

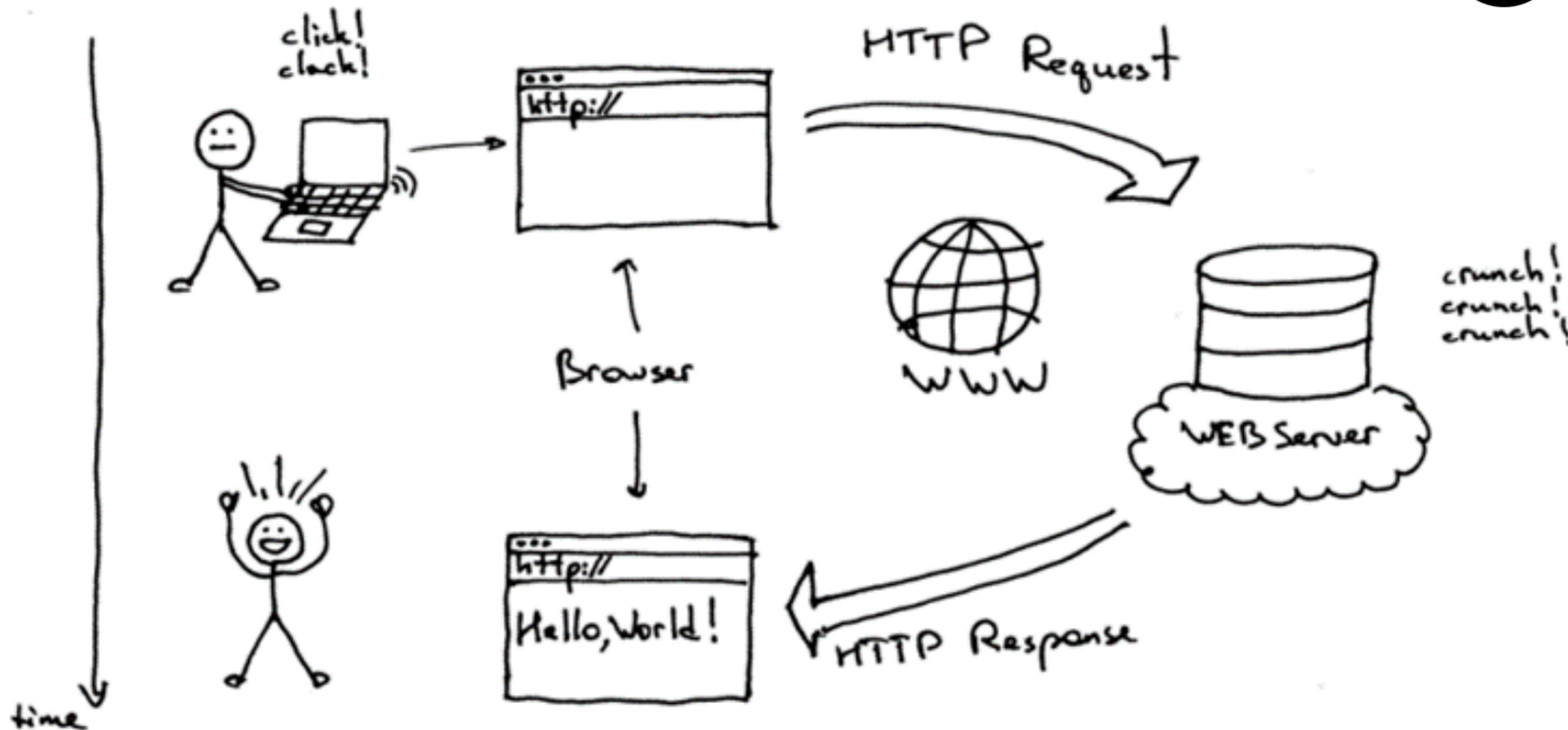


# Spring Boot Overview

# NỘI DUNG CHÍNH

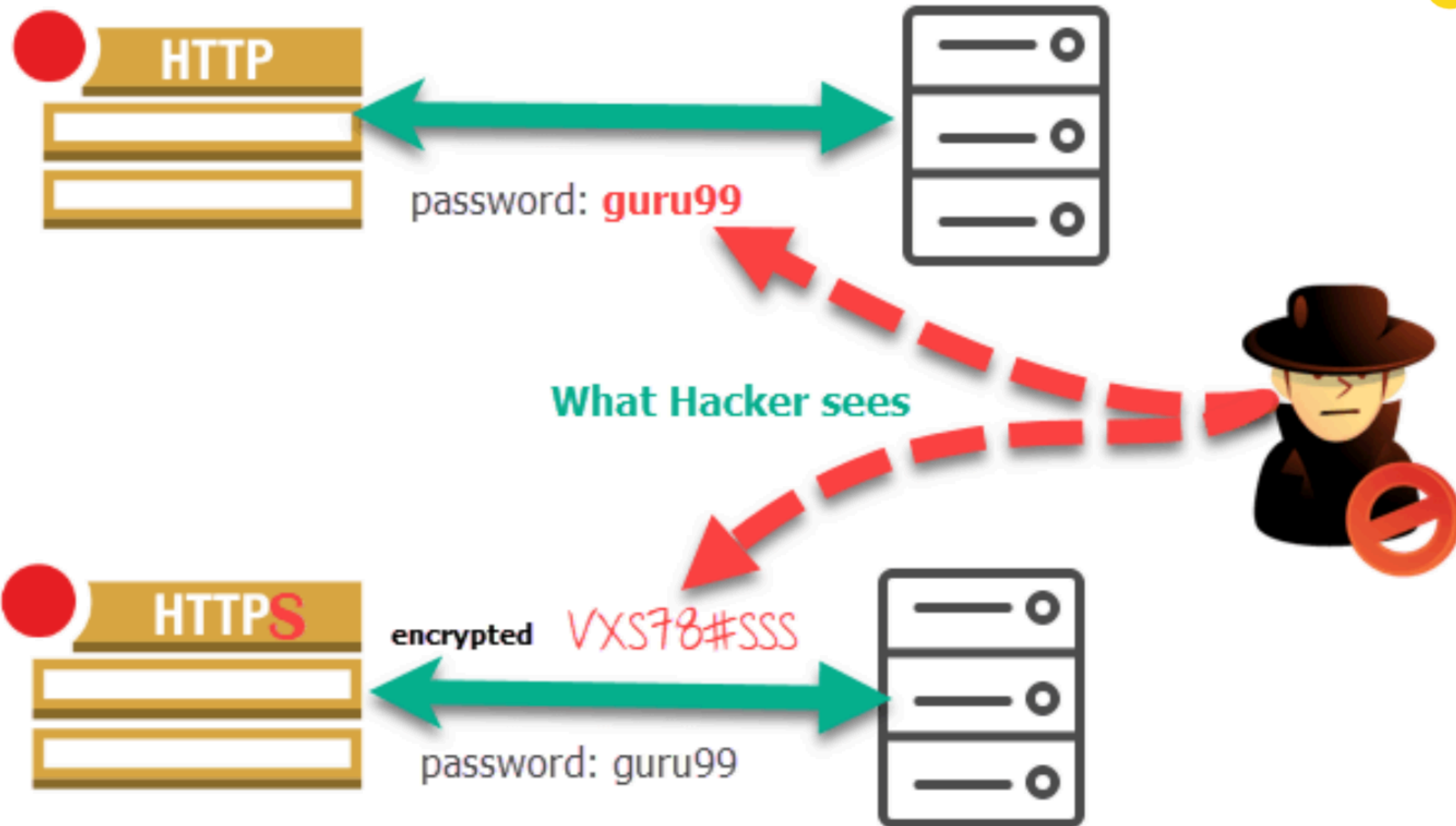
- 1 Phân biệt request/response
- 2 Phân biệt HTTP/HTTPS
- 3 Web Application và Webservice
- 4 Webservice
- 5 Phân biệt Framework và Library
- 6 Spring Boot 3/Postman
- 7 Luyện tập CRUD

# Phân biệt request/response



# Phân biệt HTTP/HTTPS

HTTP/HTTPS



# Phân biệt HTTP/HTTPS



**HTTP:** Hypertext Transfer Protocol – Giao thức truyền tải siêu văn bản

**HTTPS:** **HTTP Security** Là một phần mở rộng của **HTTP**

**Giống nhau:** Đều là 2 giao thức truyền tải siêu văn bản từ **Client** → **Server**

**Khác nhau:** **HTTPS** là giao thức **HTTP** được tích hợp thêm chứng chỉ bảo mật **SSL/TLS** → nhằm mã hóa thông tin để đảm bảo tính bảo mật cho các trang web.



# Web Application và Webservice



- **Website truyền thống** (Web application)
  - Có giao diện.
  - Thường được sử dụng bởi **con người**.
  - Có thể truy cập bởi các thành phần trong giao diện người dùng như button, textbox, form, vv.
- **Webservice**
  - Không có giao diện.
  - Các **ứng dụng** tương tác với nhau.
  - Được truy cập bởi các **HTTP Method** như **POST**, **GET**, **PUT**, **DELETE** và các method khác.

A login form with a white background and rounded corners. It contains two input fields: 'Email address or phone number' and 'Password'. Below the fields is a blue 'Log in' button. Under the button is a link 'Forgotten password?'. At the bottom is a green 'Create new account' button.

Email address or phone number

Password

Log in

[Forgotten password?](#)

Create new account

```
{
  ... "username": "QuangNN",
  ... "password": "admin"
}
```

```
<user>
  ... <username>QuangNN</username>
  ... <password>admin</password>
</user>
```

# Webservice

Flow



- Là tập hợp các giao thức và tiêu chuẩn (HTTP/HTTPS), được sử dụng để trao đổi dữ liệu giữa các ứng dụng hoặc giữa các hệ thống
- Dữ liệu đầu ra thường là **XML** hoặc **JSON**.

## XML

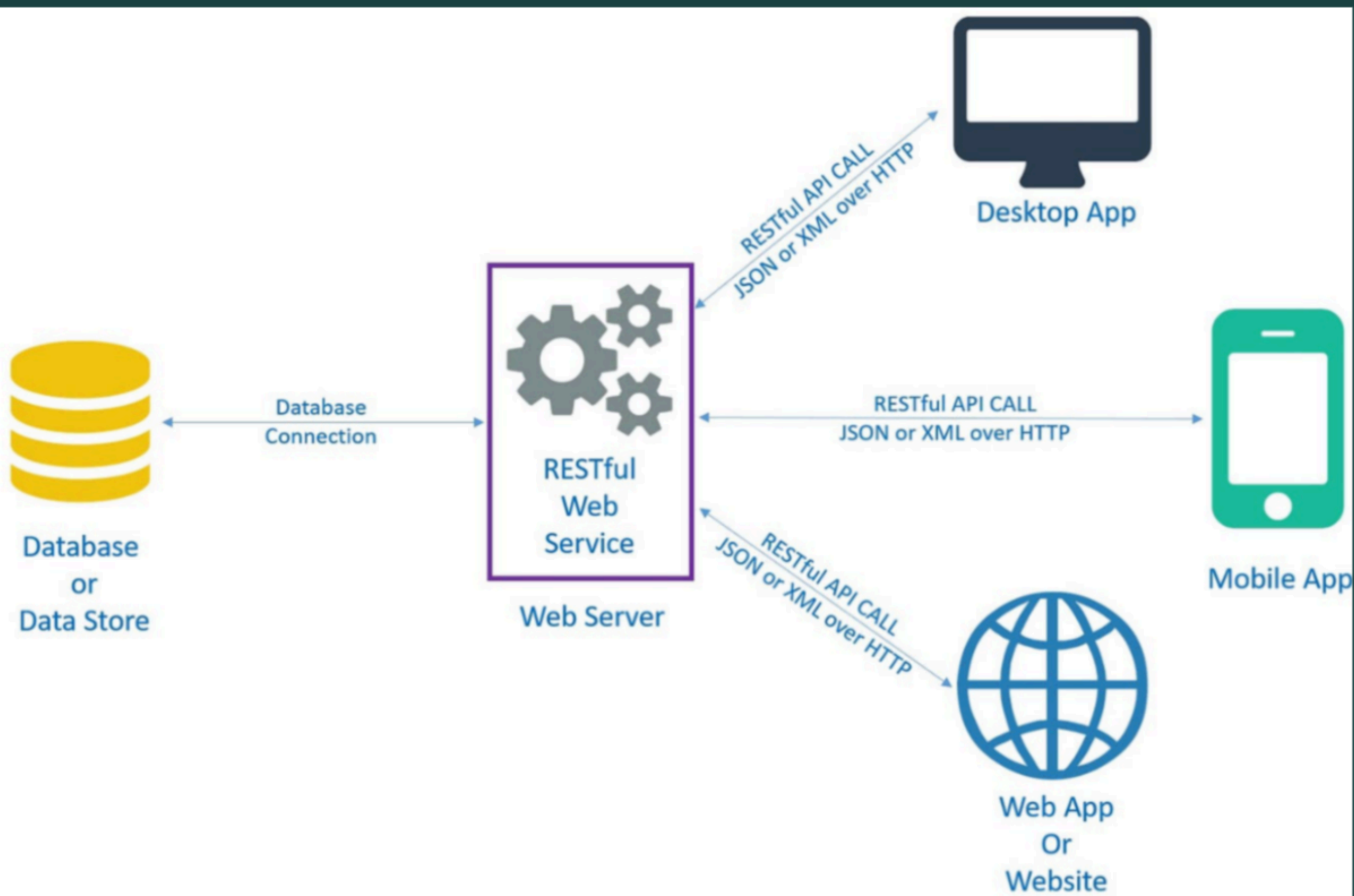
```
<user>
|   ... <username>QuangNN</username>
|   ... <password>admin</password>
|   ...
|   </user>
```

## JSON

```
... "username": "QuangNN"
... "password": "admin"
```

# Webservice

Ưu nhược  
điểm





# Ưu nhược điểm của Webservice

Các loại  
Webservice



- **Ưu điểm**
  - Hoạt động trên nhiều ứng dụng, nền tảng, hệ điều hành, ngôn ngữ khác nhau.
  - Khả năng **tái sử dụng cao**.
  - **Giảm sự phức tạp** của hệ thống và thời gian phát triển, **giảm giá thành** hoạt động, dễ dàng tương tác giữa các hệ thống với nhau.
- **Nhược điểm**
  - Lỗi hoặc dừng hoạt động của một web service có thể gây lỗi và **thiệt hại lớn** trên tất cả các hệ thống sử dụng web service đó.
  - **Đòi hỏi quan tâm đến vấn đề an toàn và bảo mật** nhiều hơn khi sử dụng web service..

# Các loại Webservice

Phương Thức  
HTTP



- **SOAP** (Simple Object Access Protocol)
  - Là giao thức **sử dụng XML** để định nghĩa dữ liệu và truyền dữ liệu thông qua HTTP.
- **REST** (Representational State Transfer)
  - Là một kiểu cấu trúc cung cấp các quy tắc để xây dựng web service.
    - **GET: Lấy thông tin** về một tài nguyên
    - **POST: Tạo mới** một tài nguyên
    - **PUT: Cập nhật hoàn toàn** một tài nguyên
    - **PATCH: Cập nhật một phần** của một tài nguyên.
    - **DELETE: Xóa** một tài nguyên.
  - Định nghĩa dữ liệu dưới dạng **XML hoặc JSON** và truyền dữ liệu thông qua HTTP.

# Phương Thức HTTP



- **GET: Lấy dữ liệu**
  - URL: /students
  - Status: 200
  - Dữ liệu: Trả về danh sách sinh viên
- **POST: Tạo mới**
  - URL: /students
  - Status: 201
  - Dữ liệu: Trả về đối tượng sinh viên vừa tạo
- **PUT: Cập nhật**
  - URL: /students/{id}
  - Status:
    - 200: Trả về đối tượng sinh viên vừa cập nhật
    - 404: Không tìm thấy sinh viên cần cập nhật
- **DELETE: Xóa**
  - URL: /students/{id}
  - Status:
    - 200: Không trả về dữ liệu trong body
    - 404: Không tìm thấy sinh viên cần xóa

# Framework

Framework &  
Library



- Là các ứng dụng phần mềm có tính **trừu tượng**(abstraction).
- **Cung cấp** một **cấu trúc cơ bản** để phát triển phần mềm.
- Lợi ích
  - **Tính tái sử dụng mã**
    - Bao gồm các thư viện và lớp có thể tái sử dụng
  - **Hỗ trợ chuẩn**
    - Tuân theo các chuẩn công nghiệp
  - **Tăng tốc độ phát triển**

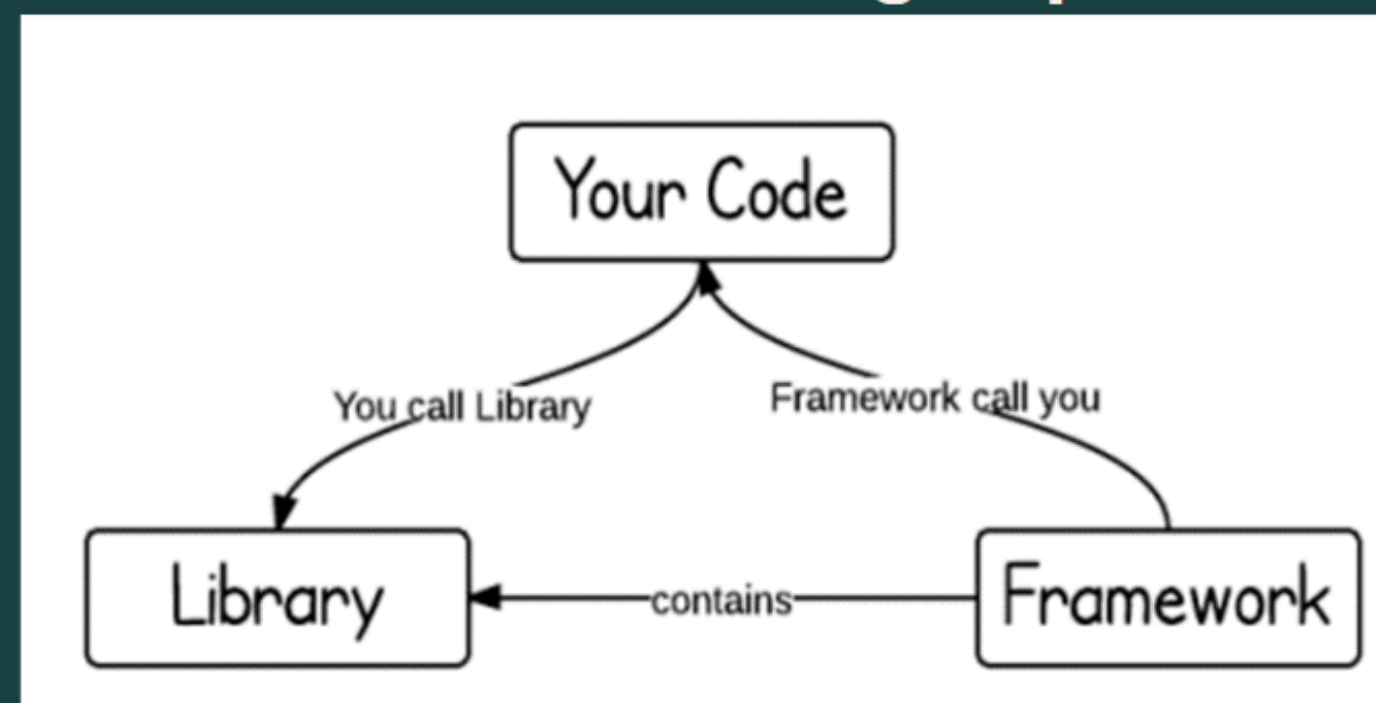


# Framework & Library



- **Framework:**

- Phải tuân theo "**luật chơi**" của Framework.
- Framework định hình cấu trúc của ứng dụng => lập trình viên phải điền mã vào các "**chỗ trống**" mà framework cung cấp.



- **Library:**

- Tự do sử dụng
- **Không bị ràng buộc** bởi cấu trúc hoặc quy tắc nào.



# Spring Boot 3

Tạo project



- **Spring Boot**
  - Nền tảng phát triển ứng dụng **Java** dựa trên **Spring Framework**.
  - Tạo các ứng dụng **độc lập**, có khả năng triển khai trực tiếp.
  - **Tích hợp sẵn** các thành phần cần thiết để phát triển nhanh chóng.
- Tính năng chính của **Spring Boot 3**:
  - Cải tiến **hiệu suất và bảo mật**.
  - Hỗ trợ **từ Java 17**.

# Cách Tạo Project Spring Boot

Các  
Annotation hay  
dùng



- Sử dụng Spring Initializr:

- Truy cập: <https://start.spring.io/>
- Chọn cấu hình (Project, Language, Spring Boot Version)
- Điền thông tin chi tiết của project
  - **Group:** Tổ chức phát triển (thường đảo ngược domain: com.sqc).
  - **Artifact:** Đặt tên cho file JAR/WAR.
  - **Name:** Tên đầy đủ của project.
  - **Package name:** Định danh package chứa mã nguồn chính.
- Chọn các dependencies cần thiết.
- Nhấn "Generate" để tải về project.

# Các Annotation hay dùng

- **@SpringBootApplication**

- Đánh dấu class chính để kích hoạt cấu hình **Spring Boot**.

- **@RestController (@Controller + @ResponseBody)**

- Đánh dấu một class là RESTful controller.

- **@RequestMapping**

Các biến thể của  
@RequestMapping



- Ánh xạ các yêu cầu HTTP đến các phương thức xử lý trong controller.

- **@PathVariable**

- Lấy giá trị của các tham số từ đường dẫn URL.

- **@RequestBody**

- Ánh xạ nội dung của yêu cầu HTTP đến đối tượng Java.

# Các Biến Thể @RequestMapping

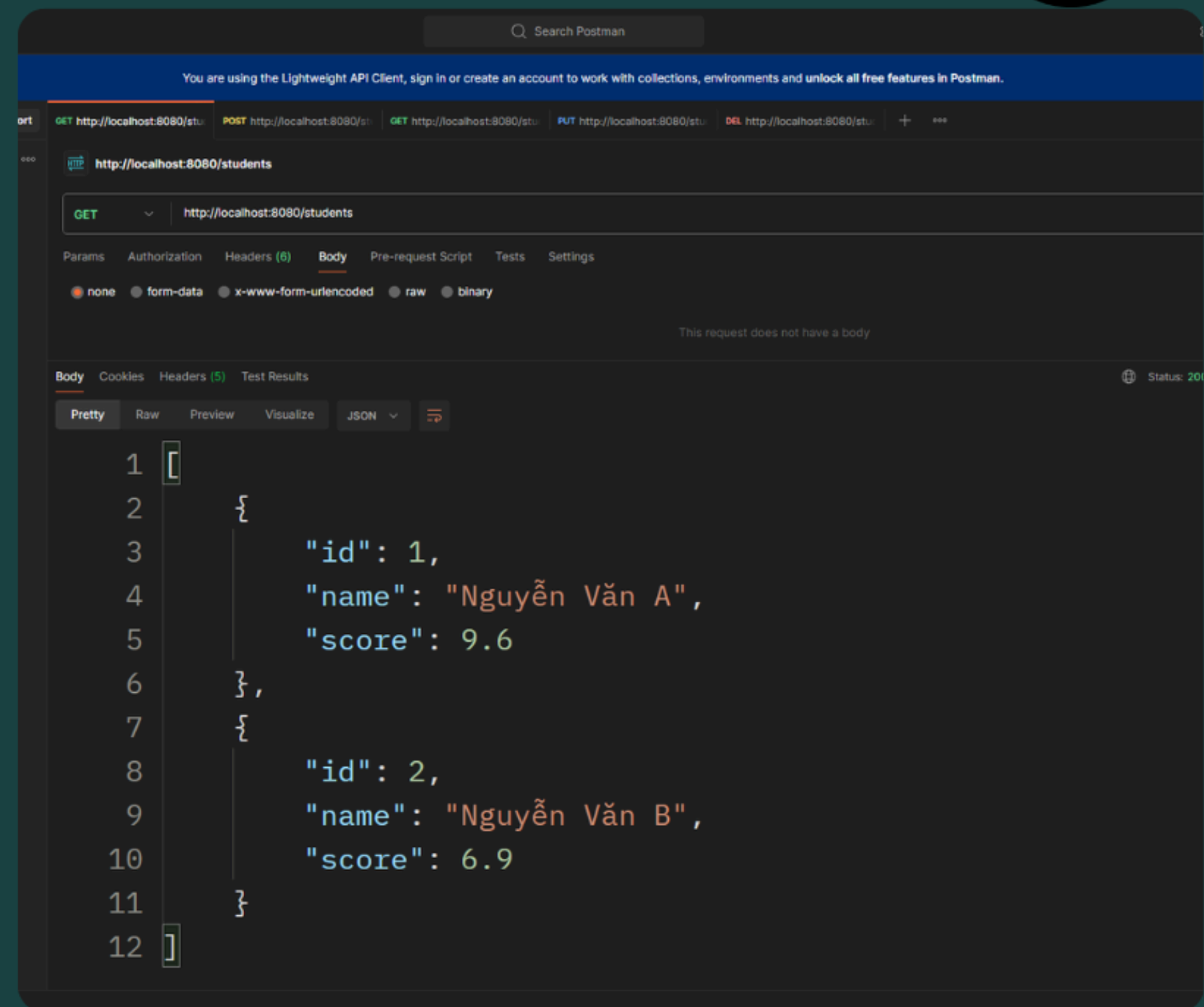


- **@GetMapping** (Xử lý các yêu cầu HTTP GET)
  - `<=> @RequestMapping(method = RequestMethod.GET).`
- **@PostMapping** (Xử lý các yêu cầu HTTP POST)
  - `<=> @RequestMapping(method = RequestMethod.POST).`
- **@PutMapping** (Xử lý các yêu cầu HTTP PUT)
  - `<=> @RequestMapping(method = RequestMethod.PUT).`
- **@DeleteMapping** (Xử lý các yêu cầu HTTP DELETE)
  - `<=> @RequestMapping(method = RequestMethod.DELETE).`

# Postman



- Công cụ phát triển và **kiểm thử API**.
- Hỗ trợ gửi các yêu cầu HTTP (GET, POST, PUT, DELETE, etc.) và kiểm tra các phản hồi từ server.
- Giao diện thân thiện, dễ sử dụng.























# CRUD Student



- **C: Create** => Tạo mới
- **R: Read** => Lấy tài nguyên
- **U: Update** => Cập nhật
- **D: Delete** => Xóa

## Student Details

[Add New Student](#)

#	Name	Address	City	Country	Actions
1	Rual Octo	Deban Steet	Newyork	USA	  
2	Demark	City Road.13	Dubai	UAE	  
3	Richa Deba	Ocol Str. 57	Berlin	Germany	  
4	James Cott	Berut Road	Paris	France	  
5	Dheraj	Bulf Str. 57	Delhi	India	  
6	Maria James	Obere Str. 57	Tokyo	Japan	  



DATABASE

**THANK  
YOU**