



FPT POLYTECHNIC



Conceive Design Implement Operate

BASIC RESTFUL API

GIẢNG VIÊN: NGUYỄN NGHIỆM

www.poly.edu.vn

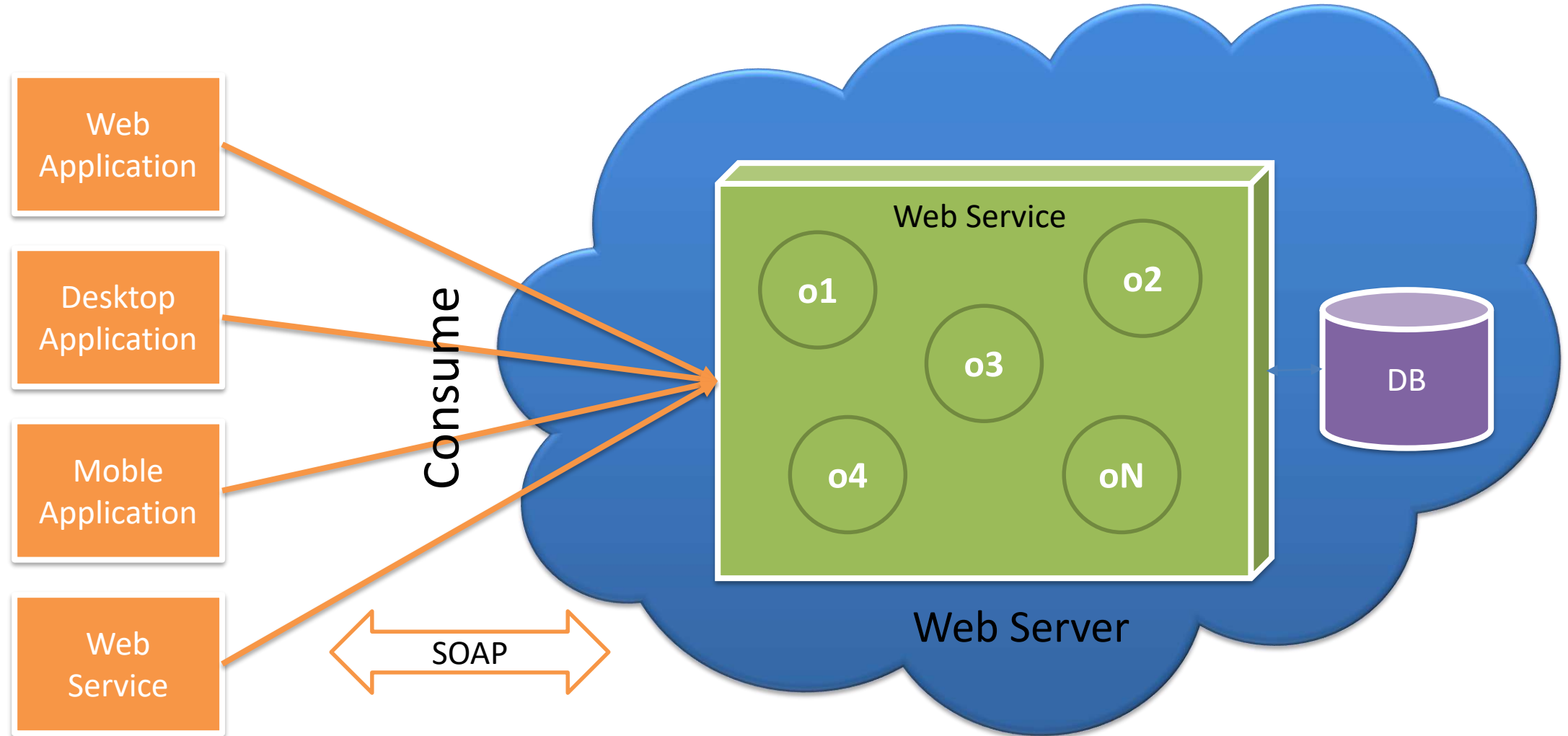
- Web Service & REST API
- Firebase REST API
- Consume REST API with Postman
- Consume REST API with AngularJS
- Consume REST API with java.net.URL
- Consume REST API with RestTemplate





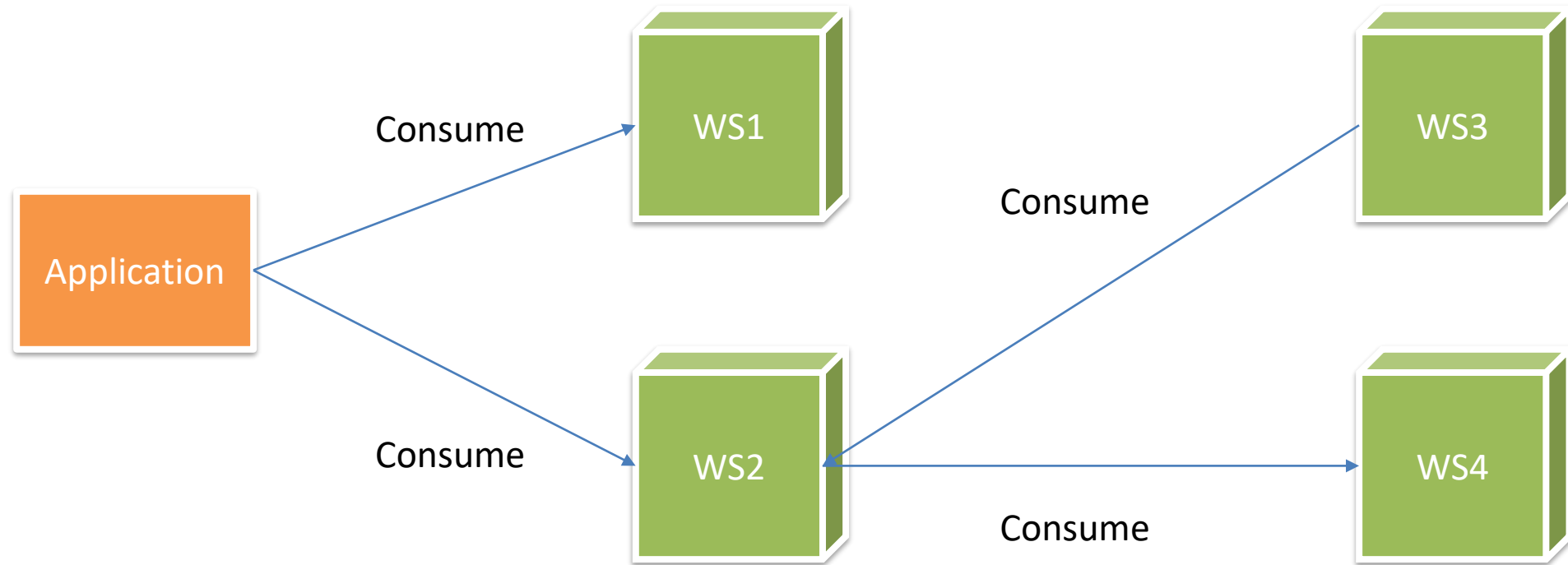
WEB SERVICE





Web Service Consumers

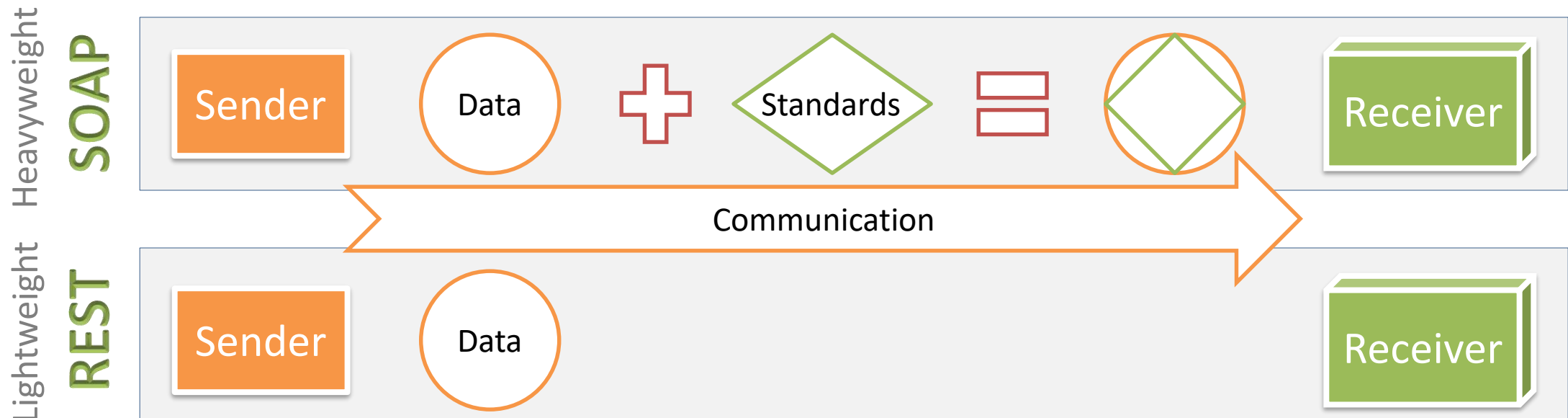
Web service operations là các phương thức cho phép các ứng dụng sử dụng qua web



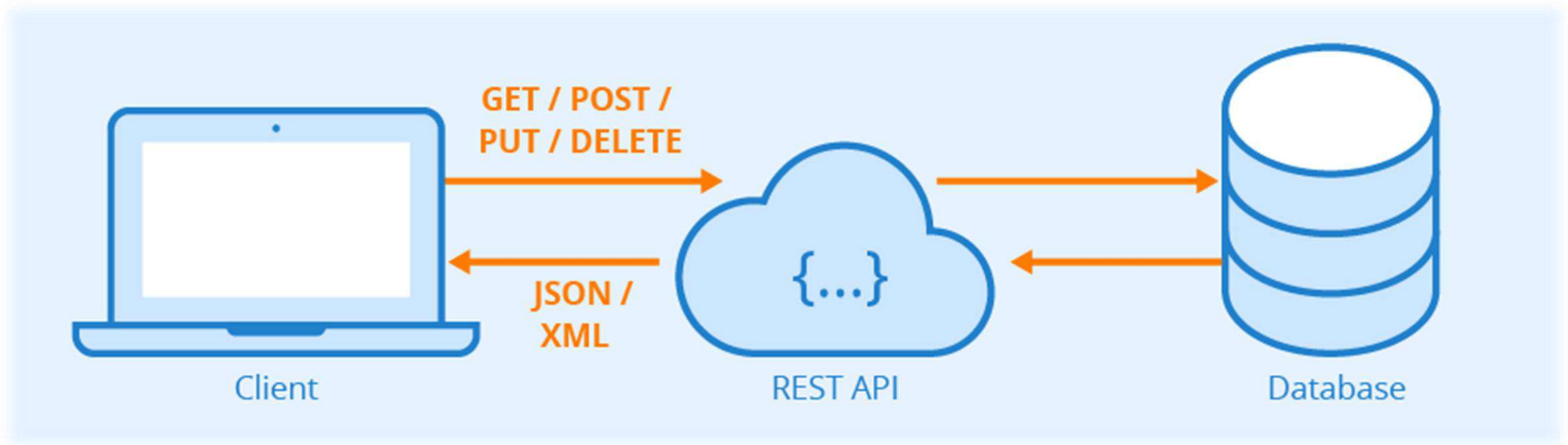


REST & REST API

- ❑ **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** 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 & POSTMAN

Firestore

Project Overview

Build

- Authentication
- Cloud Firestore
- Realtime Database**
- Storage
- Hosting
- Functions
- Machine Learning

Release & Monitor

Crashlytics, Performance, Test La...

Extensions

Spark
Free \$0/month

Upgrade

Poly Java 6

Realtime Database

Data Rules Backups Usage

Prototype and test end-to-end with the Local Emulator Suite, now with

RESTful API base URL

<https://poly-java-6-d4e0b-default-rtdb.firebaseio.com/>

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

poly-java-6-d4e0b-default-rtdb

```

- users
  - PS09013
    - contact
      - email: "phuongntdps09013@fpt.edu.vn"
      - phone: "858551822"
      - id: "PS09013"
      - name: "Nguyễn Thị Diễm Phươn
    - PS09585
  
```

Data structure

```
users: {
  "PS09013": {
    "id": "PS09013",
    "name": "Nguyễn Thị Diễm Phương",
    "contact": {
      "email": "phuongntdps09013@fpt.edu.vn",
      "phone": "858551822"
    }
  }
  ...
}
```



□ EntryPoint URLs

<https://poly-java-6-d4e0b-default-rtdb.firebaseio.com/>

[...users.json](https://poly-java-6-d4e0b-default-rtdb.firebaseio.com/users.json) => **all users**

[...users/PS09013.json](https://poly-java-6-d4e0b-default-rtdb.firebaseio.com/users/PS09013.json) => **one user**

[...users/PS09013/name.json](https://poly-java-6-d4e0b-default-rtdb.firebaseio.com/users/PS09013/name.json) => **name**

[...users/PS09013/contact.json](https://poly-java-6-d4e0b-default-rtdb.firebaseio.com/users/PS09013/contact.json) => **contact**

[...users/PS09013/contact/email.json](https://poly-java-6-d4e0b-default-rtdb.firebaseio.com/users/PS09013/contact/email.json) => **email**

GET

POST

PUT

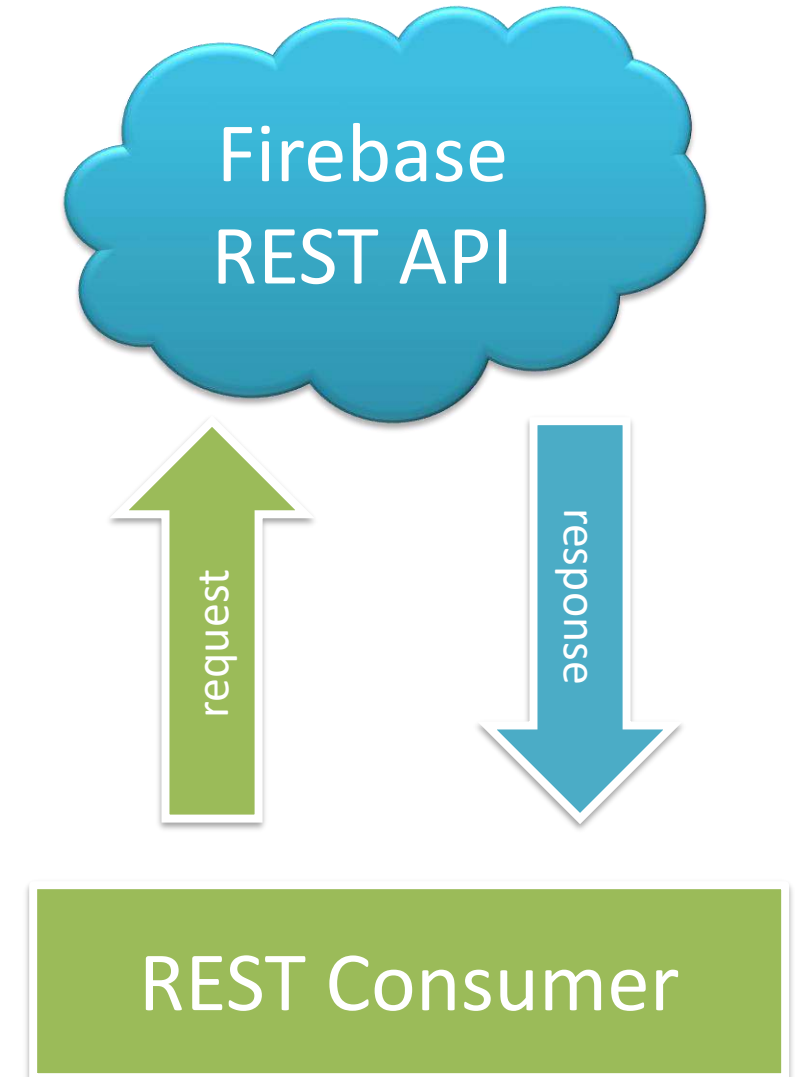
DELETE

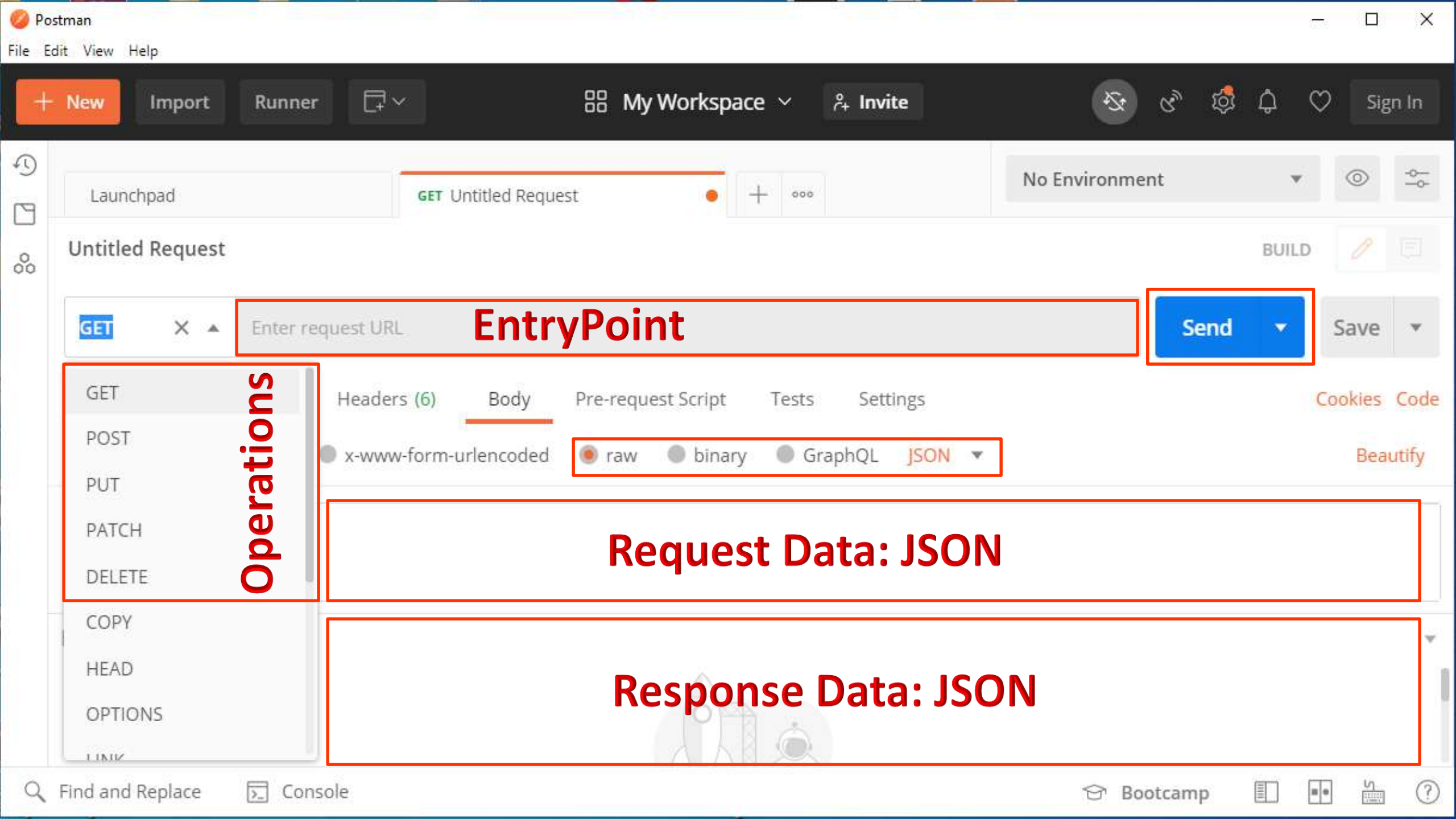
EntryPoint URL = "https://.../users/PS09013.json"

Data Structure = {
 "id": "?",
 "name": "?",
 "contact": {
 "email": "?",
 "phone": "?"
 }
}

❑ Operations

- ❖ **GET:** *request* (**url**) => *response* (**user**)
- ❖ **POST:** *request* (**url, user**) => *response* (**key**)
- ❖ **PUT:** *request* (**url, user**) => *response* (**user**)
- ❖ **DELETE:** *request* (**url**) => *response* (**null**)





EntryPoint

Send

Operations

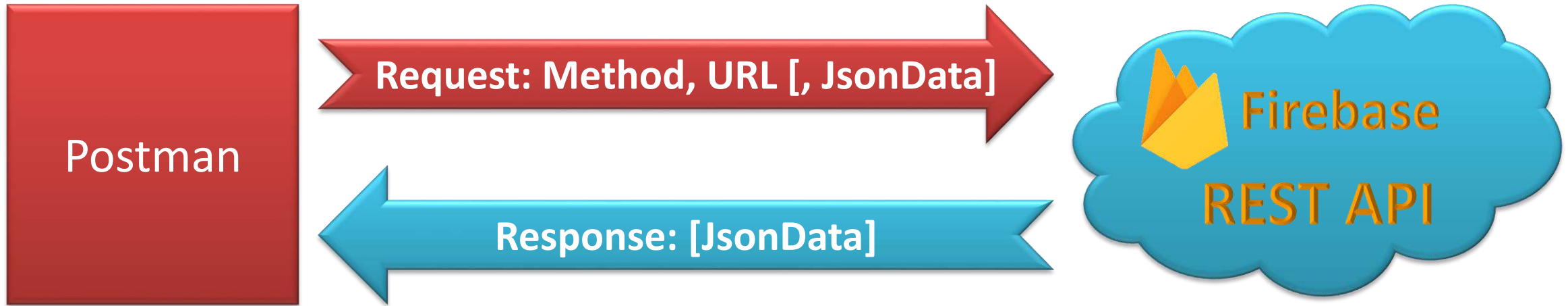
Request Data: JSON

Response Data: JSON



DEMO

REST API EXCHANGE MODEL

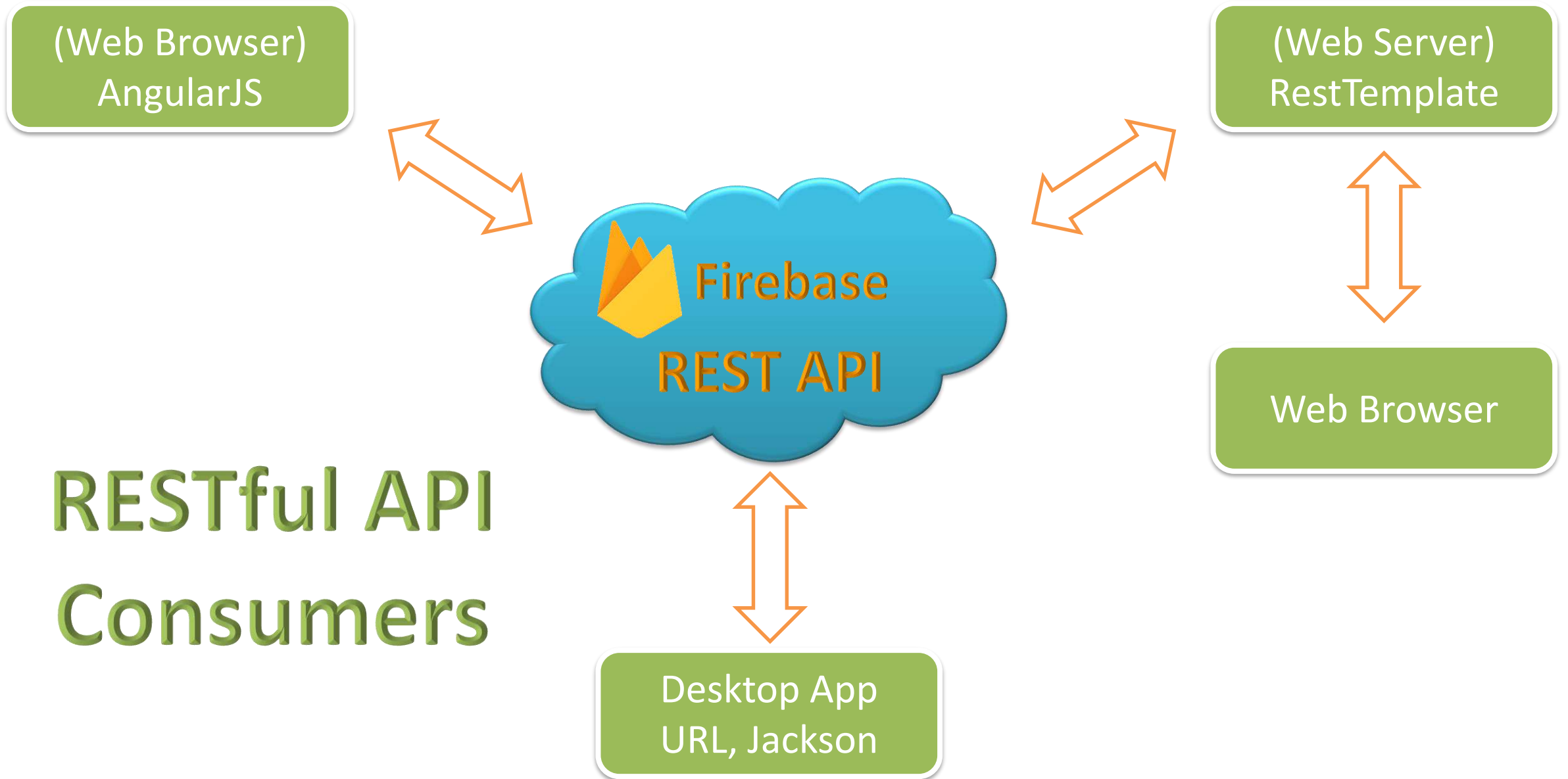


❑ Request:

- ❖ Method: GET, POST, PUT, DELETE
- ❖ URL: EntryPoint
- ❖ JsonData: JSON

❑ Response:

- ❖ JsonData: JSON

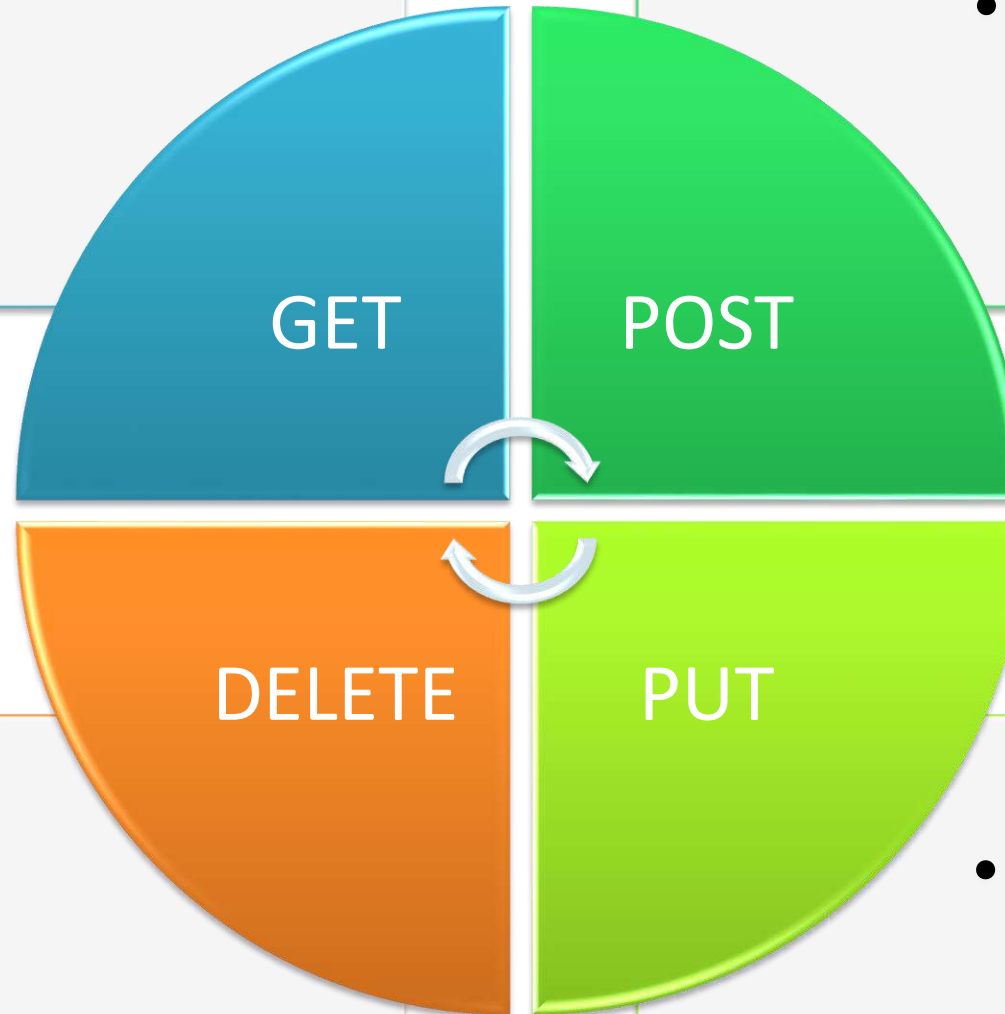




ANGULARJS \$HTTP SERVICE

- `get().then().catch()`

- `post().then().catch()`



- `delete().then().catch()`

- `put().then().catch()`

❑ REST API Operations

- ❖ `$http.get(url).then(response => {}).catch(error => {})`
- ❖ `$http.post(url, data).then(response => {}).catch(error => {})`
- ❖ `$http.put(url, data).then(response => {}).catch(error => {})`
- ❖ `$http.delete(url).then(response => {}).catch(error => {})`

❑ Ví dụ

```
$http.get("../users.json").then(response => {  
    var users = response.data;  
})  
.catch(error => {  
    console.log("Lỗi", error);  
})
```

```
$http.get(".../users.json")  
  .then(response => {  
    console.log(response.data);  
  })  
  .catch(error => {  
    console.log("Error", error);  
  });
```

GET

```
$http.delete(".../users/TeoNV.json")  
  .then(response => {  
    console.log(response.data);  
  })  
  .catch(error => {  
    console.log("Error", error);  
  });
```

DELETE

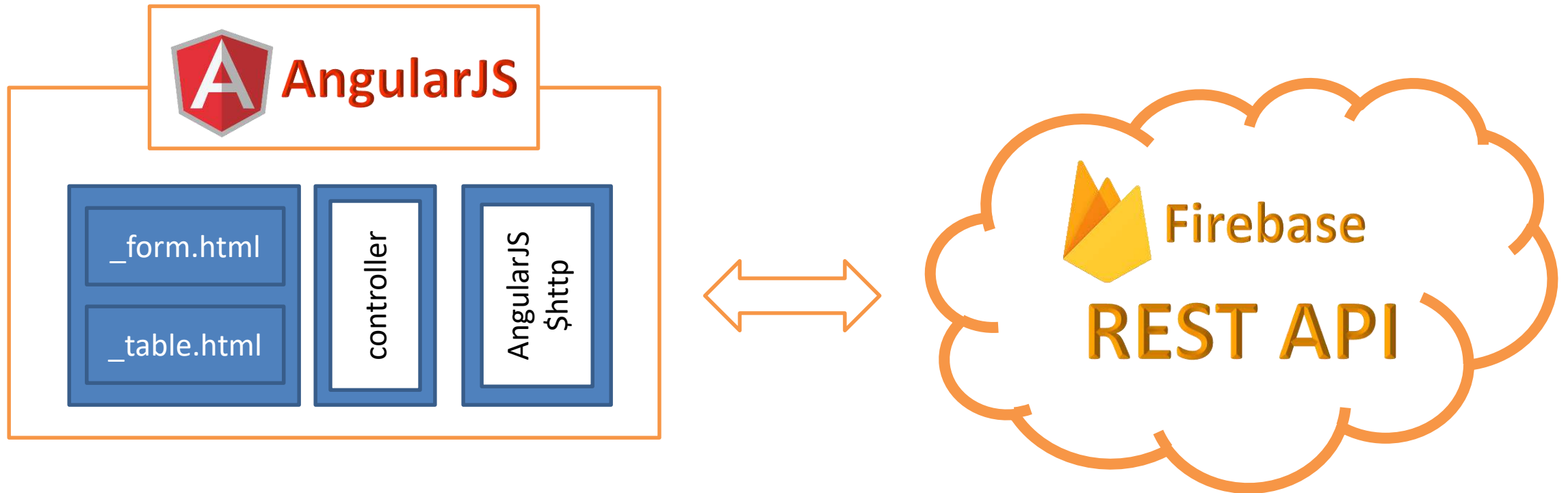
```
var data = {  
  id: "TeoNV",  
  name: "Nguyễn Văn Tèo"  
}  
$http.post(".../users.json", data)  
  .then(response => {  
    console.log(response.data);  
  })  
  .catch(error => {  
    console.log("Error", error);  
  });
```

POST

```
var data = {  
  id: "TeoNV",  
  name: "Nguyễn Văn Tèo"  
}  
$http.put(".../users/TeoNV.json", data)  
  .then(response => {  
    console.log(response.data);  
  })  
  .catch(error => {  
    console.log("Error", error);  
  });
```

PUT

ANGULARJS - DEMO APPLICATION MODEL

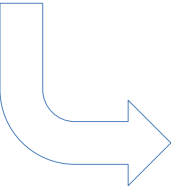




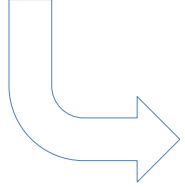



JAVA.NET.URL

```
String method = "GET" // POST, PUT, DELETE;  
// 1. REQUEST  
URL url = new URL("https://....json");  
HttpURLConnection conn = (HttpURLConnection) url.openConnection();  
conn.setRequestProperty("Accept", "application/json");  
conn.setRequestMethod(method);
```



```
// 1.1 DATA (POST & PUT only)  
if(method.equalsIgnoreCase("POST") || method.equalsIgnoreCase("PUT")) {  
    NodeObject data = ...;  
    conn.setDoOutput(true);  
    mapper.writeValue(conn.getOutputStream(), data);  
}
```



```
// 2. RESPONSE  
int responseCode = conn.getResponseCode();  
if (responseCode == 200) {  
    JsonNode response = mapper.readTree(conn.getInputStream());  
}  
conn.disconnect();
```

Jackson

❑ ObjectMapper

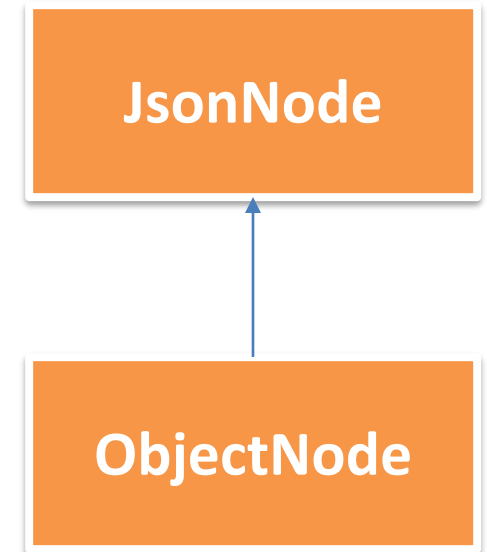
- ❖ readValue()
- ❖ readTree()
- ❖ writeValue()
- ❖ writeValueAsString()
- ❖ createObjectNode()

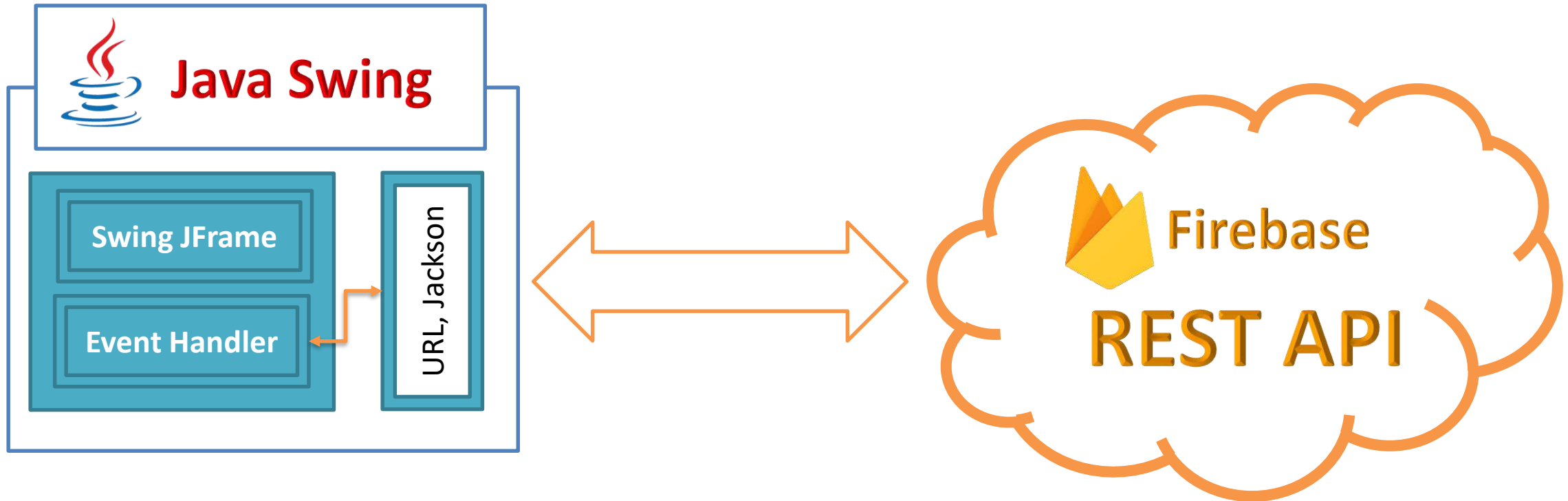
❑ JsonNode

- ❖ get()
- ❖ findValue()
- ❖ asType()
- ❖ iterator()

❑ ObjectNode

- ❖ put()
- ❖ putObject()
- ❖ putArray()







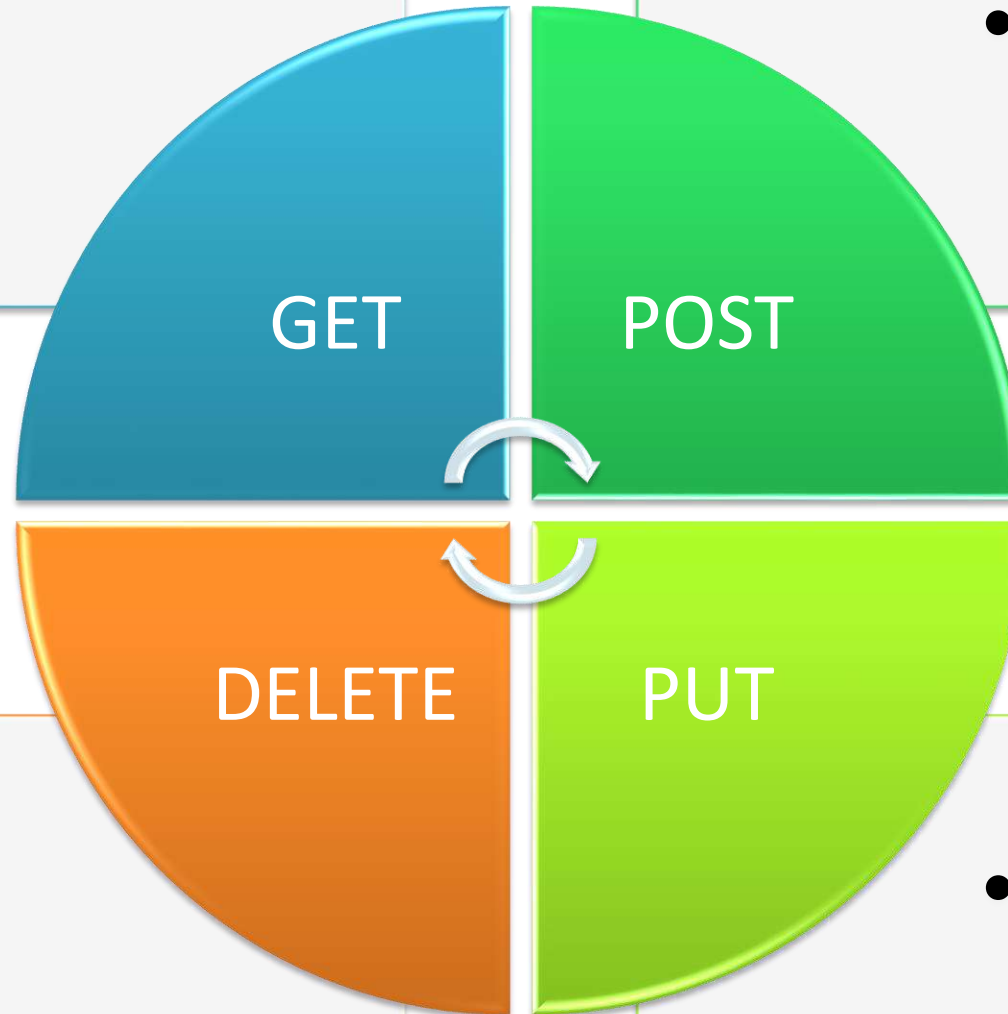
DEMO



SPRING RESTTEMPLATE

- getForObject()

- postForObject()



- delete()

- put()

❑ RestTemplate: Thực hiện các REST Operations

❖ **getForObject**(url, responseType<**T**>): **T**

❖ **delete**(url)

❖ **postForObject**(url, httpEntity, responseType<**T**>): **T**

❖ **put**(url, httpEntity)

❑ HttpEntity<T>: Đóng gói dữ liệu json gửi đến REST API

❖ new **HttpEntity**<>(**T**)

```
RestTemplate rest = new RestTemplate();
```

```
// GET
```

```
UserMap users = rest.getForObject("https://.../users.json", UserMap.class);  
User user = rest.getForObject("https://.../users/key.json", User.class);
```

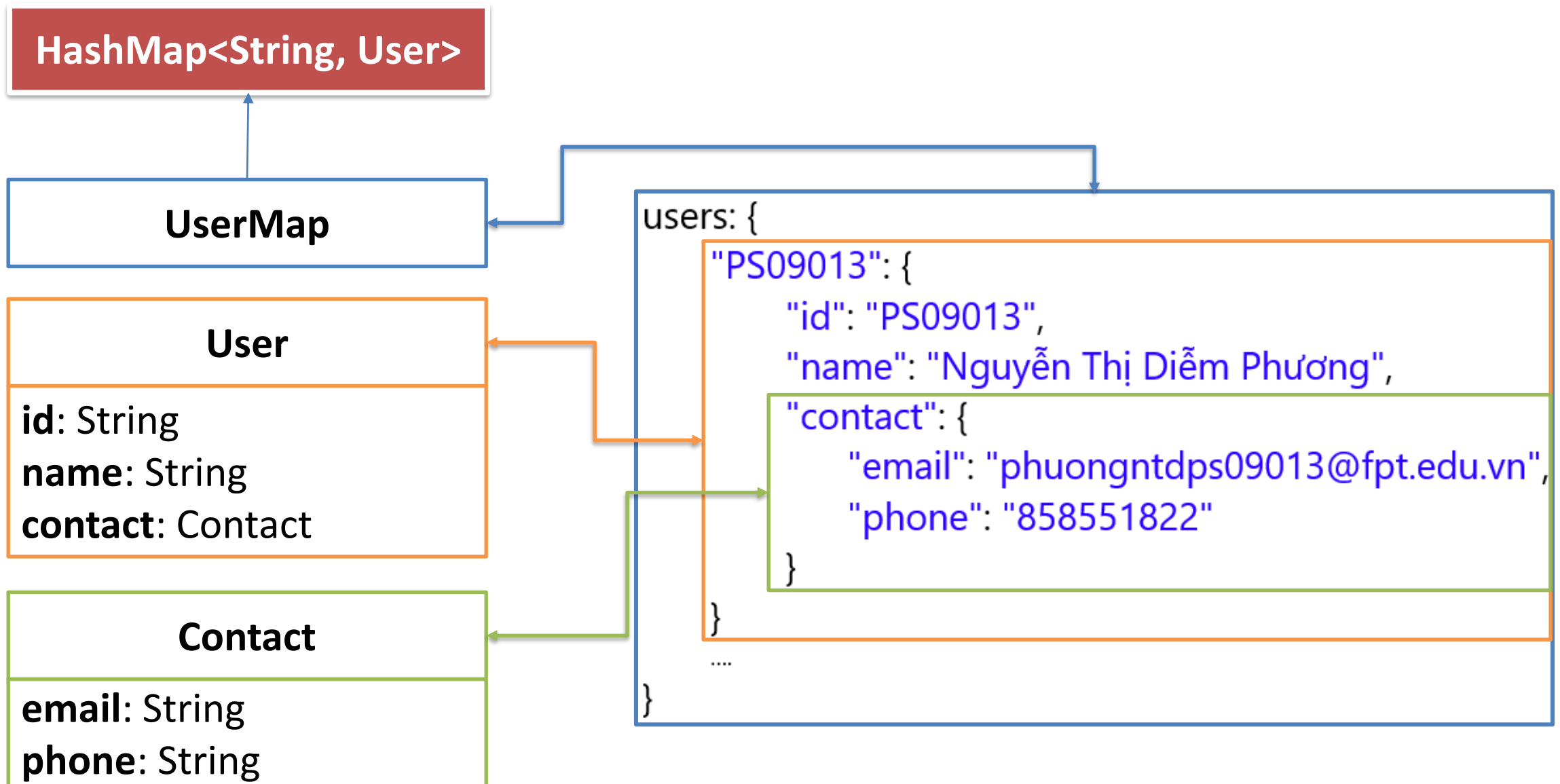
```
// DELETE
```

```
rest.delete("https://.../users/key.json");
```

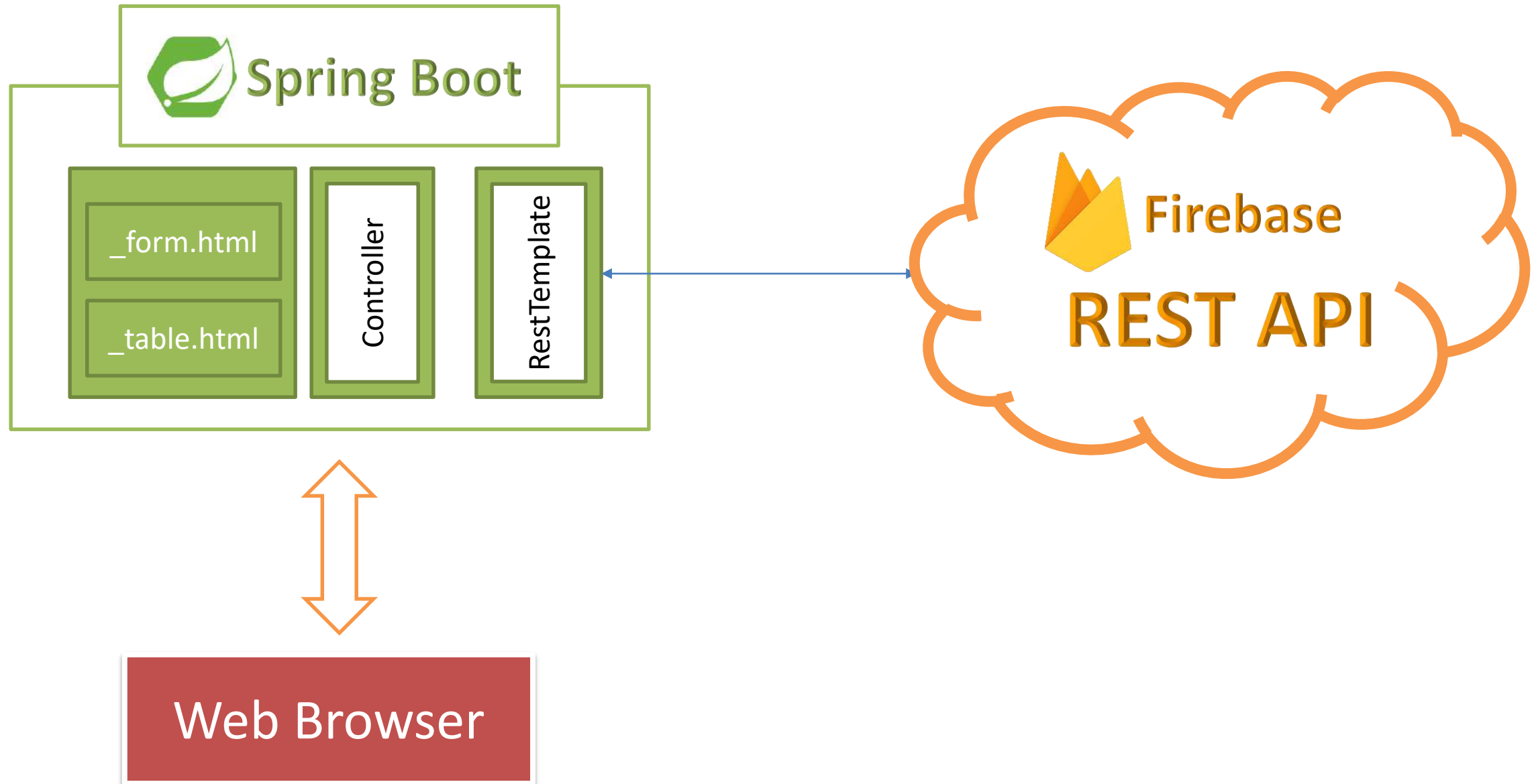
```
// POST & PUT
```

```
User data = ...;
```

```
HttpEntity<User> request = new HttpEntity<>(data);  
String key = rest.postForObject("https://.../users.json", request, String.class);  
rest.put("https://.../users/key.json", request);
```



SPRING RESTTEMPLATE – DEMO APPLICATION MODEL





- ✓ Web Service
- ✓ REST & REST API
- ✓ Firebase REST API & Postman
- ✓ Consume REST API with
 - ✓ AngularJS
 - ✓ Java.net.URL
 - ✓ RestTemplate





FPT Education
FPT POLYTECHNIC
Thank you

