



FPT POLYTECHNIC



www.poly.edu.vn



Conceive Design Implement Operate

RESTFUL WEB API 1

GIẢNG VIÊN: NGUYỄN NGHIỆM

- ❑ Giới thiệu WebService và RESTful Web API
- ❑ Firebase REST API
- ❑ REST Consumer
 - ❖ Postman
 - ❖ Axios

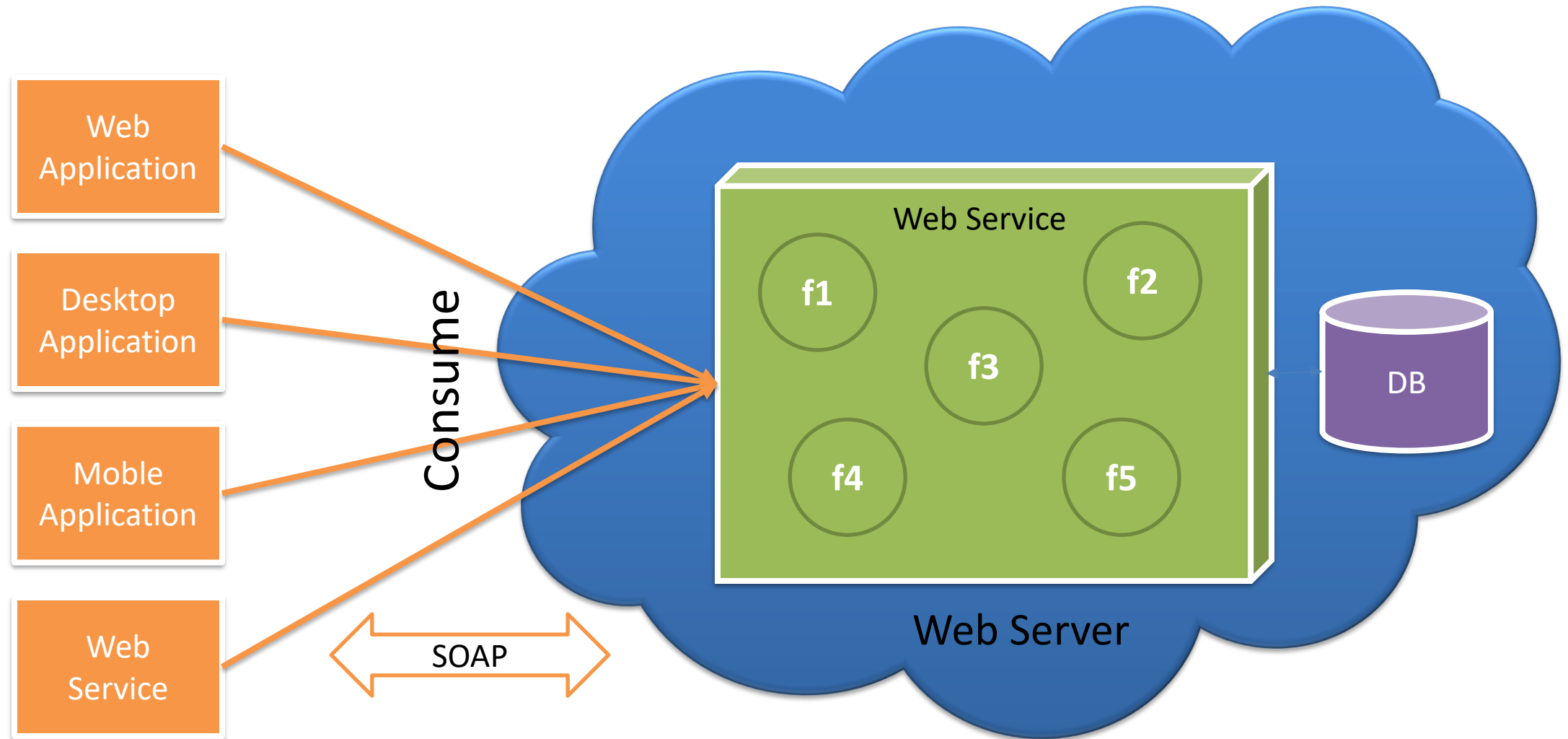




WEB SERVICE



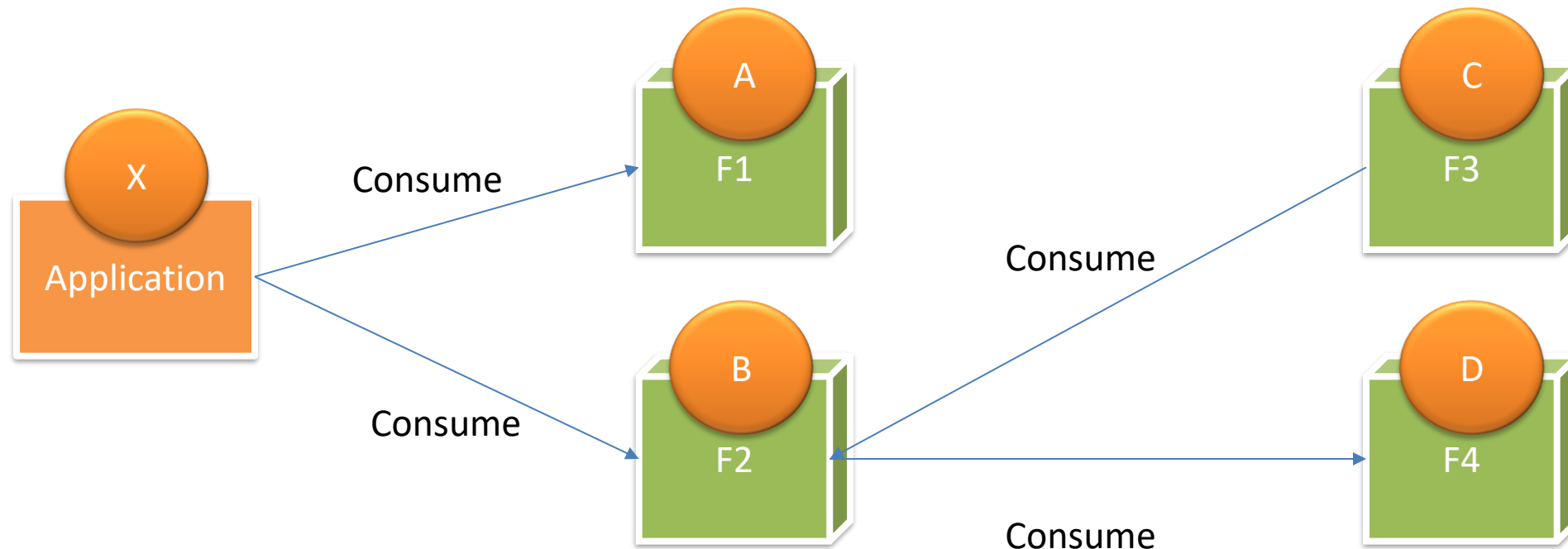
- ❑ Web Service là dịch vụ được triển khai trên web cung cấp các hoạt động (operation hay web function) phục vụ cho các ứng dụng.
- ❑ Ứng dụng có thể kết nối để tích hợp các operation nhằm thực hiện các chức năng cần thiết của ứng dụng.
- ❑ Các ứng dụng có thể thuộc nhiều thể loại khác nhau và được viết bởi bất kỳ ngôn ngữ lập trình nào như
 - ❖ Desktop Application
 - ❖ Mobile Application
 - ❖ Web Application
 - ❖ Web Service khác
- ❑ Dữ liệu truyền thông giữa Client và Web Service theo giao thức SOAP (HTTP+XML)



Web Service Consumers

Web service operations là các phương thức cho phép các ứng dụng sử dụng qua web

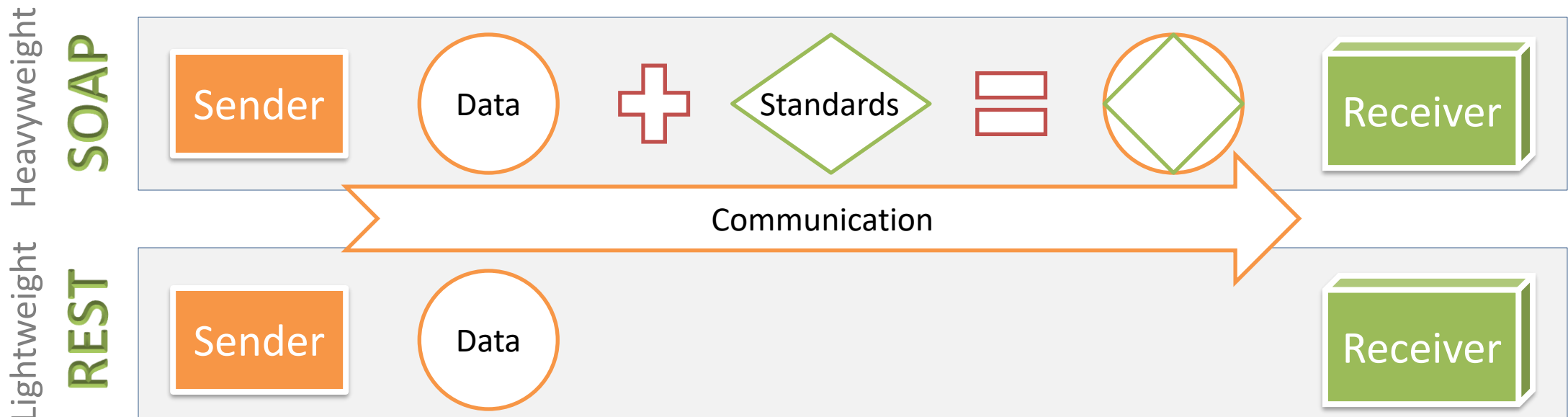
- ❑ Thông qua web service, các doanh nghiệp đối tác có thể thực hiện cộng tác, trao đổi dữ liệu với nhau một cách dễ dàng
- ❑ Ví dụ: Doanh nghiệp C và D cung cấp các function F3, F4 cho doanh nghiệp B sử dụng. Doanh nghiệp X thì được cung cấp các function F1 và F2 từ các doanh nghiệp A và B



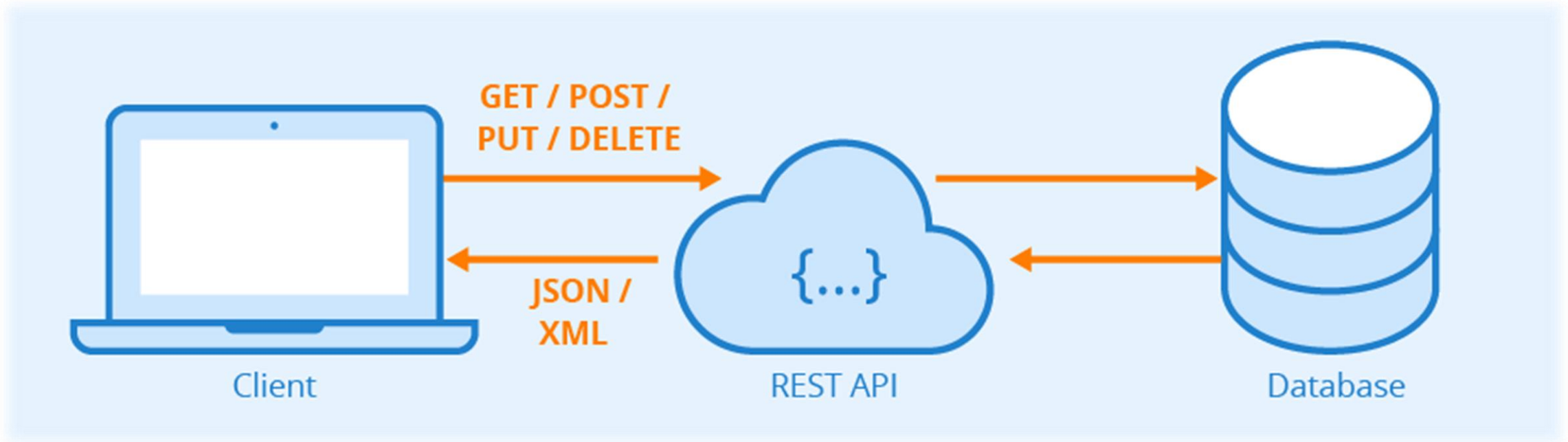


REST & REST API

- ❑ **REST** (**R**epresentational **S**tate **T**ransfer) là các quy ước biểu diễn dữ liệu chuyển đổi giữa các ứng dụng.
- ❑ **REST API** (**A**pplication **P**rogramming **I**nterface) (còn gọi là **RESTful Web API**) là Web Service hoạt động theo các tiêu chuẩn:
 - ❖ Operations: **GET**, **POST**, **PUT**, **DELETE**...
 - ❖ Transfer Data: **JSON** or **XML**/HTML



REST API COMMUNICATION MODEL



❑ Operations

- ❖ GET: (url) => response
- ❖ POST: (url, data) => response
- ❖ PUT: (url, data) => response
- ❖ DELETE: (url) => response (null)

❑ Transfer Data

- ❖ Dữ liệu trao đổi giữa Client và REST API là **JSON/XML**



FIREBASE REST API

- ❑ Firebase là một nền tảng giúp người dùng xây dựng và phát triển các web, ứng dụng và trò chơi mà họ yêu thích.
- ❑ Firebase được hỗ trợ bởi Google và tin cậy bởi hàng triệu doanh nghiệp trên thế giới.
- ❑ Ở phạm vi lưu trữ thì Firebase được xem là nơi cung cấp cơ sở dữ liệu trực tuyến thời gian thực (Realtime Database)
- ❑ Firebase cũng là nơi cung cấp RESTful Web API được viết sẵn hoạt động với dữ liệu lưu trữ trên Realtime Database.
- ❑ Để có thể sử dụng RESTful Web API của Firebase bạn cần tạo 1 dự án, tạo Realtime Database và cấu hình cho phép read và write dữ liệu (Theo hướng dẫn của slide sau)

❑ Bạn cần có tài khoản gmail

❑ Tạo Realtime Database

- ❖ Bước 1: <https://console.firebase.google.com>
- ❖ Bước 2: tạo 1 dự án (project)
- ❖ Bước 3: Vào mục **Realtime Database**
- ❖ Bước 4: Chọn tab **Rules** và thay đổi luật đọc/ghi như sau

```
{  
  "rules": {  
    ".read": true,  
    ".write": true  
  }  
}
```

firebase

Project Overview

Project shortcuts

Hosting

Realtime Database

What's new

AI Logic **NEW**

Product categories

Build

Run

fpolyedu

Realtime Database

Need help with Realtime Database? Ask Gemini

Data Rules Backups Usage Extensions

Protect your Realtime Database from threats such as billing fraud

Đường dẫn cơ sở

<https://fpolyedu.firebaseio.com>

! Your security rules are defined as public, so anyone can steal, modify or delete data in your database

`https://fpolyedu.firebaseio.com/: null`

- ❑ Realtime Database trong dự án fpolyedu có đường dẫn cơ sở là
- ❖ <https://fpolyedu.firebaseio.com>

❑ GET: Truy vấn tất cả và truy vấn theo key

❖ URL: <https://fpolyedu.firebaseio.com/students.json>

❖ URL: <https://fpolyedu.firebaseio.com/students/{key}.json>

❑ POST: Thêm mới

❖ URL: <https://fpolyedu.firebaseio.com/students.json>

❖ Data: {...}

```
{  
  "id": "SV01",  
  "name": "Sinh viên  
01",  
  "mark": 6.5,  
  "gender": false  
}
```

❑ PUT: Cập nhật

❖ URL: <https://fpolyedu.firebaseio.com/students/{key}.json>

❖ Data: {...}

❑ DELETE: Xóa

❖ URL: <https://fpolyedu.firebaseio.com/students/{key}.json>



POSTMAN



POSTMAN

- ☐ Postman là một công cụ dùng để tương tác với REST API. Bạn cần tải và cài vào máy trước khi thực hiện.
- ☐ Postman cung cấp đầy đủ các tương tác với REST API bằng các hình thức GET, POST, PUT, DELETE...

❑ POST (URL, Data) được sử dụng để thêm mới dữ liệu vào Realtime Database

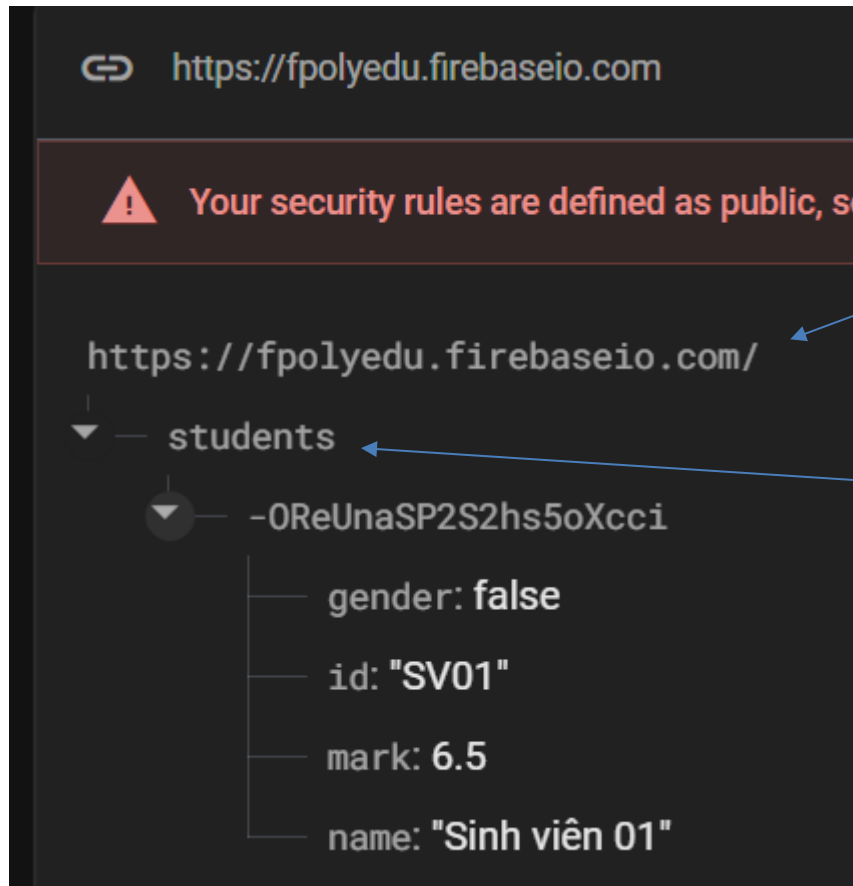
The screenshot illustrates a REST client interface with the following components and annotations:

- 1. Operation:** The **POST** method is selected in the top left.
- 2. URL:** The URL `https://fpolyedu.firebaseio.com/students.json` is entered in the top bar.
- 3.1. Nhập data:** The **Body** tab is selected, and the **raw** radio button is chosen.
- 3.2. Định dạng dữ liệu:** The **JSON** format is selected from the dropdown menu.
- 3.3. Data gửi lên REST API:** The JSON payload is entered in the body field:

```
{
  "id": "SV01",
  "name": "Sinh viên 01",
  "mark": 6.5,
  "gender": false
}
```
- 4.1. Kết quả trả về của POST là key tự tăng:** The response body shows the new key assigned to the `name` field:

```
{
  "name": "-0ReUnaSP2S2hs5oXcci"
}
```
- 4.2. Mã trạng thái trả về:** The status bar indicates a **200 OK** response.

❑ Realtime Database sau khi thực hiện POST



`https://fpolyedu.firebaseio.com/students.json`

`https://fpolyedu.firebaseio.com`

`students.json`

```
{  
  "id": "SV01",  
  "name": "Sinh viên 01",  
  "mark": 6.5,  
  "gender": false  
}
```

```
{"id": "SV01", "name": "Sinh viên 01", "mark": 6.5, "gender": false}
```

```
{"id": "SV02", "name": "Sinh viên 02", "mark": 7.5, "gender": false}
```

```
{"id": "SV03", "name": "Sinh viên 03", "mark": 8.5, "gender": false}
```

```
{"id": "SV04", "name": "Sinh viên 04", "mark": 9.5, "gender": true}
```

```
{"id": "SV05", "name": "Sinh viên 05", "mark": 5.5, "gender": true}
```

Realtime Database sau khi POST 5 sinh viên

```
https://fpolyedu.firebaseio.com/  
└─ students  
  ├── -0ReUnaSP2S2hs5oXcci  
  ├── -0Re_wrD_2hVSTzGjMJ0  
  ├── -0Re_zNPXY0yhvSM9_uh  
  ├── -0Rea4d05euKAPMK5PQu  
  └── -0Rea8s7Tj1iCiV0TnAz
```

GET

https://fpolyedu.firebaseio.com/students.json

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (8) Test Results

200 OK 518 ms 722 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "-0ReUnaSP2S2hs5oXcci": {
3     "gender": false,
4     "id": "SV01",
5     "mark": 6.5,
6     "name": "Sinh viên 01"
7   },
8   "-0Re_wiD_2hVSTzGjMJ0": {
9     "gender": false,
10    "id": "SV02",
11    "mark": 7.5,
12    "name": "Sinh viên 02"
13  }
```

The screenshot shows a REST client interface with a GET request to `https://fpolyedu.firebaseio.com/students/-OReUnaSP2S2hs5oXcci.json`. The response is a JSON object with the following data:

```
{  "gender": false,  "id": "SV01",  "mark": 6.5,  "name": "Sinh viên 01"}
```

The interface includes tabs for Params, Authorization, Headers (8), Body, Pre-request Script, Tests, Settings, and Cookies. The Body tab is active, showing the JSON response in a Pretty format. The status bar indicates a 200 OK response with a 519 ms latency and 354 B of data.

❏ `https://fpolyedu.firebaseio.com/students/-OReUnaSP2S2hs5oXcci.json`

PUT <https://fpolyedu.firebaseio.com/students/-OReUnaSP2S2hs5oXcci.json> Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary JSON Beautify

```
1 {
2   "id": "SV01",
3   "name": "Sinh viên 500",
4   "mark": 10,
5   "gender": false
6 }
```

Body Cookies Headers (8) Test Results 200 OK 521 ms 354 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "gender": false,
3   "id": "SV01",
4   "mark": 10,
5   "name": "Sinh viên 500"
6 }
```

<https://fpolyedu.firebaseio.com/students/-OReUnaSP2S2hs5oXcci.json>

```
students
└── -OReUnaSP2S2hs5oXcci
    ├── gender: false
    ├── id: "SV01"
    ├── mark: 10
    └── name: "Sinh viên 500"
```

The screenshot shows a REST client interface with a DELETE method selected for the URL `https://fpolyedu.firebaseio.com/students/-OReUnaSP2S2hs5oXcci.json`. The response status is 200 OK, with a response time of 458 ms and a body size of 295 B. The response body is displayed as `null`.

`https://fpolyedu.firebaseio.com/students/-OReUnaSP2S2hs5oXcci.json`

-OReUnaSP2S2hs5oXcci Đã bị xóa

```
students
├── -ORe_wrD_2hVSTzGjMJ0
├── -ORe_zNPXY0yhvSM9_uh
├── -ORea4d05euKAPMK5PQu
├── -ORea8s7Tj1iCiV0TnAz
```




AXIOS

❑ Axios là thư viện JavaScript cung cấp các phương thức tương tác với REST API

❖ <https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js>

❑ Các phương thức tương tác với REST API

❖ GET: axios.**get**(url).**then**(resp => {}).**catch**(error => {})

❖ POST: axios.**post**(url, data).**then**(resp => {}).**catch**(error => {})

❖ PUT: axios.**put**(url, data).**then**(resp => {}).**catch**(error => {})

❖ DELETE: axios.**delete**(url).**then**(resp => {}).**catch**(error => {})

Trong đó: axios là đối tượng được cấp bởi thư viện. Các phương thức

➤ get(), post(), put(), delete() gửi request đến REST API

➤ then(), catch() là các callback được gọi khi nhận được response từ REST API. Nếu thành công thì then() được gọi, ngược lại thì catch() được gọi

```
var url = 'https://fpolyedu.firebaseio.com/students.json';  
axios.get(url).then(resp => {  
  console.log("Success", resp.data);  
}).catch(error => {  
  console.log("Error", error);  
});
```

Success

▼ Object **i**

- ▶ -ORe_wrD_2hVSTzGjMJ0: {gender: false, id: 'SV02', mark: 7.5, name: 'Sinh viên 02'}
- ▶ -ORe_zNPXY0yhvSM9_uh: {gender: false, id: 'SV03', mark: 8.5, name: 'Sinh viên 03'}
- ▶ -ORea4d05euKAPMK5PQu: {gender: true, id: 'SV04', mark: 9.5, name: 'Sinh viên 04'}
- ▶ -ORea8s7Tj1iCiV0TnAz: {gender: true, id: 'SV05', mark: 5.5, name: 'Sinh viên 05'}
- ▶ [[Prototype]]: Object

```
var key = "-ORe_wrD_2hVSTzGjMJ0";  
var url = `https://fpolyedu.firebaseio.com/students/${key}.json`;  
axios.get(url).then(resp => {  
  console.log("Success", resp.data);  
}).catch(error => {  
  console.log("Error", error);  
})
```

Success

```
▼ {gender: false, id: 'SV02', mark: 7.5, name: 'Sinh viên 02'}  
  gender: false  
  id: "SV02"  
  mark: 7.5  
  name: "Sinh viên 02"  
  ► [[Prototype]]: Object
```

```
var url = 'https://fpolyedu.firebaseio.com/students.json';  
var data = {  
  "id": "SV006",  
  "name": "Sinh viên 006",  
  "mark": 4.5,  
  "gender": true  
}  
axios.post(url, data).then(resp => {  
  console.log("Success", resp.data);  
}).catch(error => {  
  console.log("Error", error);  
})
```

Success

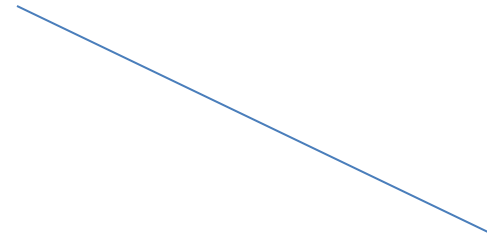
▼ {name: '-ORek1tom3NFdFwnEeV0'}
 name: "-ORek1tom3NFdFwnEeV0"
 ► [[Prototype]]: Object

```
var key = "-ORe_wrD_2hVSTzGjMJ0";  
var url = `https://fpolyedu.firebaseio.com/students/${key}.json`;  
var data = {  
  "id": "SV006",  
  "name": "Sinh viên 600",  
  "mark": 8.5,  
  "gender": true  
}  
  
axios.put(url, data).then(resp => {  
  console.log("Success", resp.data);  
}).catch(error => {  
  console.log("Error", error);  
})
```

Success

```
▼ {gender: true, id: 'SV006', mark: 8.5, name: 'Sinh viên 600'}  
  gender: true  
  id: "SV006"  
  mark: 8.5  
  name: "Sinh viên 600"  
  ► [[Prototype]]: Object
```

```
var key = "-ORe_wrD_2hVSTzGjMJ0";  
var url = `https://fpolyedu.firebaseio.com/students/${key}.json`;  
axios.delete(url).then(resp => {  
  console.log("delete() success", resp.data);  
}).catch(error => {  
  console.log("delete() error", error);  
})
```



```
delete() success  
null
```

- ✓ Giới thiệu WebService và RESTful Web API
- ✓ Firebase REST API
- ✓ REST Consumer
 - ✓ Postman
 - ✓ Axios





FPT Education

FPT POLYTECHNIC

Thank you