#### Lesson Learned:

- 1. Membuat sebuah Rest Controller yang mana berfungsi untuk membuat aplikasi yang kita kembangkan dapat mengirimkan data ke luar/ke aplikasi lain.
- 2. Menggunakan Postman untuk melihat response data-data yang didapatkan dari tiap request yang kita buat. Ada 4 tipe yang digunakan pada tutorial ini, yaitu **GET**, **POST**, **PUT**, **DELETE**.
- 3. Untuk salah satu contoh **GET** di tutorial (View a Flight) yaitu akan mengembalikan sebuah FlightModel yang dikeluarkan dalam bentuk JSON dengan parameter request flightNumber-nya.
- 4. Untuk salah satu contoh **POST** di tutorial (Add a Flight) yaitu akan mengembalikan FlightModel juga dalam bentuk JSON.
- 5. Untuk salah satu contoh **PUT** di tutorial (Update a Flight) yaitu akan meng*update* flight bersangkutan (sesuai ID yang dimasukkan), mengembalikan sebuah String yang menyatakan sukses atau tidaknya data terupdate.
- 6. Untuk salah satu contoh **DELETE** di tutorial (Delete a Flight) yaitu akan menghapus flight dan mengembalikan sebuah String yang memberitahu sukses atau tidaknya deletion yang dilakukan.
- 7. Mock Server dibuat untuk menjadi *dummy* dari aplikasi produser yang bisa diakses oleh *user* dari luar, response-response dapat juga dibuat melalui Examples.

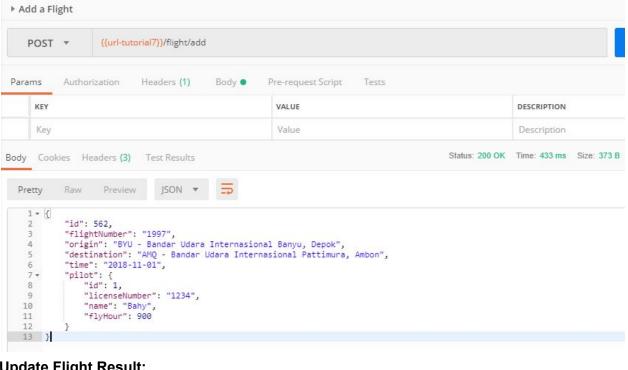
#### Latihan:

 Membuat 5 Webservice baru di Postman untuk melakukan CRUD dari objek Flight. Setiap output kurang lebih sama seperti Webservice yang dibuat pada Pilot. Webservice sendiri merupakansekumpulan fungsi program untuk melakukan pekerjaan tertentu yang dalam hal ini dapat melakukan manipulasi data, mengambil, menambahkan atau

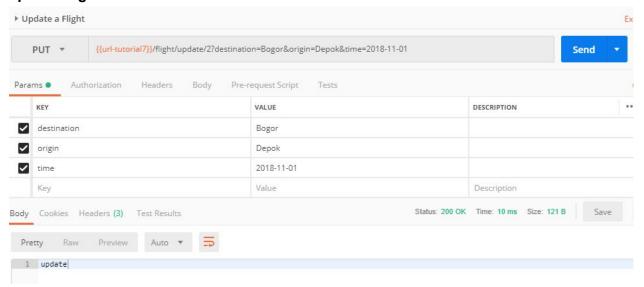
### menghapus.

```
@RestController
@RequestMapping("/flight")
public class FlightController {
    @Autowired
    private FlightService flightService;
    @PostMapping(value = "/add")
    public FlightModel addFlightSubmit(@RequestBody FlightModel flight) {
        return flightService.addFlight(flight);
    @PutMapping(value = "/update/{flightId}")
    public String updateFlightSubmit(@PathVariable("flightId") long flightId,
            @RequestParam("destination") String destination,
            @RequestParam("origin") String origin,
            @RequestParam("time") Date time) {
        FlightModel flight = flightService.getFlightDetailById(flightId);
        if (flight.equals(null)) {
            return "Couldn't find your flight";
        flight.setDestination(destination);
        flight.setOrigin(origin);
        flight.setTime(time);
        flightService.addFlight(flight);
        return "update";
    }
    @GetMapping(value = "/view/{flightNumber}")
    public FlightModel flightView(@PathVariable("flightNumber") String flightNumber) {
        FlightModel flight = flightService.getFlightDetailByFlightNumber(flightNumber).get();
        return flight;
    }
    @GetMapping(value = "/all")
    public List<FlightModel> flightViewAll() {
        return flightService.getAllFlight();
    }
 @DeleteMapping(value = "/{flightId}")
 public String deleteFlight(@PathVariable("flightId") long flightId) {
     FlightModel flight = flightService.getFlightDetailById(flightId);
     flightService.deleteFlight(flight);
     return "flight has been deleted";
 }
```

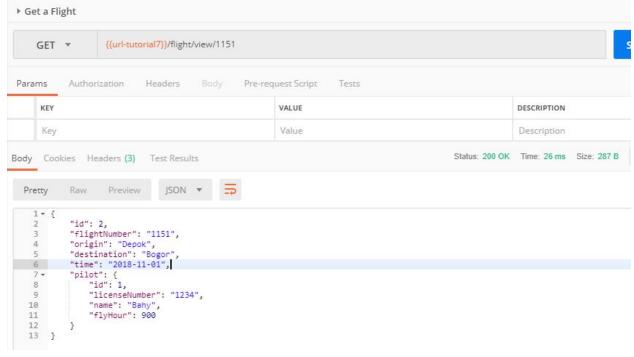
## **Add Flight Result:**



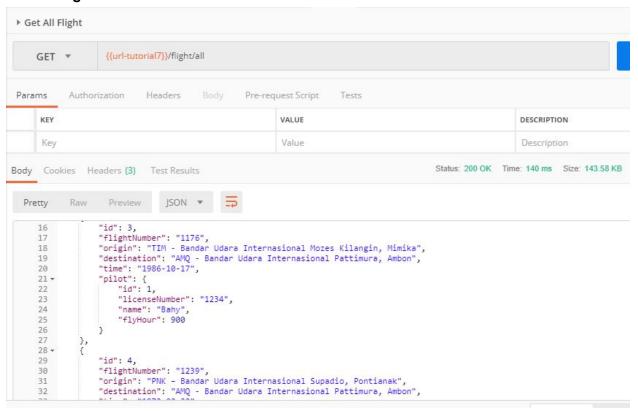
## **Update Flight Result:**



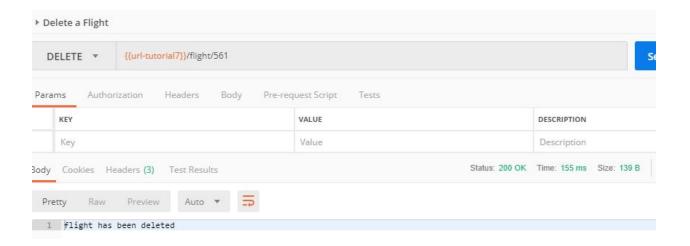
**Get Flight Result:** 



## **Get All Flight Result:**



**Delete Flight Result:** 



2. Menggunakan API dari Amadeus Travel Innovation Sandbox untuk mengambil data Airport yang ada di kota yang dimasukkan oleh user. Cara pengerjaannya hampir sama seperti Get Status pada latihan, yaitu mengambil data menggunakan Service Consumer dan mengembalikan/menunjukkan data sebagai Service Producer. Yang dilakukan adalah mengambil url + apikey dari api.sandbox.amadeus.com lalu simpan menjadi variable (airportUrl) di Setting.java, lalu buat sebuah Rest Controller lagi bernama AirportController. Lalu dibuat sebuah GetMapping yang mengambil parameter city untuk di pass ke API sebagai term. Sehingga nantinya API akan mengembalikan semua Airport yang terdapat pada City yang kita input, dalam bentuk JSON.

### URL API yang dimasukkan:

```
pliotControl... PilotControl... PilotServic... pilo
```

# Lakukan pemanggilan dan pengambilan data menggunakan API di Airport Controller:

```
application....
J PilotControl...
J PilotServic...
                                                                D PilotServic...
                                                                                   📔 update.html 🛮 📓 tutorial7/p...
                                                                                                                               ☐ PilotDetail.... ☐ Setting.java ☐ AirportCont... 🏻
   1 package com.apap.tutorial6.controller;
   {\tt 3} {\scriptsize \scriptsize \textcircled{\#} import org.springframework.beans.factory.annotation.Autowired;} \underline{\ }
 13 @RestController
     @RequestMapping("/airport")
 15 public class AirportController {
           RestTemplate restTemplate;
 190
          public RestTemplate restAirport() {
 21
                return new RestTemplate();
 22
           @GetMapping(value = "/{city}")
           public String getAirport(@PathVariable("city") String city) throws Exception {
   String path = Setting.airportUrl + city + "&country=ID&all_airports=true";
                return restTemplate.getForEntity(path, String.class).getBody();
```

## Hasil pemanggilan GET melalui Postman (Kota: Makassar):

