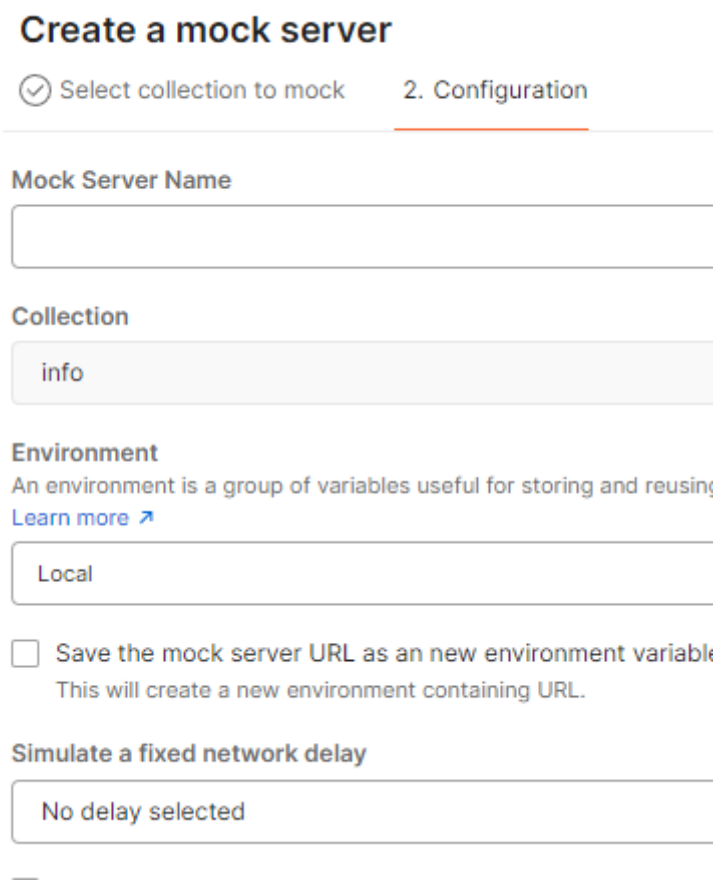


REST Verb	Action	Success	Failure
GET	Fetches a record or set of resources from the server	200	404
OPTIONS	Fetches all available REST operations	200	-
POST	Creates a new set of resources or a resource	201	404, 409
PUT	Updates or replaces the given record	200, 204	404
PATCH	Modifies the given record	200,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

Mock API on Postman

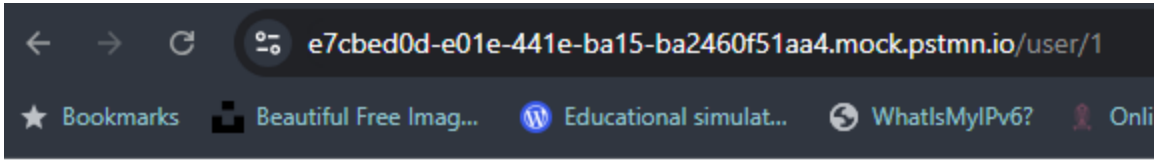
Create Mock server by Mock collection menu





Mock API on Postman

Test on Postman or Browser



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

Axios

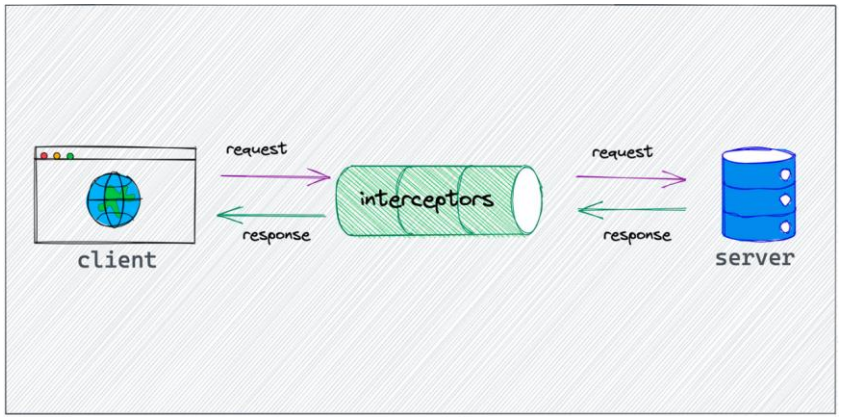
Axios is a simple promise based HTTP client for the browser and node.js. Axios provides a simple to use library in a small package with a very extensible interface.

Install Package

> npm install axios

Coding

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



Create file for call service

/lib/userService.ts

```ts
import axios from 'axios';

const API_URL = 'https://66b9a695fa763ff550f8fa56.mockapi.io/api/v1/user';

export interface User {
    id: string;
    name: string;
    email: string;
    avatar?: string;
    experience?: string;
    password?: string;
    createdAt?: string;
}

export const getUsers = async (): Promise<User[]> => {
    const response = await axios.get<User[]>(API_URL);
    return response.data;
};

export const getUserById = async (id: string): Promise<User> => {
    const response = await axios.get<User>(`${API_URL}/${id}`);
    return response.data;
};
```

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

```tsx
import { ReactNode } from 'react';

export default function Layout({ children }: { children: ReactNode }) {
    return (
        <div>
            <header>
                <h1>My CRUD App</h1>
            </header>
            <main>{children}</main>
            <footer>
                <p>© 2024 My CRUD App</p>
            </footer>
        </div>
    );
}

```

Create route /user

/app/user/page.tsx

final > app > user > page.tsx > ...

```tsx
'use client';

import Link from 'next/link';
import { useEffect, useState } from 'react';
import { getUsers, deleteUser, User } from '../../lib/userService';

export default function UsersPage() {
    const [users, setUsers] = useState<User[]>([]);

    useEffect(() => {
        const fetchUsers = async () => {
            const data = await getUsers();
            setUsers(data);
        };
        fetchUsers();
    }, []);

    const handleDelete = async (id: string) => {
        await deleteUser(id);
        setUsers(users.filter(user => user.id !== id));
    };

    return (
        <div>
            <h1>Users</h1>
            <Link href="/user/create">Create New User</Link>
            <ul>
                {users.map(user => (
                    <li key={user.id}>
                        {user.name} - {user.email}
                        <Link href={`/user/edit/${user.id}`}>Edit</Link>
                        <button onClick={() => handleDelete(user.id)}>Delete</button>
                    </li>
                ))}
            </ul>
        </div>
    );
}
```

Create route /user

/app/user/create/page.tsx

final > app > user > create > page.tsx > CreateUser > handleSubmit > userData

```tsx
'use client';

import { useState } from 'react';
import { useRouter } from 'next/navigation';
import { createUser, User } from '../../../lib/userService';

export default function CreateUser() {
    const [name, setName] = useState<string>('');
    const [email, setEmail] = useState<string>('');
    const [avatar, setAvatar] = useState<string>('');
    const [experience, setExperience] = useState<string>('');
    const router = useRouter();

    const handleSubmit = async (e: React.FormEvent) => {
        e.preventDefault();
        const userData: Omit<User, 'id'> = {
            name,
            email,
            avatar,
            experience,
            password: 'defaultpassword',
        };
        await createUser(userData);
        router.push('/user');
    };

```

Create route /user

/app/user
/create/
page.tsx

```tsx
    return (
        <div>
            <h1>Create User</h1>
            <form onSubmit={handleSubmit}>
                <input
                    type="text"
                    placeholder="Name"
                    value={name}
                    onChange={(e) => setName(e.target.value)}
                    required
                />
                <input
                    type="email"
                    placeholder="Email"
                    value={email}
                    onChange={(e) => setEmail(e.target.value)}
                    required
                />
                <input
                    type="text"
                    placeholder="Avatar URL"
                    value={avatar}
                    onChange={(e) => setAvatar(e.target.value)}
                />
                <input
                    type="text"
                    placeholder="Experience"
                    value={experience}
                    onChange={(e) => setExperience(e.target.value)}
                />
                <button type="submit">Create</button>
            </form>
        </div>
    );
}
```

Create route /user

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

```tsx
'use client';

import { useState, useEffect } from 'react';
import { useRouter, useParams } from 'next/navigation';
import { getUserById, updateUser, User } from '../../../../lib/userService';

export default function EditUser() {
    const [name, setName] = useState<string>('');
    const [email, setEmail] = useState<string>('');
    const [avatar, setAvatar] = useState<string>('');
    const [experience, setExperience] = useState<string>('');
    const router = useRouter();
    const { id } = useParams() as { id: string };

    useEffect(() => {
        const fetchUser = async () => {
            const data = await getUserById(id);
            setName(data.name);
            setEmail(data.email);
            setAvatar(data.avatar || '');
            setExperience(data.experience || '');
        };
        fetchUser();
    }, [id]);
```

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

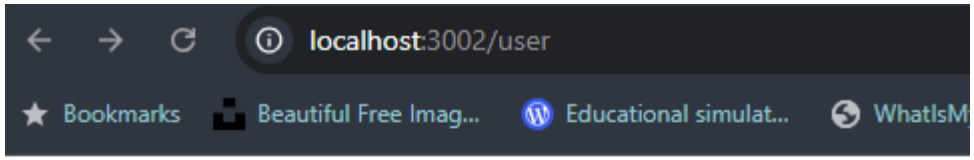
/app/user/edit/[id]/page.tsx

```
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.comEdit Delete
- Ray Daugherty - Ladarius_Douglas@example.netEdit Delete
- Alfredo Parker - Sanford27@example.orgEdit Delete
- Mrs. Alma Moore - Giuseppe_Bogan88@example.netEdit Delete
- Ethel Zulauf - Beaulah.Schultz10@example.netEdit Delete
- Pearl Beier - Hortense_Hayes70@example.netEdit Delete
- Julia Langworth - Khalil.Stanton@example.comEdit Delete
- Horace Kirlin - Summer.Hane96@example.netEdit Delete
- Mrs. Troy Mante IV - Vergie.Stroman42@example.comEdit Delete
- Hector Nienow - Tressa_Little80@example.orgEdit Delete

My CRUD App

Edit User

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

Create CRUD

Guideline

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

Q & A