

1. Membuat rest controller yang mana akan bisa membuat aplikasi bisa menjadi API yang bisa mengeluarkan data ke luar.

Pada PilotController

```
@RestController
@RequestMapping("/pilot")
public class PilotController {
    @Autowired
    private PilotService pilotService;

    @PostMapping(value = "/add")
    private PilotModel addPilotSubmit(@RequestBody PilotModel pilot) {
        return pilotService.addPilot(pilot);
    }

    @GetMapping(value = "/view/{licenseNumber}")
    public PilotModel pilotView(@PathVariable("licenseNumber") String licenseNumber) {
        PilotModel pilot = pilotService.getPilotDetailByLicenseNumber(licenseNumber);
        return pilot;
    }

    @PutMapping(value = "/update/{pilotId}")
    private String updatePilotSubmit(@PathVariable("pilotId") long pilotId,
        @RequestParam("name") String name,
        @RequestParam("flyHour") int flyHour) {
        PilotModel pilot = pilotService.getPilotDetailById(pilotId);
        if (pilot.equals(null)) {
            return "Couldn't find your pilot";
        }

        pilot.setName(name);
        pilot.setFlyHour(flyHour);
        pilotService.updatePilot(pilotId, pilot);
        return "update";
    }

    @DeleteMapping(value = "/delete")
    private String deletePilot(@RequestParam("pilotId") long pilotId) {
        PilotModel pilot = pilotService.getPilotDetailById(pilotId);
        pilotService.deletePilot(pilot);
        return "success";
    }
}
```

- Fungsi addPilotSubmit akan mengembalikan PilotModel yang mana nantinya ketika dipanggil akan mengeluarkan data berbentuk JSON dari PilotModel terkait.
- Fungsi pilotView akan mengembalikan PilotModel yang membutuhkan licenseNumber untuk mencari pilot terkait dan nantinya ketika dipanggil akan mengeluarkan data berbentuk JSON.
- Fungsi updatePilotSubmit akan mengupdate pilot yang membutuhkan id pilot untuk mencari pilot terkait dan akan mengembalikan string yang menunjukkan bahwa data telah diupdate atau tidak menemukan pilot yang mau diupdate.
- Fungsi deletePilot akan mendelete pilot dan membutuhkan id pilot untuk mencari pilot terkait dan akan mengembalikan string yang menunjukkan bahwa pilot telah didelete.

Pada FlightController

```
@RestController
@RequestMapping(value="/flight")
public class FlightController {
    @Autowired
    private FlightService flightService;

    @PostMapping(value="/add")
    private FlightModel addFlightSubmit(@RequestBody FlightModel flight) {
        return flightService.addFlight(flight);
    }

    @GetMapping("/view/{flightNumber}")
    public FlightModel view(@PathVariable("flightNumber") String flightNumber) {
        FlightModel flight = flightService.getFlightDetailByFlightNumber(flightNumber);
        return flight;
    }

    @PutMapping(value = "/update/{flightId}")
    private String updateFlightSubmit(@PathVariable("flightId") long flightId,
        @RequestParam("destination") Optional<String> destination,
        @RequestParam("origin") Optional<String> origin,
        @RequestParam("time") Optional<String> time) {
        FlightModel flight = flightService.getFlightDetailById(flightId);
        if (flight.equals(null)) {
            return "Couldn't find your flight";
        }
        if (destination.isPresent()) {
            flight.setDestination(destination.get());
        }
        if (time.isPresent()) {
            flight.setTime(time.get());
        }
        if (origin.isPresent()) {
            flight.setOrigin(origin.get());
        }
        flightService.updateFlight(flight);
        return "Flight Update Success";
    }

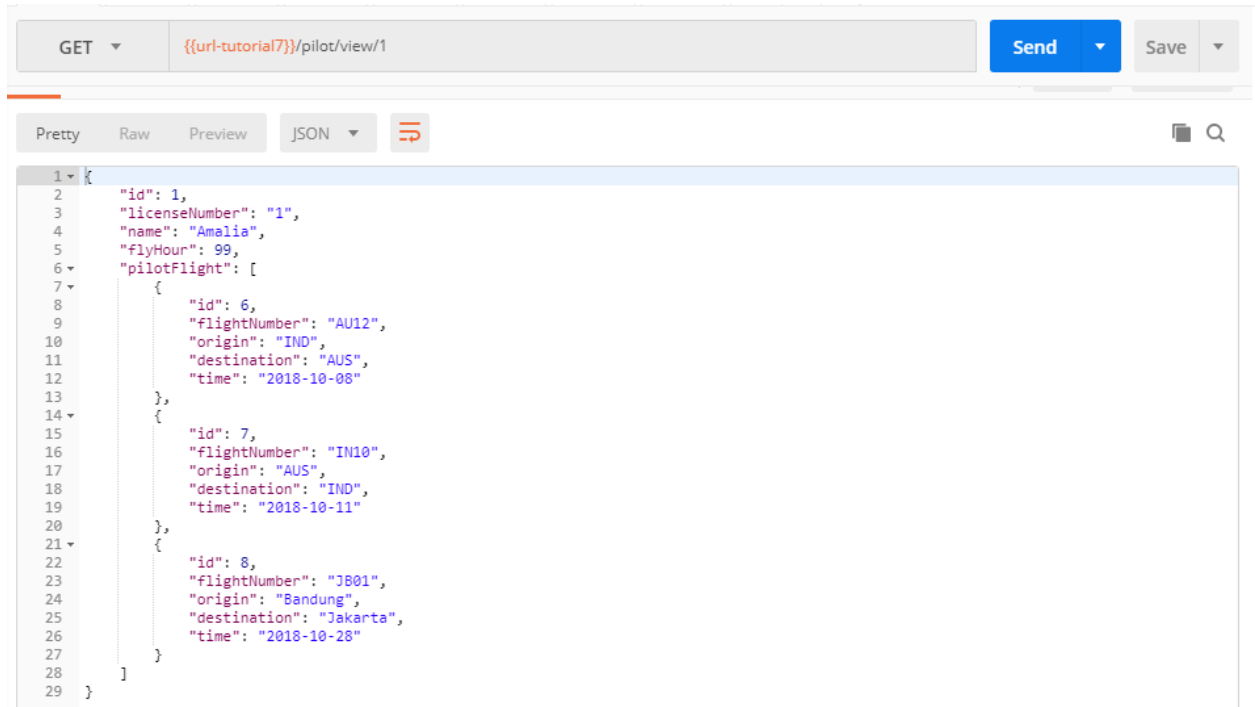
    @DeleteMapping(value = "/delete/{flightId}")
    private String deleteFlight(@PathVariable("flightId") long flightId) {
        flightService.deleteFlight(flightId);
        return "Flight has been deleted";
    }

    @GetMapping("/all")
    public List<FlightModel> retrieveListFlight() {
        return flightService.getAllFlight();
    }
}
```

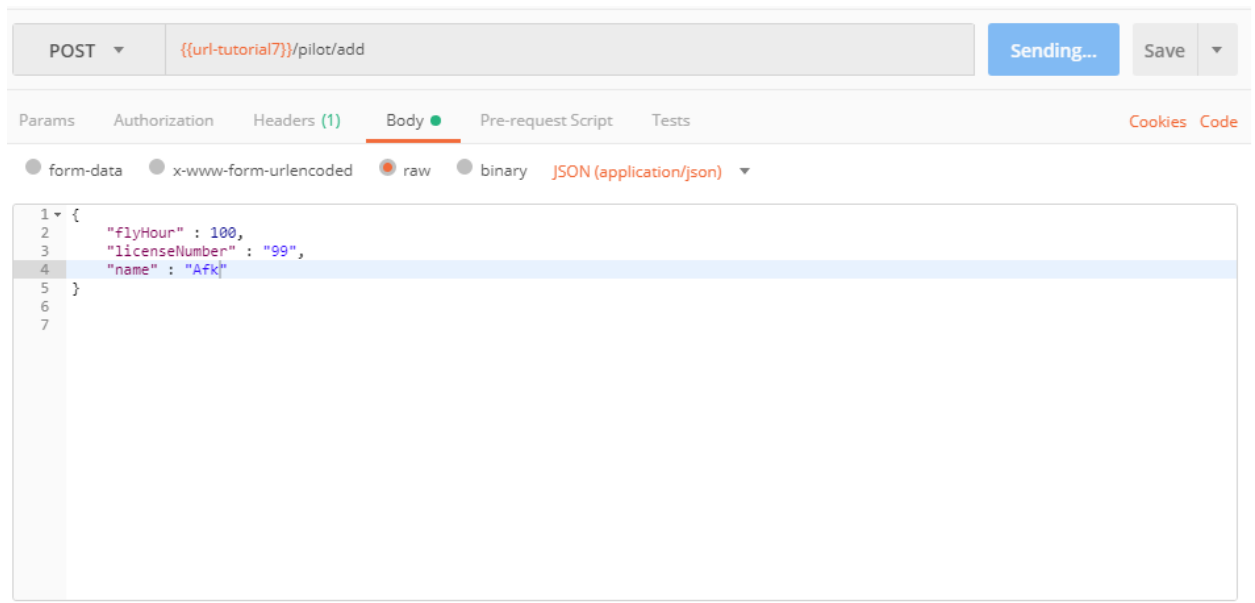
- Fungsi addFlightSubmit akan mengembalikan FlightModel yang mana nantinya ketika dipanggil akan mengeluarkan data berbentuk JSON dari FlightModel terkait.
- Fungsi view akan mengembalikan FlightModel yang membutuhkan flightNumber untuk mencari flight terkait dan nantinya ketika dipanggil akan mengeluarkan data berbentuk JSON dari FlightModel terkait.
- Fungsi updateFlightSubmit akan mengupdate flight yang membutuhkan id flight untuk mencari flight terkait dan akan mengembalikan string yang menunjukkan bahwa data telah diupdate atau tidak menemukan flight yang mau diupdate.
- Fungsi deleteFlight akan mendelete flight dan membutuhkan id flight untuk mencari flight terkait dan akan mengembalikan string yang menunjukkan bahwa flight telah didelete.
- Fungsi retrieveListFlight akan mengembalikan List dari semua FlightModel yang ada, dan dikeluarkan dalam bentuk JSON.

2. Menggunakan aplikasi Postman untuk melihat data-data yang didapatkan dari request-request yang dibuat.

- Mendapatkan pilot



- Menambah pilot baru



Output:

```
1 {  
2   "id": 9,  
3   "licenseNumber": "99",  
4   "name": "Afk",  
5   "flyHour": 100,  
6   "pilotFlight": null  
7 }
```

- Mengupdate pilot

GET `{{url-tutorial7}}/pilot/view/1` Send Save

Params Authorization Headers Body Pre-request Script Tests Cookies Code

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (3) Test Results Status: 200 OK Time: 35 ms Size: 469 B Save Download

Pretty Raw Preview JSON 🔍

```
1 {  
2   "id": 1,  
3   "licenseNumber": "1",  
4   "name": "Amalia",  
5   "flyHour": 99,  
6   "pilotFlight": [  
7     {
```

Updated pilot:

GET `{{url-tutorial7}}/pilot/view/1` Send Save

Params Authorization Headers Body Pre-request Script Tests Cookies Code

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (3) Test Results Status: 200 OK Time: 89 ms Size: 471 B Save Download

Pretty Raw Preview JSON 🔍

```
1 {  
2   "id": 1,  
3   "licenseNumber": "1",  
4   "name": "New Amel",  
5   "flyHour": 99,  
6   "pilotFlight": [  
7     {
```

- Mendelete pilot

DELETE `{{url-tutorial7}}/pilot/delete?pilotId=9` Send Save

Params Authorization Headers Body Pre-request Script Tests Cookies Code

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	pilotId	9			
	Key	Value	Description		

Body Cookies Headers (3) Test Results Status: 200 OK Time: 325 ms Size: 122 B Save Download

Pretty Raw Preview Auto ≡ 📄 🔍

```
1 success
```

- Menambahkan flight

POST `{{url-tutorial7}}/flight/add` Send Save

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Code

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json)

```
1 {
2   "flightNumber": "TST1",
3   "origin": "TEST",
4   "destination": "TSET",
5   "time": "2018-10-10",
6   "pilot": {
7     "id": 8,
8     "licenseNumber": "69",
9     "name": "pe1",
10    "flyHour": 69
11  }
12 }
13
```

Output:

Pretty Raw Preview JSON ≡ 📄 🔍

```
1 {
2   "id": 14,
3   "flightNumber": "TST1",
4   "origin": "TEST",
5   "destination": "TSET",
6   "time": "2018-10-10"
7 }
```

- Mendapatkan flight

The screenshot shows a REST client interface with a GET request to `{{url-tutorial7}}/flight/view/LA02`. The response status is 200 OK, with a time of 117 ms and a size of 232 B. The response body is displayed in JSON format:

```
1 {
2   "id": 13,
3   "flightNumber": "LA02",
4   "origin": "AAAA",
5   "destination": "ASDWHAWBVFCIKAW H",
6   "time": "2018-10-11"
7 }
```

- Mendelete flight

The screenshot shows a REST client interface with a DELETE request to `{{url-tutorial7}}/flight/delete/14`. The response status is 200 OK, with a time of 324 ms and a size of 139 B. The response body is displayed in text format:

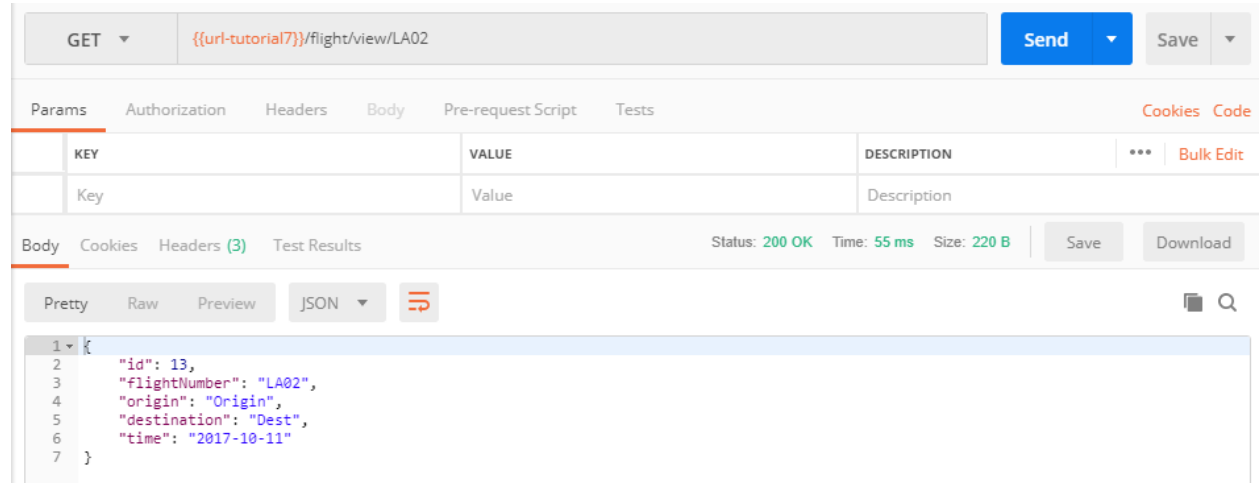
```
1 Flight has been deleted
```

- Mengupdate flight

The screenshot shows a REST client interface with a PUT request to `{{url-tutorial7}}/flight/update/13?destination=Dest&origin=Origin&time=2017-10-11`. The response status is 200 OK, with a time of 576 ms and a size of 137 B. The response body is displayed in text format:

```
1 Flight Update Success
```

Updated flight:




GET `{{url-tutorial7}}/flight/view/LA02` Send Save

Params Authorization Headers Body Pre-request Script Tests Cookies Code

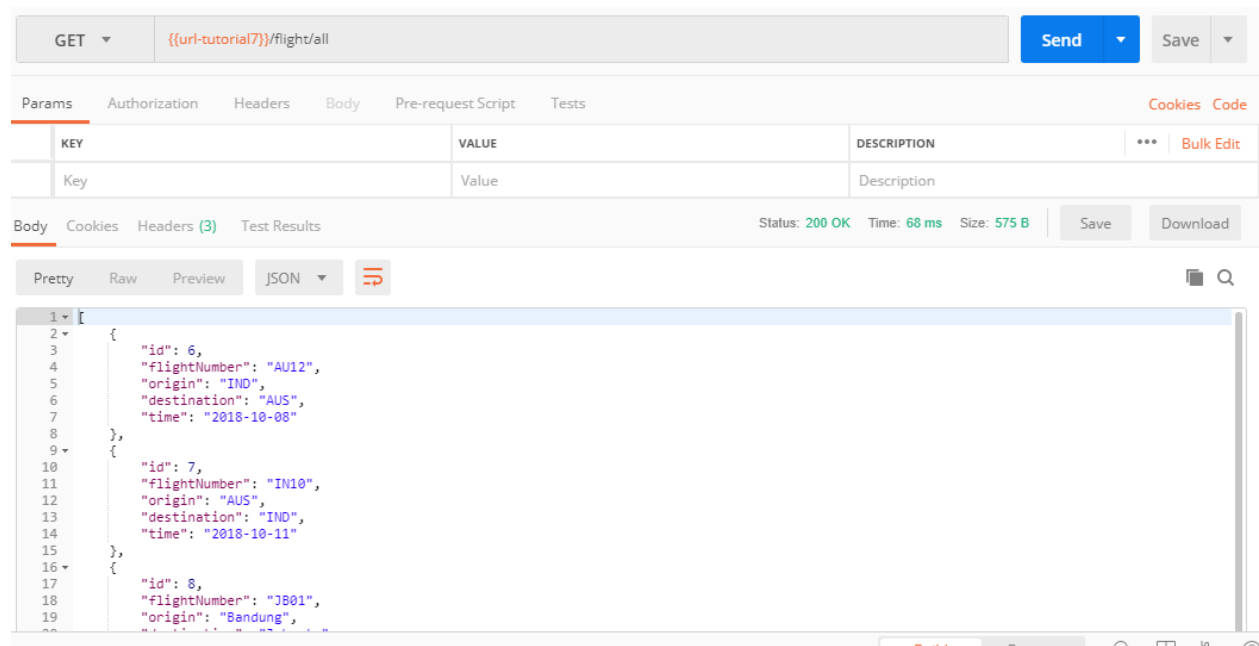
KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (3) Test Results Status: 200 OK Time: 55 ms Size: 220 B Save Download

Pretty Raw Preview JSON 

```
1 {
2   "id": 13,
3   "flightNumber": "LA02",
4   "origin": "Origin",
5   "destination": "Dest",
6   "time": "2017-10-11"
7 }
```

- Lihat semua flight




GET `{{url-tutorial7}}/flight/all` Send Save

Params Authorization Headers Body Pre-request Script Tests Cookies Code

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (3) Test Results Status: 200 OK Time: 68 ms Size: 575 B Save Download

Pretty Raw Preview JSON 

```
1 [
2   {
3     "id": 6,
4     "flightNumber": "AU12",
5     "origin": "IND",
6     "destination": "AUS",
7     "time": "2018-10-08"
8   },
9   {
10    "id": 7,
11    "flightNumber": "IN10",
12    "origin": "AUS",
13    "destination": "IND",
14    "time": "2018-10-11"
15  },
16  {
17    "id": 8,
18    "flightNumber": "JB01",
19    "origin": "Bandung",
20    "time": "2018-10-11"
21  }
22 ]
```

Write-up Tutorial 7

- Membuat Mock Server yang akan menjadi dummy aplikasi yang akan menjadi producer yang bisa diakses oleh orang luar, dengan membuat response-response dari request yang bisa dibuat oleh consumer nantinya.

- Request get status

contoh request:

GET `{{pilot}}/pilot?licenseNumber=1111` Send Save

Params Authorization Headers Body Pre-request Script Tests Cookies Code

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	licenseNumber	1111			
	Key	Value	Description		

Contoh response:

GET `{{pilot}}/pilot?licenseNumber=1111`

Params Headers Body

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	licenseNumber	1111			
	Key	Value	Description		

EXAMPLE RESPONSE

Body Headers (1) Status Enter Response Code

Pretty Raw Preview Text ≡

```

1 {
2   "status" : "inactive"
3 }
```

EXAMPLE RESPONSE

Body Headers (1) Status Enter Response Code

	KEY	VALUE
	Content-Type	application/json; charset=UTF8
	Key	Value

- Request get full status

Contoh request:

POST `{{pilot}}/pilot` Send Save

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Code

☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary JSON (application/json)

```

1 {
2   "flyHour": 600,
3   "licenseNumber": "777",
4   "name": "dia"
5 }
```


Contoh response:

EXAMPLE REQUEST

POST `{{pilot}}/pilot`

Params Headers (1) Body

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

EXAMPLE RESPONSE

Body Headers (1) Status Enter Response Code

Pretty Raw Preview JSON

```

1 {
2   "status": "inactive",
3   "valid-until": "2025-20-30"
4 }

```

EXAMPLE RESPONSE

Body Headers (1) Status Enter Response Code

KEY	VALUE
Content-Type	application/json;charset=UTF8
Key	Value

Lalu akan mendapatkan link untuk mengakses mock server tersebut, kemudian dimasukkan ke dalam url yang diakses dari postman.

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	url-tutorial7		localhost:2016/			
<input checked="" type="checkbox"/>	pilot		https://3c6c0308-24de-48a8-b811-c50db1e547d8.mock....			

Setelah itu, akan bisa diakses mock servernya

GET `{{pilot}}/pilot?licenseNumber=1111` Send Save

Params Authorization Headers Body Pre-request Script Tests Cookies Code

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/> licenseNumber	1111			
Key	Value	Description		

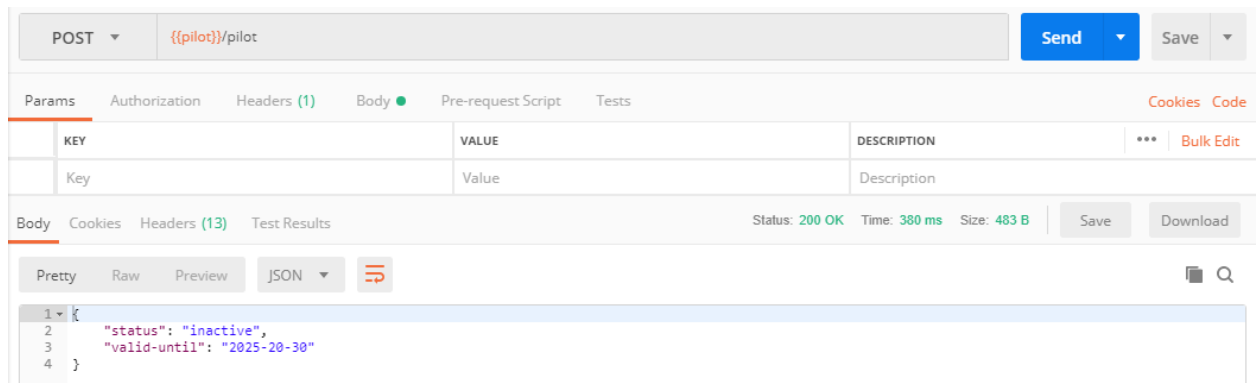
Body Cookies Headers (13) Test Results Status: 200 OK Time: 2786 ms Size: 457 B Save Download

Pretty Raw Preview JSON

```

1 {
2   "status": "inactive"
3 }

```



Lalu, pada controller kita dicoba untuk mengambil data dari API yang sebelumnya dibuat

```
@Autowired
RestTemplate restTemplate;

@Bean
public RestTemplate rest() {
    return new RestTemplate();
}

@GetMapping(value="/status/{licenseNumber}")
public String getStatus(@PathVariable("licenseNumber") String licenseNumber) throws Exception {
    String path = Setting.pilotUrl + "/pilot?licenseNumber=" + licenseNumber;
    return restTemplate.getForEntity(path, String.class).getBody();
}

@GetMapping(value="/full/{licenseNumber}")
public PilotDetail postStatus(@PathVariable("licenseNumber") String licenseNumber) throws Exception {
    String path = Setting.pilotUrl + "/pilot";
    PilotModel pilot = pilotService.getPilotDetailByLicenseNumber(licenseNumber);
    PilotDetail detail = restTemplate.postForObject(path, pilot, PilotDetail.class);
    return detail;
}
}
```

Membuat kelas setting yang menyimpan url dari mock server

```
public class Setting {
    final public static String pilotUrl = "https://3c6c0308-24de-48a8-b811-c50db1e547d8.mock.pstmn.io";
    final public static String Amadeus_Airport_API = "https://api.sandbox.amadeus.com/v1.2/airports/autocomplete?apikey=";
}
```

Yang mana akan data yang diterima dimasukkan ke dalam kelas PilotDetail, dapat dilihat pada line yang melakukan `postForObject` kepada kelas PilotDetail

```

@JsonIgnoreProperties(ignoreUnknown = true)
public class PilotDetail {
    private String status;

    @JsonProperty("valid-until")
    private Date validUntil;

    public void setStatus(String status) {
        this.status = status;
    }

    public void setValidUntil(Date validUntil) {
        this.validUntil = validUntil;
    }

    public String getStatus() {
        return status;
    }

    public Date getValidUntil() {
        return validUntil;
    }
}

```

Maka setelah itu, data akan diterima pada tutorial yang dibuat

GET ▼ {{url-tutorial7}}/pilot/status/1111 Send Save ▼

Params Authorization Headers Body Pre-request Script Tests Cookies Code

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (3) Test Results Status: 200 OK Time: 2766 ms Size: 142 B Save Download

Pretty Raw Preview Auto ≡

```

1 {
2   "status": "inactive"
3 }

```

GET ▼ {{url-tutorial7}}/pilot/full/1111 Send Save ▼

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Code

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (3) Test Results Status: 200 OK Time: 6932 ms Size: 196 B Save Download

Pretty Raw Preview JSON ≡

```

1 {
2   "status": "inactive",
3   "valid-until": "2026-08-30T00:00:00.000+0000"
4 }

```

4. Menjadi producer dengan mengambil data dari API Amadeus

```
public class Setting {
    final public static String pilotUrl = "https://3c6c0308-24de-48a8-b811-c50db1e547d8.mock.pstmn.io";
    final public static String Amadeus_Airport_API = "https://api.sandbox.amadeus.com/v1.2/airports/autocomplete?apikey=";
}
```

Data kemudian akan dioper keluar berdasarkan parameter kota-nya yang akan diambil dari negara Indonesia yang akan dibuat pada controller baru

```
@RestController
@RequestMapping(value="/airport")
public class AirportController {

    @Autowired
    RestTemplate restTemplate;

    @Bean
    public RestTemplate rest2() {
        return new RestTemplate();
    }

    @GetMapping("/city/{city}")
    public String getAirport(@PathVariable("city") String city) throws Exception {
        String path = Setting.Amadeus_Airport_API + "&term=" + city + "&country=ID&all_airports=true";
        return restTemplate.getForEntity(path, String.class).getBody();
    }
}
```

Contoh pengambilan data:

The screenshot shows a REST client interface with a GET request to the URL `{{url-tutorial7}}/airport/city/Jakarta`. The response status is 200 OK, with a time of 4384 ms and a size of 400 B. The response body is displayed in a JSON format, showing a list of airports for Jakarta.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
[
  {
    "value": "JKT",
    "label": "Jakarta [JKT]"
  },
  {
    "value": "CGK",
    "label": "Jakarta - Soekarno - Hatta International Airport [CGK]"
  },
  {
    "value": "HLP",
    "label": "Jakarta - Halim Perdanakusuma Airport [HLP]"
  }
]
```