VÕ TIẾN
Thảo luận kiến thức CNTT trường BK về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

Design Framework CS K23

CNPM & HTTT CS23

Thiết kế mô hình Server - Client giao thức HTTP

Thảo luận kiến thức CNTT trường BK
về KHMT(CScience), KTMT(CEngineering)
https://www.facebook.com/groups/khmt.ktmt.cse.bku

TP. HỒ CHÍ MINH, THÁNG 6/2025

Mục lục

1 HTTP **2**

2 Header và cách trình duyệt quản lí cookie **3**

3 Postman **5**

4 Trình Duyệt **6**

5 Dùng Swagger **7**

6 Thiết Kế hệ thông **8**

7 Yêu Cầu **9**

1 HTTP

HTTP là một **giao thức tầng ứng dụng** trong mô hình OSI, sử dụng mô hình **client-server**, nơi client (ví dụ trình duyệt web) gửi yêu cầu (request) đến server và server phản hồi (response) lại.

Cấu trúc một yêu cầu HTTP (HTTP Request)

1. Dòng yêu cầu (Request line):

   ```
   GET /index.html HTTP/1.1
   ```

   - `GET`: phương thức HTTP
   - `/index.html`: tài nguyên được yêu cầu
   - `HTTP/1.1`: phiên bản giao thức

2. Header (Tiêu đề): Chứa thông tin bổ sung như loại trình duyệt, định dạng chấp nhận...

   ```
   Host: www.example.com
   User-Agent: Mozilla/5.0
   Accept: text/html
   ```

3. Body (nội dung): (Không bắt buộc, thường dùng trong POST hoặc PUT) – chứa dữ liệu gửi lên server.

Cấu trúc một phản hồi HTTP (HTTP Response)

1. Dòng trạng thái (Status line):

   ```
   HTTP/1.1 200 OK
   ```

   - `HTTP/1.1`: phiên bản
   - `200`: mã trạng thái
   - `OK`: thông điệp trạng thái

2. Header:

   ```
   Content-Type: text/html
   Content-Length: 1024
   ```

3. Body: Nội dung thực tế (HTML, JSON, ảnh, v.v.) phản hồi từ server.

Phương thức HTTP	**Mục đích**	**Mã trạng thái HTTP**	**Ý nghĩa**
GET	Yêu cầu lấy tài nguyên	200	OK (Thành công)
POST	Gửi dữ liệu lên server	301	Moved Permanently (Chuyển hướng)
PUT	Cập nhật tài nguyên	400	Bad Request (Lỗi yêu cầu)
DELETE	Xoá tài nguyên	401	Unauthorized (Chưa xác thực)
PATCH	Cập nhật một phần tài nguyên	403	Forbidden (Bị từ chối truy cập)
HEAD	Giống GET nhưng không có body	404	Not Found (Không tìm thấy)
OPTIONS	Lấy các phương thức được hỗ trợ	500	Internal Server Error (Lỗi server)

Bảng 1: *Các phương thức HTTP phổ biến và các mã trạng thái HTTP phổ biến*

https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference

2 Header và cách trình duyệt quản lí cookie

Header	Ý nghĩa
Host	Chỉ định tên host và cổng của server (ví dụ: www.example.com)
User-Agent	Thông tin về trình duyệt hoặc client gửi request
Accept	Xác định các định dạng dữ liệu mà client có thể xử lý (ví dụ: text/html, application/json)
Accept-Language	Ngôn ngữ ưu tiên của client (ví dụ: en-US, vi-VN)
Accept-Encoding	Các kiểu nén dữ liệu mà client hỗ trợ (ví dụ: gzip, deflate)
Content-Type	Kiểu dữ liệu của body request hoặc response (ví dụ: application/json, text/html)
Content-Length	Độ dài (byte) của nội dung body
Authorization	Thông tin xác thực (thường dùng cho các API, ví dụ: Bearer token)
Cache-Control	Chỉ định cách thức cache của dữ liệu (ví dụ: no-cache, max-age=3600)
Cookie	Gửi cookie từ client lên server
Set-Cookie	Dùng trong response để server gửi cookie xuống client
Location	Dùng trong redirect để chỉ định URL mới
Referer	URL của trang giới thiệu, từ đó người dùng đến trang hiện tại
Connection	Kiểm soát việc giữ kết nối (ví dụ: keep-alive, close)
ETag	Giá trị định danh phiên bản tài nguyên, dùng để cache hiệu quả

Bảng 2: *Các HTTP Header thường gặp*

Quy trình tổng quát khi đăng nhập và sử dụng cookie và cách hoạt động Header:Cookie

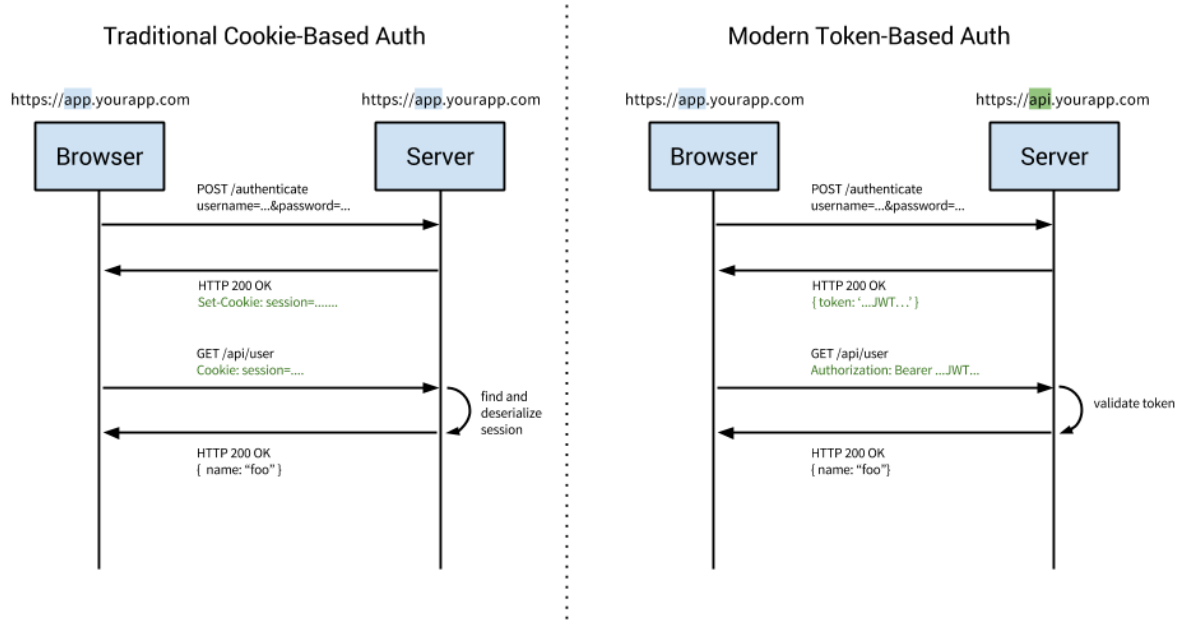
1. Client gửi thông tin đăng nhập
   - Người dùng nhập `username` và `password` vào form.
   - Trình duyệt gửi HTTP POST request chứa thông tin đăng nhập đến server:

     ```
     POST /login HTTP/1.1
     Content-Type: application/x-www-form-urlencoded

     username=alice&password=123456
     ```
2. Server kiểm tra thông tin
   - Server kiểm tra thông tin đăng nhập với database.
   - Nếu hợp lệ, server tạo một session (phiên làm việc).
3. Server gửi Cookie chứa session
   - Server gửi response với header `Set-Cookie:`

     ```
     HTTP/1.1 200 OK
     Set-Cookie: session_id=abc123xyz; Path=/; HttpOnly; Secure
     ```
4. Trình duyệt lưu Cookie
   - Trình duyệt lưu cookie `session_id=abc123xyz`.
5. Các request sau tự động gửi Cookie
   - Trình duyệt tự động gửi Cookie trong các request tiếp theo:

     ```
     GET /profile HTTP/1.1
     Cookie: session_id=abc123xyz
     ```
   - Server dựa vào `session_id` để nhận diện người dùng.



Cookie-based:

1. Client gửi POST /login (username, password)
2. Server tạo session → session_id → lưu ở server
3. Server gửi Set-Cookie: session_id=xyz
4. Client gửi các request sau kèm Cookie: session_id=xyz
5. Server kiểm tra session_id → xác thực user

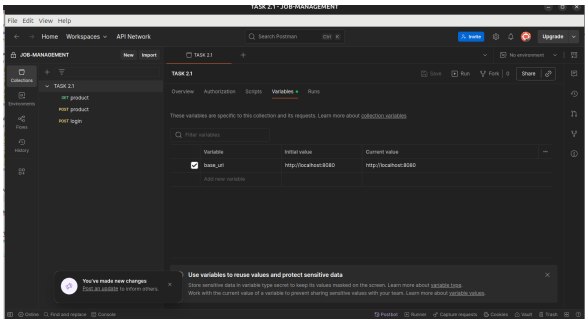
Token-based (JWT): => làm ở task trước

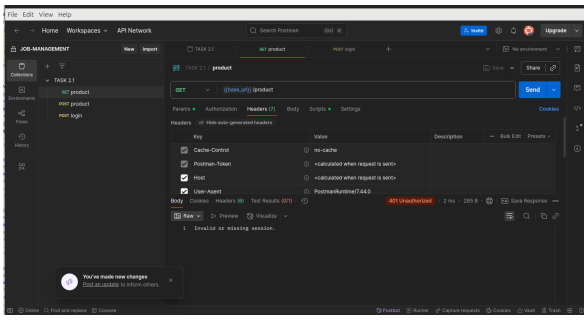
1. Client gửi POST /login (username, password)
2. Server tạo JWT token → trả về cho client
3. Client lưu token (localStorage hoặc Cookie)
4. Client gửi các request sau với header: Authorization: Bearer <token>
5. Server xác thực token (giải mã, kiểm tra chữ ký, claims)

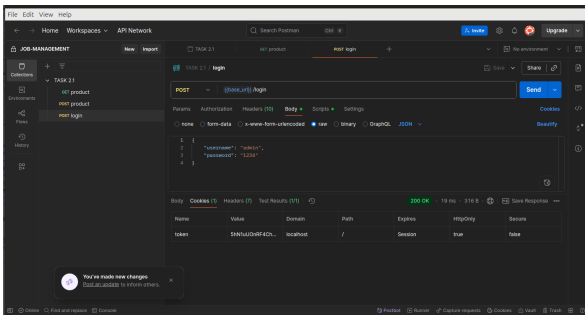
3 Postman

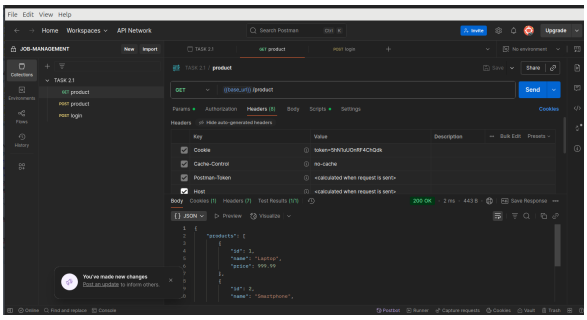
Postman là một công cụ phần mềm phổ biến dùng để kiểm thử (test) API — các giao diện lập trình ứng dụng. Nó giúp các nhà phát triển gửi các yêu cầu HTTP (GET, POST, PUT, DELETE, v.v) đến server và nhận phản hồi để kiểm tra xem API hoạt động đúng hay không.

- **Ảnh 1** — Cách sử dụng biến toàn cục trong Postman để tái sử dụng giá trị trong URL, header hoặc body với cú pháp `{{variable_name}}`.
- **Ảnh 2** — Lấy danh sách sản phẩm thất bại do thiếu cookie xác thực.
- **Ảnh 3** — Tạo session và cookie thành công sau khi đăng nhập qua API.
- **Ảnh 4** — Lấy được danh sách sản phẩm thành công khi có cookie xác thực.







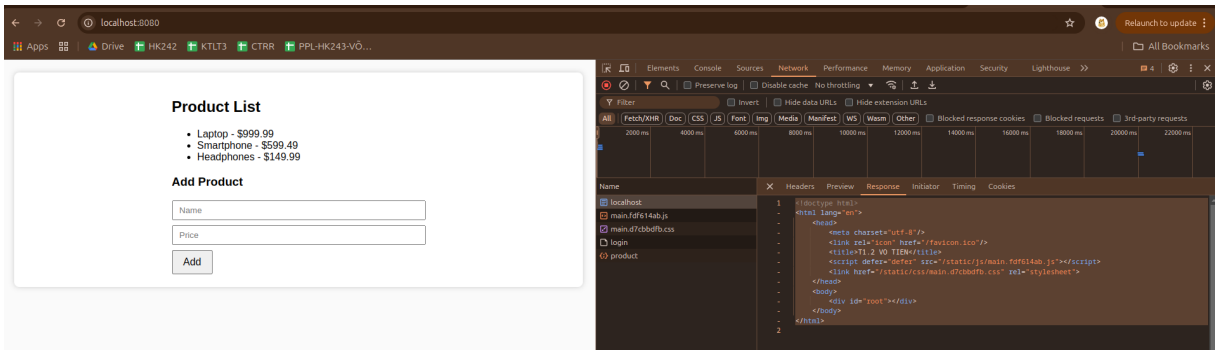


```
GET /product HTTP/1.1
User-Agent: PostmanRuntime/7.44.0
Accept: */*
Cache-Control: no-cache
Postman-Token: 883de2a1-5564-4aae-a2f6-6cffa755a802
Host: localhost:8080
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: token=bvDQb34QfAGa4XLT
```

4 Trình Duyệt

Trình duyệt là phần mềm giúp người dùng truy cập và hiển thị các trang web trên Internet.

- Nó gửi yêu cầu (request) tới máy chủ và nhận về dữ liệu (HTML, CSS, JavaScript, hình ảnh...) để hiển thị.
- Hỗ trợ các tính năng như quản lý cookie, bộ nhớ cache, bảo mật, và chạy mã JavaScript.
- Cung cấp công cụ phát triển (DevTools) giúp debug và kiểm tra giao diện, network, performance.
- Trình duyệt xử lý cơ chế CORS, bảo vệ người dùng khỏi các truy cập trái phép.

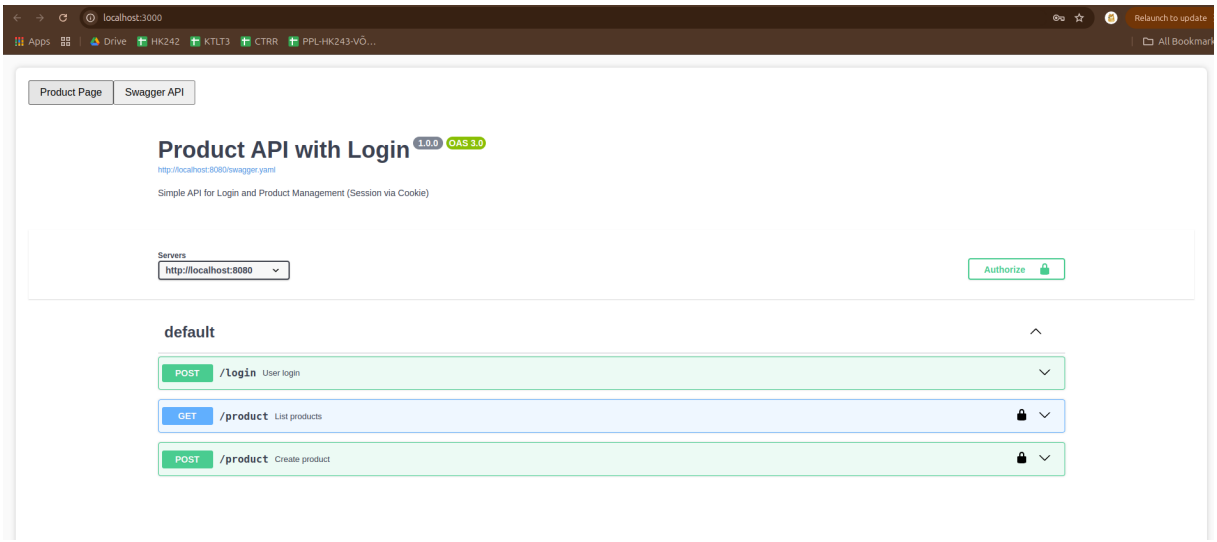



```
GET /static/js/main.fdf614ab.js HTTP/1.1
Host: localhost:8080
Connection: keep-alive
sec-ch-ua-platform: "Linux"
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/129.0.0.
sec-ch-ua: "Google Chrome";v="129", "Not=A?Brand";v="8", "Chromium";v="129"
sec-ch-ua-mobile: ?0
Accept: */*
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: script
Referer: http://localhost:8080/
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: en-US,en;q=0.9,vi;q=0.8
Cookie: token=w7P4reutt2dIS0Zq
```

5 Dùng Swagger

Swagger là **một bộ công cụ mã nguồn mở** giúp thiết kế, xây dựng, mô tả, kiểm thử và tài liệu hóa các API RESTful. Nó rất phổ biến trong cộng đồng phát triển API vì giúp đơn giản hóa việc tạo và duy trì tài liệu API, đồng thời giúp người dùng và lập trình viên dễ dàng hiểu và thử nghiệm API.

Hiện nay, Swagger đã được phát triển thành **OpenAPI Specification (OAS)** — một tiêu chuẩn mô tả API phổ biến nhất hiện tại. Tuy vậy, người ta vẫn quen gọi các công cụ đi kèm là **Swagger tools**.



1. Viết file mô tả API (swagger.yaml hoặc tự sinh).

```yaml
openapi: 3.0.3
info:
  title: Example API with Cookie Auth
  version: 1.0.0

servers:
  - url: http://localhost:8080

components:
  securitySchemes:
    bearerAuth:
      type: http
      scheme: bearer

security:
  - cookieAuth: []

paths:
  /product:
    get:
      .......
```

2. Gửi yêu cầu `POST /login` lấy **token**.
3. Dán token vào nút `Authorize` trong Swagger UI.
4. Chọn endpoint, bấm `Try it out` và `Execute` để gọi API.

Lưu ý: Token sẽ được tự động thêm vào header `Authorization` khi gọi API.

6 Thiết Kế hệ thông

1. **Frontend** Chứa mã nguồn giao diện người dùng (client-side), ví dụ React, Vue hoặc các file tĩnh HTML/CSS/JS phục vụ frontend.
   - `npm install` cài thư viện
   - `npm start` bắt đầu chạy và chỉnh UI cho đẹp hợp lí gì đó
   - `npm run build` build ra file index.html, sau đó trình duyệt lại tự gọi các file js và css nếu cần

   ```
   ## Router.h
   res.setFileBody("./frontend/build/index.html", "text/html");

   ## frontend/build/index.html
   <!doctype html>
   <html lang="en">

   <head>
       <meta charset="utf-8" />
       <link rel="icon" href="/favicon.ico" />
       <title>T1.2 VO TIEN</title>
       <script defer="defer" src="/static/js/main.fdf614ab.js"></script>
       <link href="/static/css/main.d7cbbdfb.css" rel="stylesheet">
   </head>

   <body>
       <div id="root"></div>
   </body>

   </html>
   ```

2. **controllers/**: Chứa các lớp hoặc module xử lý logic nghiệp vụ, nhận request từ router rồi tương tác với models hoặc services để xử lý dữ liệu, trả kết quả về client.
3. **core/**: Chứa các thành phần cốt lõi hoặc cấu hình nền tảng cho dự án
4. **http/**: Chứa các lớp hoặc module liên quan đến xử lý HTTP như Request, Response, hoặc helper để làm việc với giao thức HTTP.
5. **models/**: Chứa các định nghĩa mô hình dữ liệu (data models), thường là các lớp hoặc schema ORM tương tác trực tiếp với database.
6. **public/**: Chứa các file tĩnh phục vụ trực tiếp cho client như hình ảnh, fonts, CSS, JavaScript hoặc file HTML.
7. **router/**: Chứa các định nghĩa tuyến đường (route) cho server, ánh xạ các URL đến controllers tương ứng.
8. **server/**: Chứa mã nguồn liên quan đến cấu hình và khởi chạy server (ví dụ Express hoặc HTTP server), cũng như các phần khởi tạo liên quan.
9. **utils/**: Chứa các hàm tiện ích, helper functions hoặc các module dùng chung phục vụ nhiều phần khác nhau trong dự án.

7 Yêu Cầu

1. Hiện thực **LoginController::handle** để trả về **Set-Cookie**
2. Chạy dự án và thử qua trình duyệt, postman, swagger
3. Hiện thực Frontend và Backend thêm UPDATE và DELETE Product
4. Viết swagger.yaml cho UPDATE và DELETE Product
5. Liệt kê Web và mobile thường dùng các kiểu xác thực phổ biến
6. HTTPS là gì khác gì HTTP
7. Liệt Kê các cổng dùng phổ biến trong hệ thống.
8. Hãy chạy dự án FW cho ngôn ngữ bạn muốn code sau này và thêm Login, API Product giống C++ và swagger tự động vào (các ngôn ngữ hiện nay đều hỗ trợ)