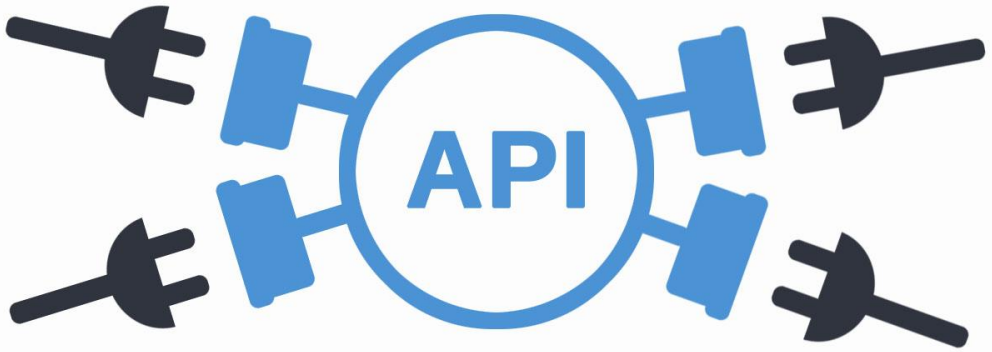
Theory 09
API Overview, Fetch API, Axios

WEEK 02

Nội dung chính

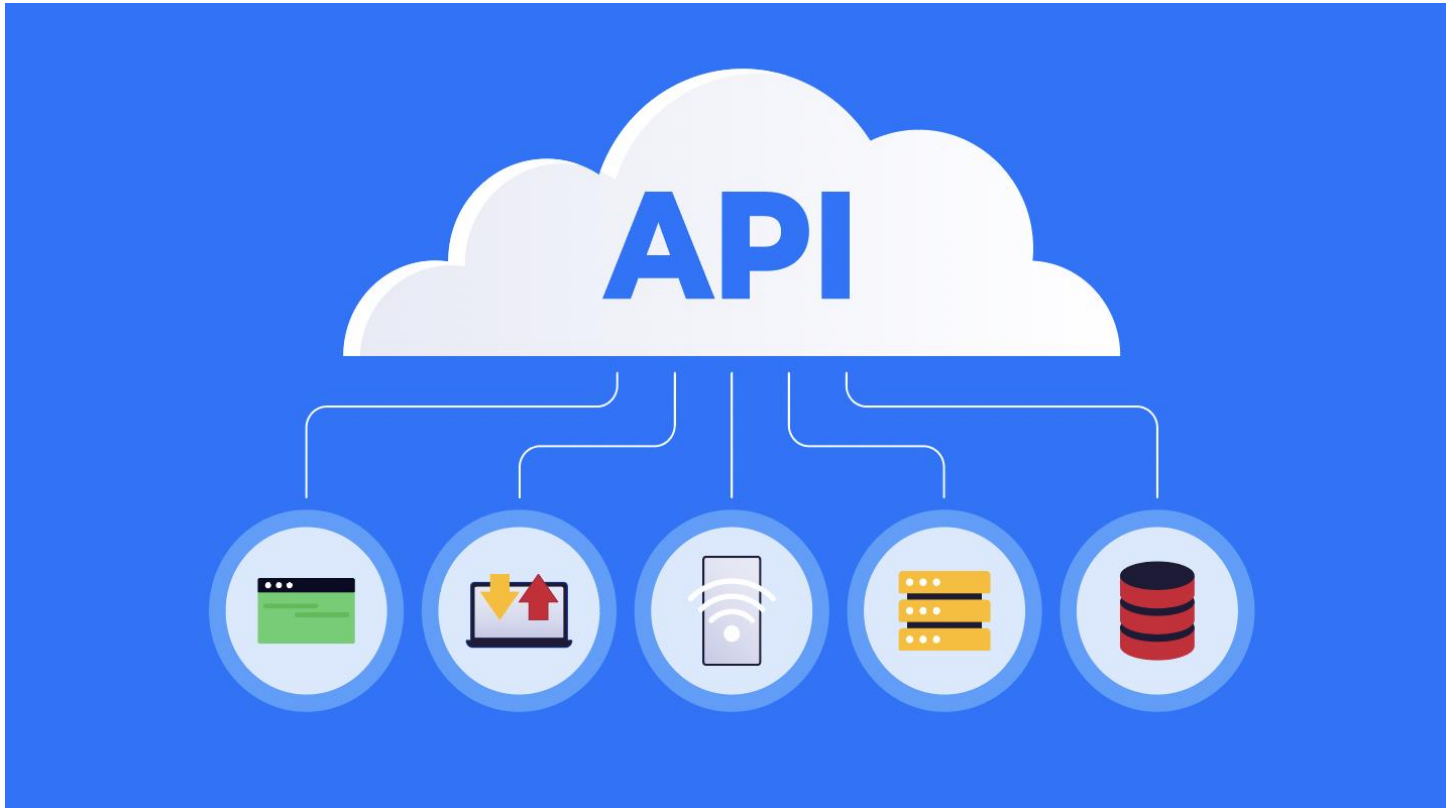
API, Web API, Restful API

❑ **API** viết tắt của **Application Programming Interface** (giao diện lập trình ứng dụng)
❑ Đây là phương tiện cho hai hoặc nhiều ứng dụng trao đổi, tương tác với nhau, tạo ra tương tác giữa người dùng với ứng dụng hiệu quả và tiện lợi hơn.
❑ Với **API**, các lập trình viên có thể tiếp cận, truy xuất dữ liệu từ máy chủ thể hiện chúng trên ứng dụng phần mềm hoặc website của mình một cách dễ dàng hơn



API, Web API, Restful API

❑ **API** bao gồm: Web API, API trên hệ điều hành, API của Library/Framework



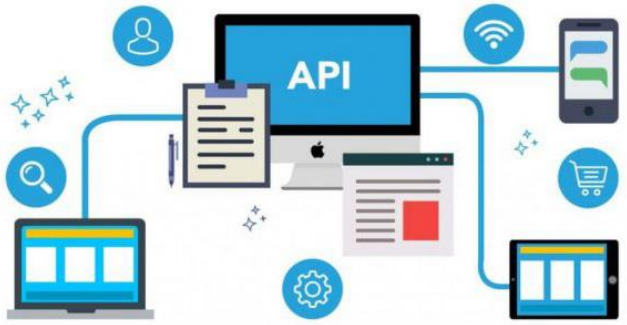
API, Web API, Restful API

❑ Những tính năng của **Web API** bao gồm:

❍ **Tự động hóa sản phẩm**: Đối với Web API, sẽ giúp người dùng có thể dễ dàng tự động quản lý được công việc.

❍ **Tích hợp linh động :** API cho phép lấy nội dung ở bất kỳ Website hay ứng dụng nào đó một cách dễ dàng, khiến trải nghiệm người dùng được tăng lên.

❍ **Cập nhật thông tin theo thời gian thực**: API giúp thay đổi và cập nhật những thông tin mới theo thời gian thực.



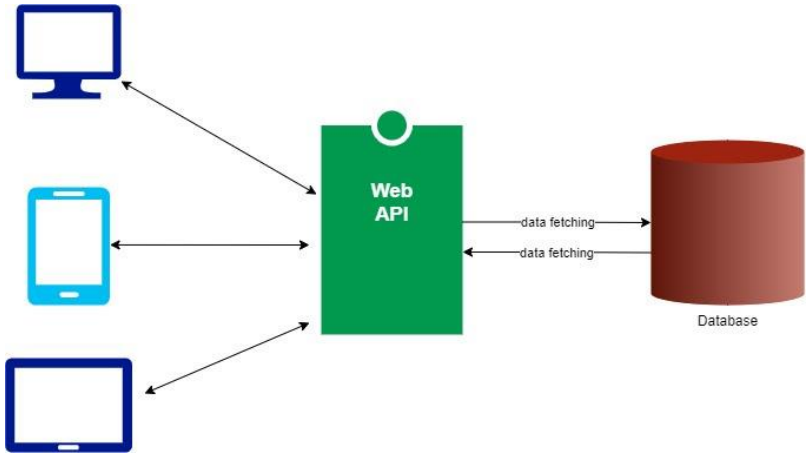
API, Web API, Restful API

❑ Ưu điểm của **Web API**:

❍ **Web API** được sử dụng khá rộng rãi ở trên các ứng dụng như: **Desktop**, **Mobile** và cả ứng dụng ở Website (**Web App**).

❍ Linh hoạt đối với các dạng dữ liệu trả về Client: **JSON**, **XML**,...

❍ Dễ dàng xây dựng được **HTTP service**: **URI**, **request/response headers**, **caching**, **versioning**, **content formats** và cả **host** trong ứng dụng.



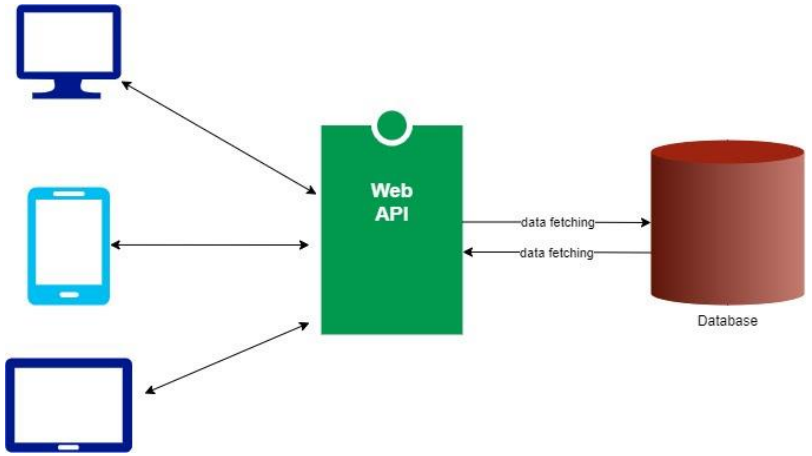
API, Web API, Restful API

❑ Ưu điểm của **Web API**:

❍ Với mã nguồn mở có thể giúp hỗ trợ những chức năng của **RESTful** một cách đầy đủ.

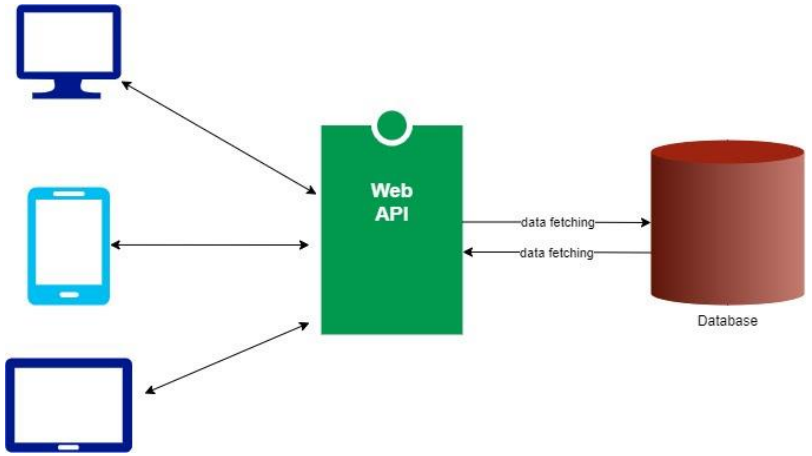
❍ Hỗ trợ về thành phần **MVC** như: **routing**, **controller**, **action result**, **filter**, **model binder**, **IoC container**, **dependency injection**, **unit test**

❍ Giao tiếp 2 chiều được xác nhận, các giao dịch có thể đảm bảo độ tin cậy cao.



API, Web API, Restful API

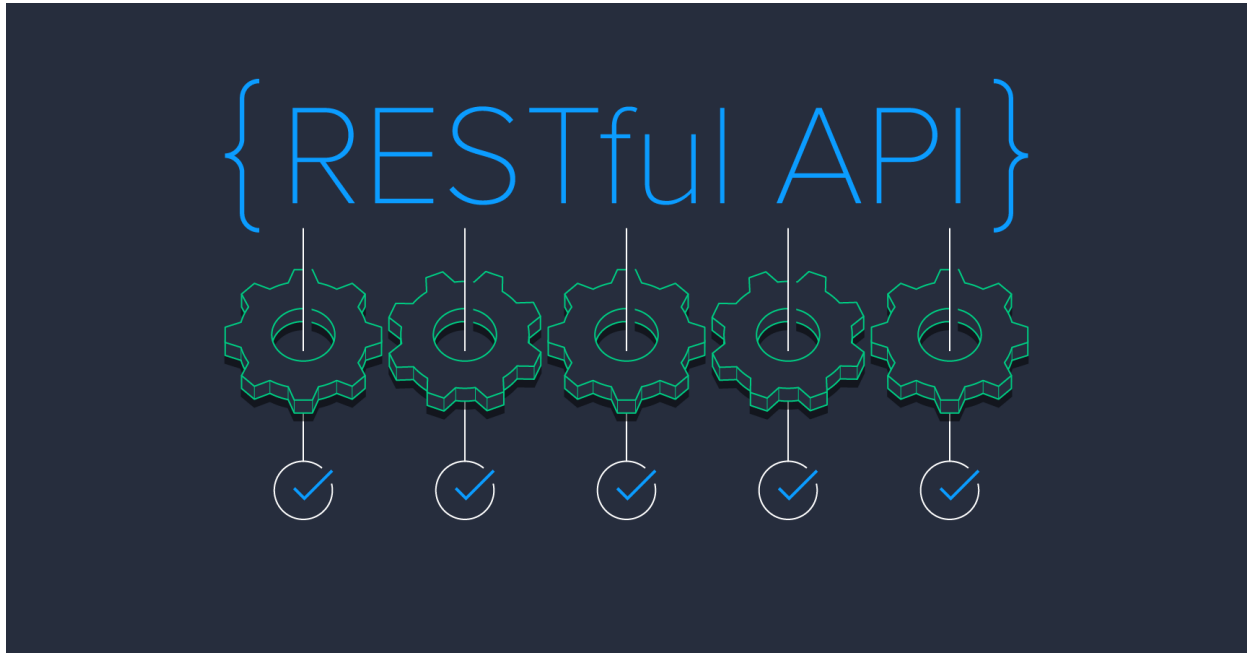
- Ưu điểm của **Web API**:
  - Web API chưa được gọi là **RESTful Service** bởi nó chỉ mới hỗ trợ **GET**, **POST**.
  - Nếu muốn sử dụng tốt nhất bạn cần có kiến thức và am hiểu thật sự về backend
  - Khá mất thời gian cho việc phát triển cũng như nâng cấp, vận hành.
  - Hệ thống có thể bị tấn công nếu như không giới hạn chức năng hay điều kiện



API, Web API, Restful API

❑ **RESTful API** là gì?

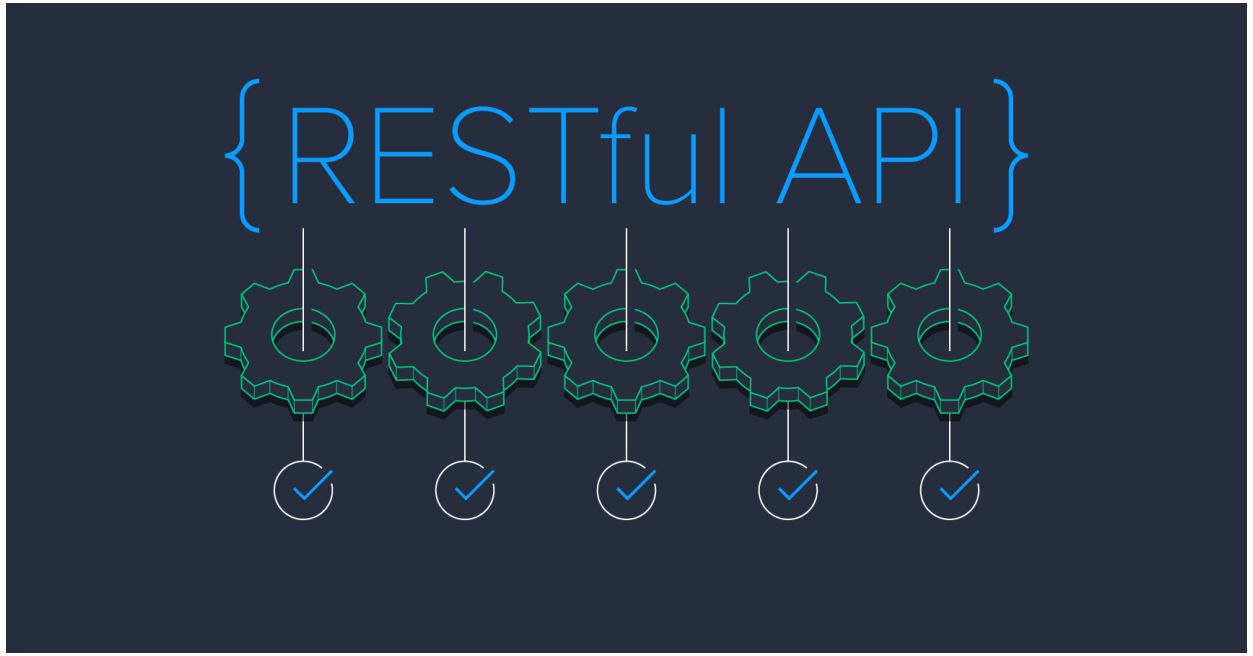
❍ **RESTful API** là một trong những kiểu thiết kế **API** được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile…) khác nhau giao tiếp với nhau



API, Web API, Restful API

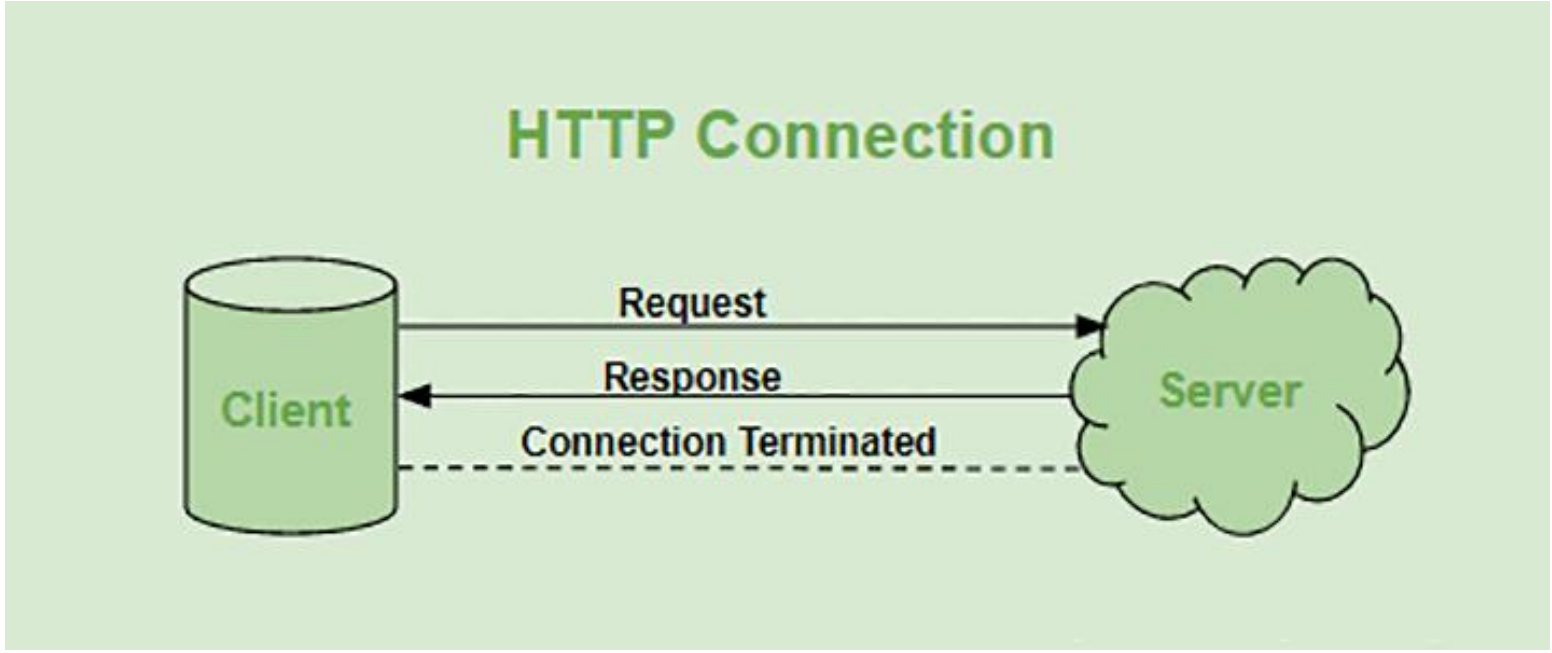
❑ **RESTful API** là gì?

❍ **RESTful API** có dữ liệu được trả về với nhiều định dạng khác nhau như: XML, HTML, JSON,...



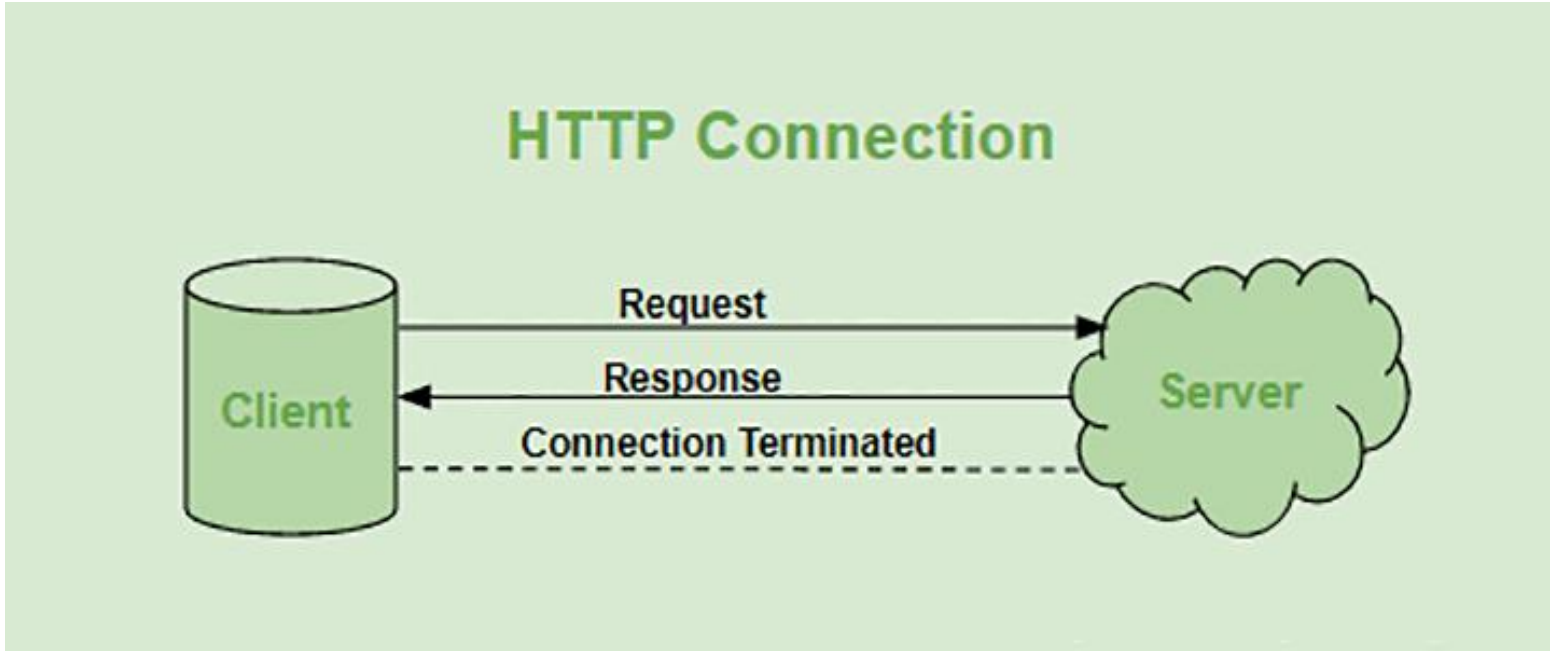
Làm việc với HTTP

- **HTTP** (Hyper Text Transfer Protocol) là một giao thức nằm ở tầng ứng dụng (Application layer) của tập giao thức **TCP/IP**, sử dụng để truyền nhận dữ liệu giữa các hệ thống phân tán thông qua internet.



Làm việc với HTTP

- **HTTP client** thiết lập một kết nối **TCP** đến **server**. **Client** và **server** sẽ truyền nhận dữ liệu với nhau thông qua kết nối này. Bao gồm: địa chỉ **IP**, loại giao thức giao vận (**TCP**), và **port** (mặc định là 80).

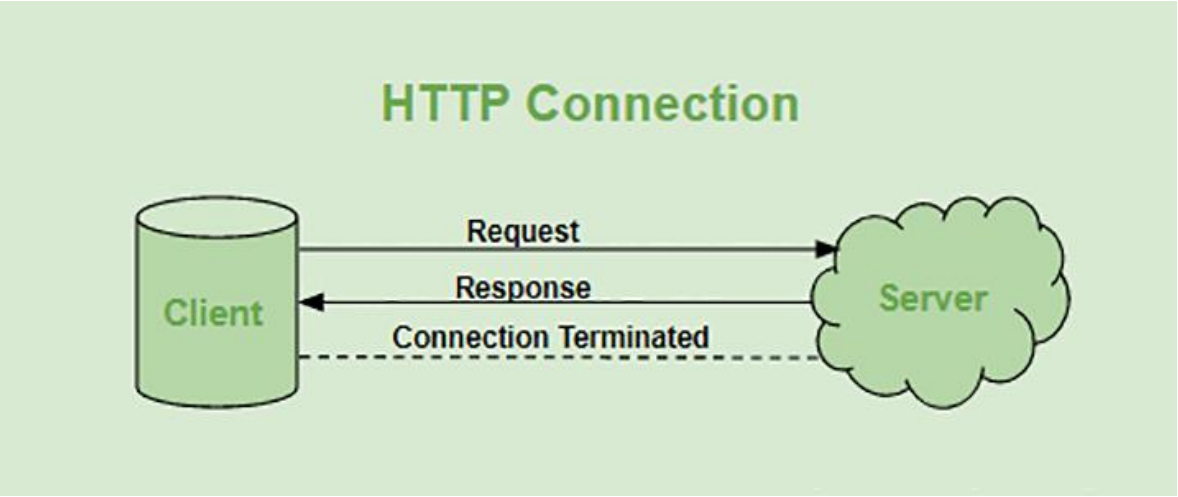


Làm việc với HTTP

❑ Server sẽ nhận và xử lý request từ client thông qua socket, sau đó đóng gói dữ liệu tương ứng và gửi một HTTP response về cho client). Dữ liệu trả về sẽ là một file HTML chứa các loại dữ liệu khác nhau như văn bản, hình ảnh,…

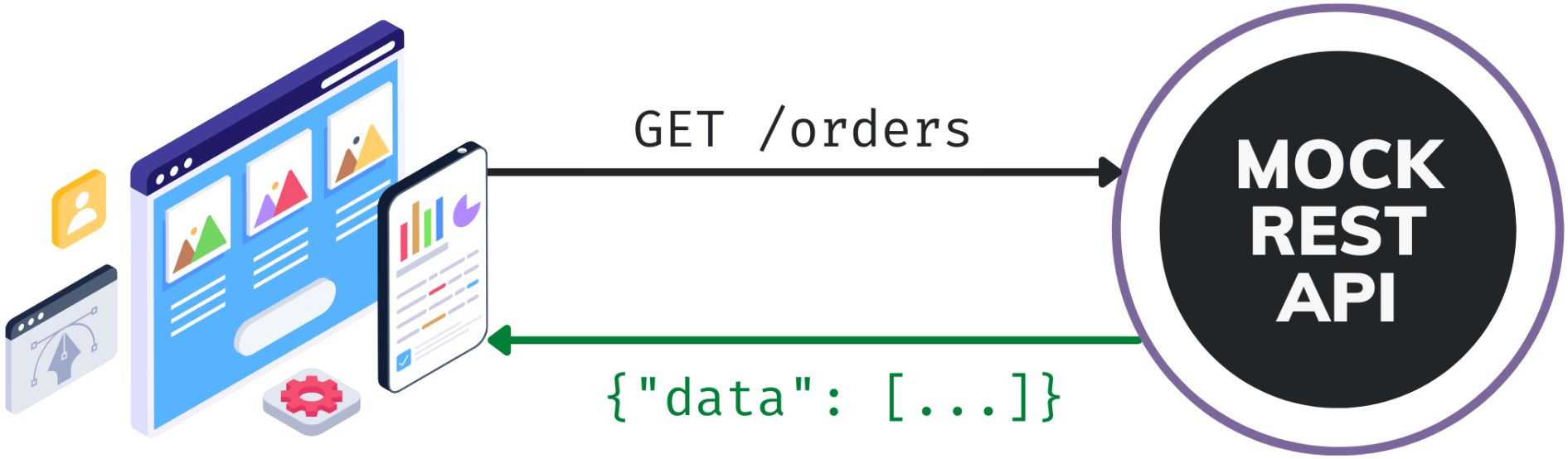
❑ Server đóng kết nối TCP.

❑ Client nhận được dữ liệu phản hồi từ server và đóng kết nối TCP



Tạo Backend API với Mock API

❑ Các nhà phát triển **Frontend** và các nhà phát triển **Backend** có thể làm việc song song, do đó phát triển phần mềm trở nên nhanh chóng hơn.
❑ Các nhà phát triển **Frontend** có thể bắt đầu với các API mô phỏng mà không cần chuyên sâu các kỹ năng của **Backend.**



Tạo Backend API với Mock API

❑ Sử dụng **Mock API** để tạo sẵn các đầu **API phía Backend** sử dụng cho **Frontend**.

Gọi Backend API với Fetch API

- Viết hàm sử dụng **fetch()** có sẵn để gọi **API** lấy danh sách các user từ **MockAPI**.
  - Tạo component **UserList** như sau:

```
import React, { useEffect, useState } from 'react';

const UserList = () => {

    return (
      <ul>
          {users.map(user => (
          <li key={user.id}>{user.name} - {user.email}</li>
          ))}
      </ul>
    );
};

export default UserList;
```

Gọi Backend API với Fetch API

- Viết hàm sử dụng **fetch()** có sẵn để gọi **API** lấy danh sách các user từ **MockAPI**.
  - Tạo 2 state **users, loading** trong component **UserList** như sau:

```
const [users, setUsers] = useState([]);
const [loading, setLoading] = useState(true);

useEffect(() => {
    // xử lý gọi API
}, []);

if (loading) return <p>Đang tải dữ liệu...</p>;
```

Gọi Backend API với Fetch API

- Viết hàm sử dụng **fetch()** có sẵn để gọi **API** lấy danh sách các user từ **MockAPI**.
  - Thêm code trong **useEffect()** để gọi API bằng hàm **fetch() + Promise** thuần:

```
useEffect(() => {
    fetch('https://jsonplaceholder.typicode.com/users')
        .then (response => {
            if (!response.ok) throw new Error('Lỗi khi gọi API');
            return response.json();
         }).then(data => {
            setUsers(data);
            setLoading(false);
         }).catch(error => {
            console.error('Lỗi: ', error);
            setLoading(false);
         });
}, []);
```

Gọi Backend API với Fetch API

- Viết hàm sử dụng **fetch()** có sẵn để gọi **API** lấy danh sách các user từ **MockAPI**.
  - Thêm code trong **useEffect()** để gọi API bằng hàm **fetch() + async/await**:

```
useEffect(() => {
    const fetchUsers = async () => {
        try {
            const response = await fetch('https://jsonplaceholder.typicode.com/users');
            if (!response.ok) throw new Error('Lỗi khi gọi API');
            const data = await response.json();
            setUsers(data);
        } catch (error) {
            console.error(error);
        } finally {
            setLoading(false);
        }
    };

    fetchUsers();
}, []);
```

Gọi Backend API với Axios

❑ Viết hàm sử dụng **axios + Promise** gọi **API** lấy danh sách các user từ **MockAPI**.

```
getUsers = () => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      axios
        .get("http://localhost:3001/api/users")
        .then(res => {
          resolve(res);
        }).catch(err => {
          reject(err);
        });
    }, 3000);
  });
};
```

Gọi Backend API với Axios

❑ Viết hàm sử dụng **axios + Promise** gọi **API** lấy danh sách các user từ **MockAPI**.

```
13    componentDidMount() {
14      this.setState({ loading: true });
15      this.getUsers()
16          .then(res => {
17            this.setState({ users: res.data });
18          }).catch(err => {
19            throw err;
20          }).finally(() => {
21            this.setState({ loading: false });
22          });
23    }
```

Gọi Backend API với Axios

❑ Viết hàm sử dụng **axios** gọi **axios + Promise** lấy danh sách các user từ **MockAPI**.

```
39    render() {
40      const { loading, users } = this.state;
41      if (loading) return <p>loading...</p>;
42      return (
43        <div>
44          <h1>Users</h1>
45          <ul>
46            {users.map(user => (
47              <li key={user.id}> {user.name} </li>
48            ))}
49          </ul>
50        </div>
51      );
52    }
```

Gọi Backend API với Axios

❑ Viết hàm sử dụng **axios + async/await** gọi **API** lấy danh sách sách từ **MockAPI**.

```
4   function Books() {
5       const BOOK_MANAGEMENT_API = "https://[redacted].mockapi.io/api";
6       const [books, setBooks] = useState([]);
7
8       useEffect(() => {
9           axios
10              .get(`${BOOK_MANAGEMENT_API}/books`)
11              .then(res => {
12                  setBooks(res.data);
13              })
14              .catch(err => {
15                  throw err;
16              });
17      }, [books]);
18
19      function handleCreate() {
20          window.location.href = "/book/add";
21      }
22
23  >   return (···
53      );
54  }
55
56  export default Books;
```

Gọi Backend API với Axios

❑ Viết hàm sử dụng **axios + async/await** gọi **API** lấy danh sách sách từ **MockAPI**.

```
const BOOK_MANAGEMENT_API = "https://[redacted].mockapi.io/api";
const { bookId } = useParams();
const [book, setBook] = useState([]);

useEffect(() => {
  if (bookId) {
    axios
      .get(`${BOOK_MANAGEMENT_API}/books/${bookId}`)
      .then(res => {
        setBook(res.data);
      })
      .catch(err => {
        throw err;
      });
  }
}, [bookId]);
```

Gọi Backend API với Axios

❑ Viết hàm sử dụng **axios + async/await** gọi **API** lấy danh sách sách từ **MockAPI**.

```
31  function handleSubmit() {
32    axios
33      .post(`${BOOK_MANAGEMENT_API}/books`, book)
34      .then(res => {
35        alert(
36          `${isCreate ? "Create" : "Edit"} book ${JSON.stringify(
37            res.data
38          )} successfully!!!`
39        );
40        window.location.href = "/";
41      })
42      .catch(err => {
43        throw err;
44      });
45  }
```

Gọi Backend API với Axios

❑ Viết hàm sử dụng **axios + async/await** gọi **API** lấy danh sách sách từ **MockAPI**.

```
31  function handleSubmit() {
32    axios
33      .put(`${BOOK_MANAGEMENT_API}/books/${bookId}`, book)
34      .then(res => {
35        alert(
36          `${isCreate ? "Create" : "Edit"} book ${JSON.stringify(
37            res.data
38          )} successfully!!!`
39        );
40        window.location.href = "/";
41      })
42      .catch(err => {
43        throw err;
44      });
45  }
```

Gọi Backend API với Axios

❑ Viết hàm sử dụng **axios + async/await** gọi **API** lấy danh sách sách từ **MockAPI**.

```
27  function removeBook() {
28    if (bookId) {
29      axios
30      .delete(`${BOOK_MANAGEMENT_API}/books/${bookId}`)
31      .then(res => {
32        alert(
33          `Remove book ${JSON.stringify(
34            res.data
35          )} successfully!!!`
36        );
37        window.location.href = "/";
38      })
39      .catch(err => {
40        throw err;
41      });
42    }
43  }
```

Tóm tắt bài học

- ❑ API, Web API, Restful API
- ❑ Quá trình làm việc với HTTP
- ❑ Tạo được API với Mock API
- ❑ Gọi Backend API với Fetch API
- ❑ Gọi Backend API với Axios