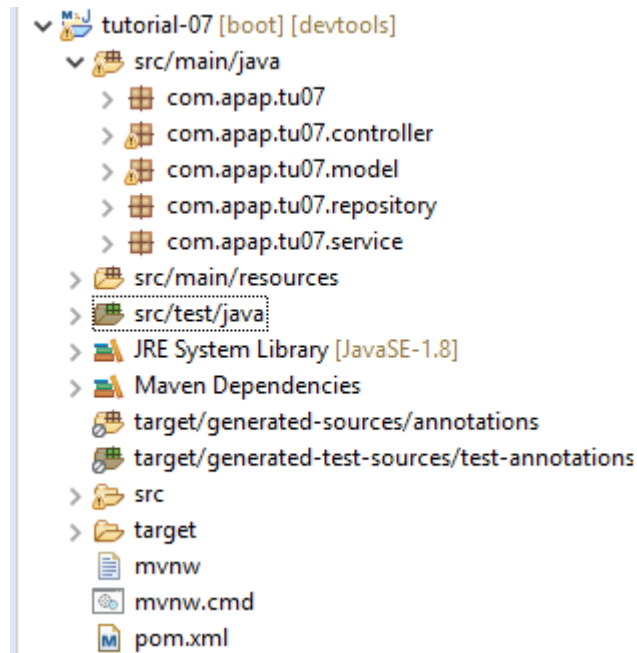
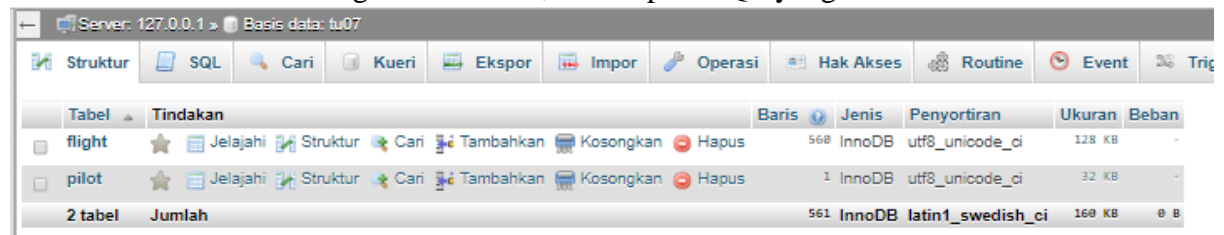


1. Pendahuluan

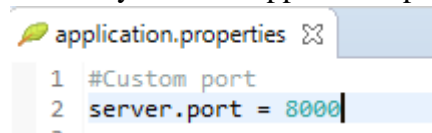
- Mengubah nama package menjadi *07. klik kanan package >> Refactor >> Rename



- Membuat database dengan nama tu07, lalu import SQL yang ada di Scele.



- Menyesuaikan application.properties sesuai dengan environment



2. Membuat RestController

- Mengubah pilot controller.

```
package com.apap.tu06.controller;

import com.apap.tu06.model.PilotModel;
import com.apap.tu06.service.PilotService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

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

    @PostMapping(value = "/add")
    public 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).get();
        return pilot;
    }

    @DeleteMapping(value = "/delete")
    public String deletePilot(@RequestParam("pilotId") long pilotId) {
        PilotModel pilot = pilotService.getPilotDetailById(pilotId).get();
        pilotService.deletePilot(pilot); // Buat implementasi method ini
        return "success";
    }

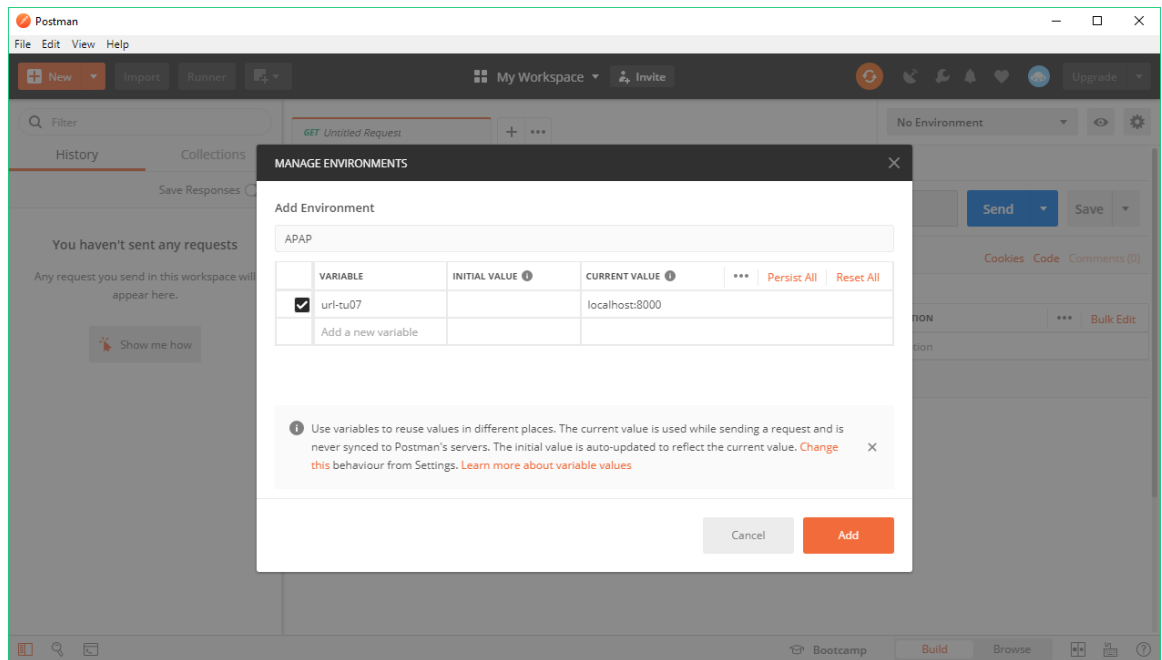
    @PutMapping(value = "/update/{pilotId}")
    public String updatePilotSubmit(@PathVariable("pilotId") long pilotId,
        @RequestParam("name") String name,
        @RequestParam("flyHour") int flyHour) {
        PilotModel pilot = pilotService.getPilotDetailById(pilotId).get();
        if (pilot.equals(null))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

        pilot.setName(name);
        pilot.setFlyHour(flyHour);
        pilotService.updatePilot(pilotId, pilot); // Buat implementasi method ini
        return "update";
    }
}
```

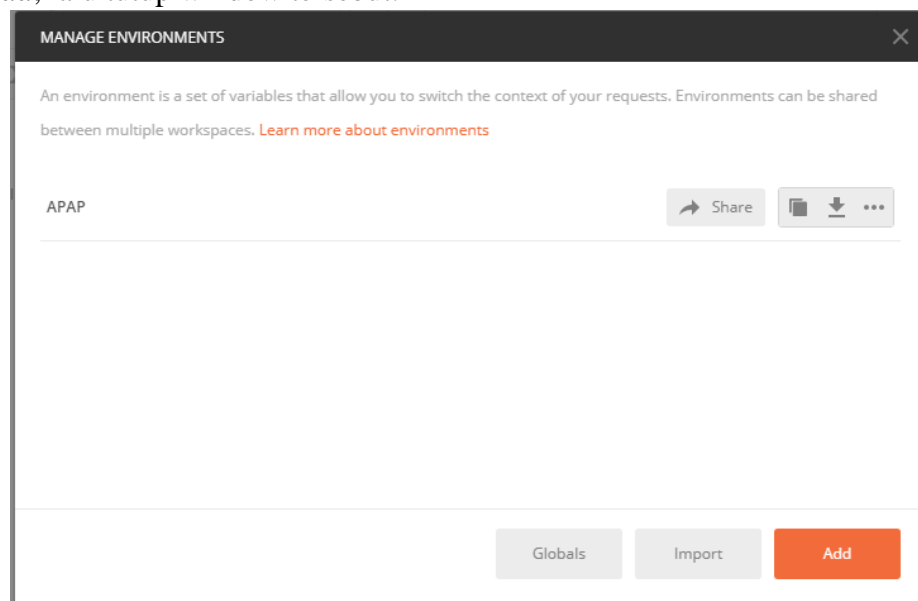
3. Mengakses method GET

- a. Membuka Postman, klik New >> Environment. Lalu, membuat *environment* baru bernama “APAP” dan membuat variable sesuai dengan spesifikasi.

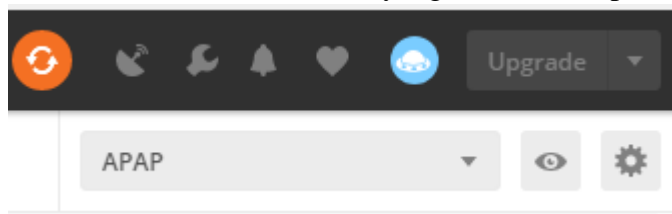
Sary Yuni A
1706106980



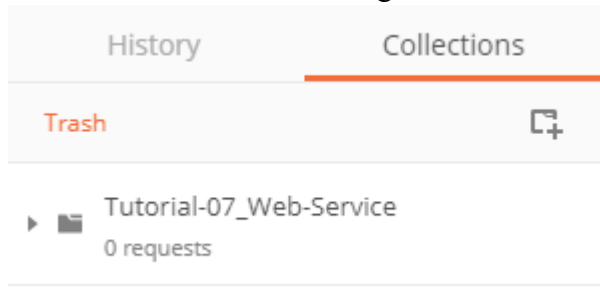
Klik *Add*, lalu tutup window tersebut.



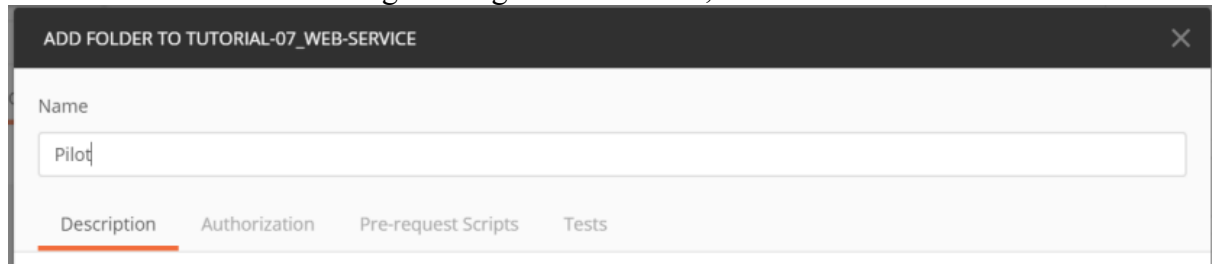
b. Pilih *environment* APAP yang telah dibuat pada bagian kanan atas Postman.



- c. Buat Collection baru dengan nama “Tutorial-07_Web-Service”.



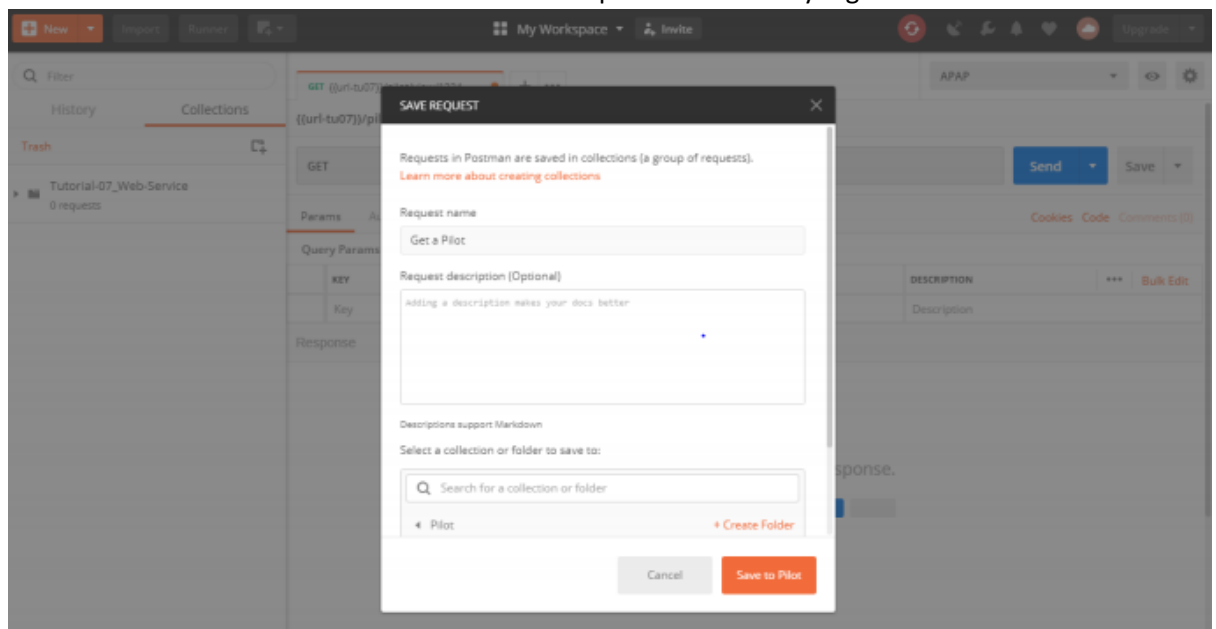
- d. Membuat folder baru dengan mengakses menu dot, beri nama “Pilot”.



- e. Membuat request baru dengan method GET dengan url “{{urltu07}}/pilot/view/1234”, bilangan 1234 merupakan license number pada DB Anda.

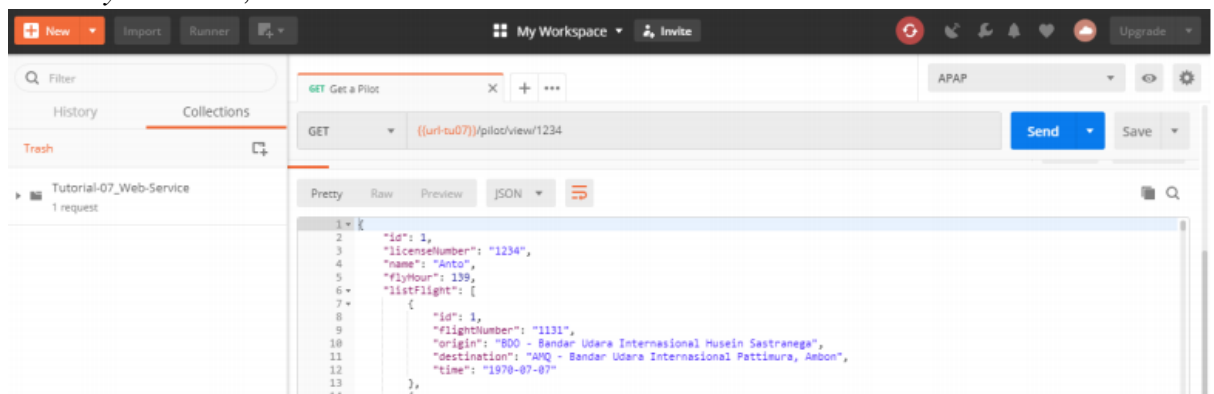


- f. Klik Save >> beri nama “Get a Pilot” dan Save pada folder Pilot yang sudah dibuat.

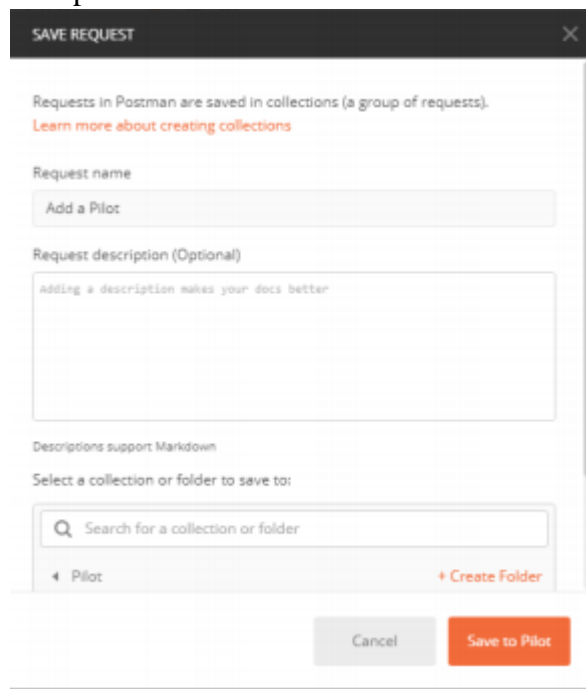


Sary Yuni A
1706106980

- g. Mengirim request dengan menekan tombol “Send”. Akan ada pesan pada *response body* di bawah, sbb :

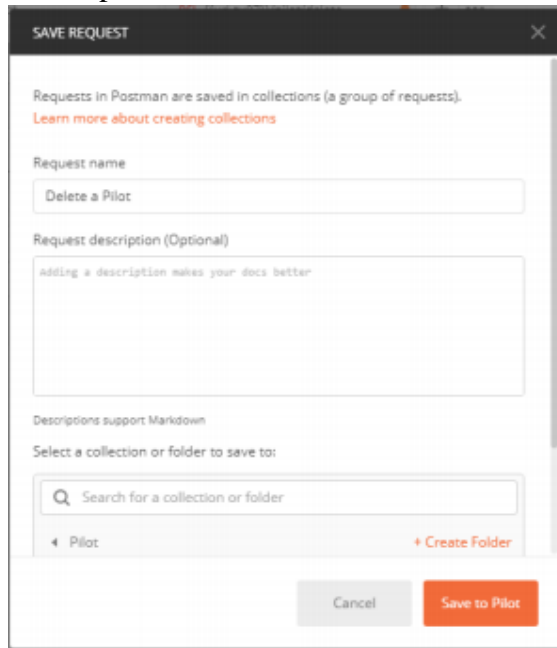


- h. Buat *request* lainnya untuk melengkapi akses ke seluruh *web service* yang ada pada PilotController (empat *method*). Simpan seluruhnya pada folder Pilot.
- Add pilot



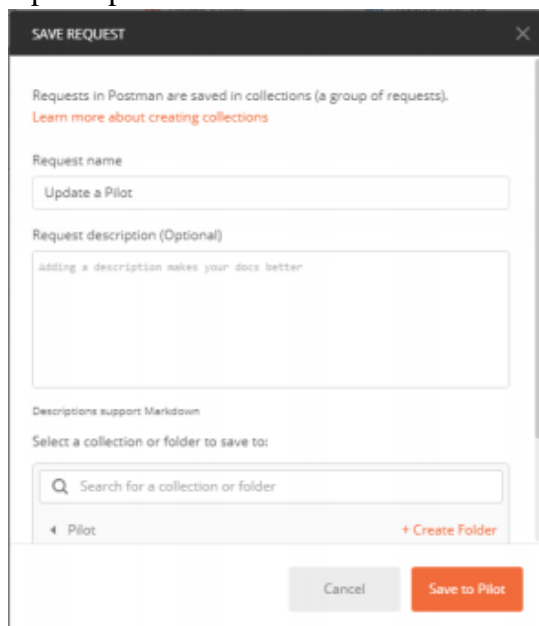
Sary Yuni A
1706106980

- Delete pilot



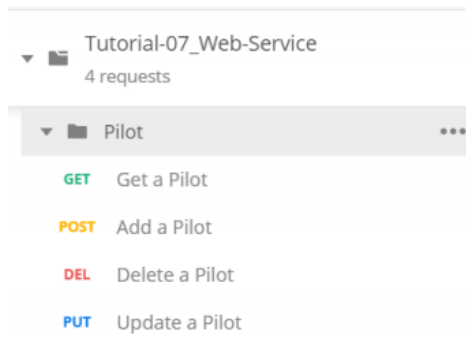
The image shows the 'SAVE REQUEST' dialog in Postman. At the top, it says 'Requests in Postman are saved in collections (a group of requests). Learn more about creating collections'. Below this, there is a 'Request name' field containing 'Delete a Pilot'. Underneath is a 'Request description (Optional)' text area with the placeholder text 'Adding a description makes your docs better'. Below the description, it says 'Descriptions support Markdown'. Then, there is a section 'Select a collection or folder to save to:' with a search bar containing 'Search for a collection or folder'. Below the search bar, there is a list item 'Pilot' with a folder icon to its left and a '+ Create Folder' link to its right. At the bottom right, there are two buttons: 'Cancel' and 'Save to Pilot'.

- Update pilot



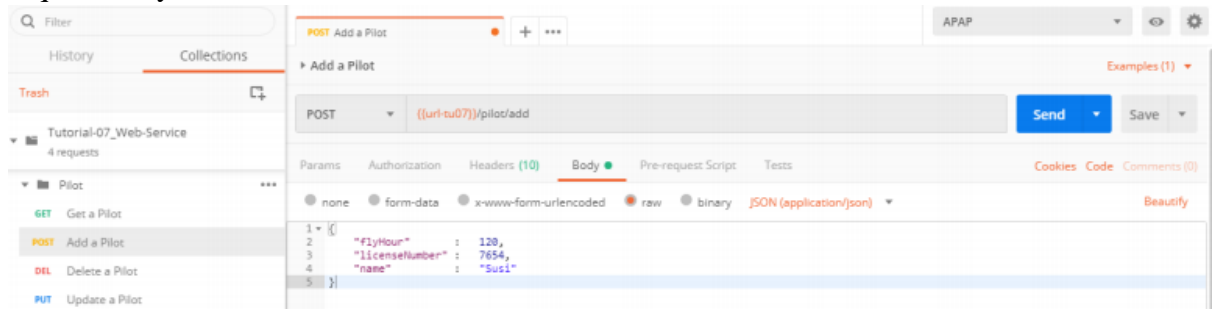
The image shows the 'SAVE REQUEST' dialog in Postman, similar to the previous one. The 'Request name' field now contains 'Update a Pilot'. The 'Request description (Optional)' text area still has the placeholder text 'adding a description makes your docs better'. The 'Select a collection or folder to save to:' section is identical, showing the 'Pilot' folder. The 'Cancel' and 'Save to Pilot' buttons are at the bottom right.

- Hasil

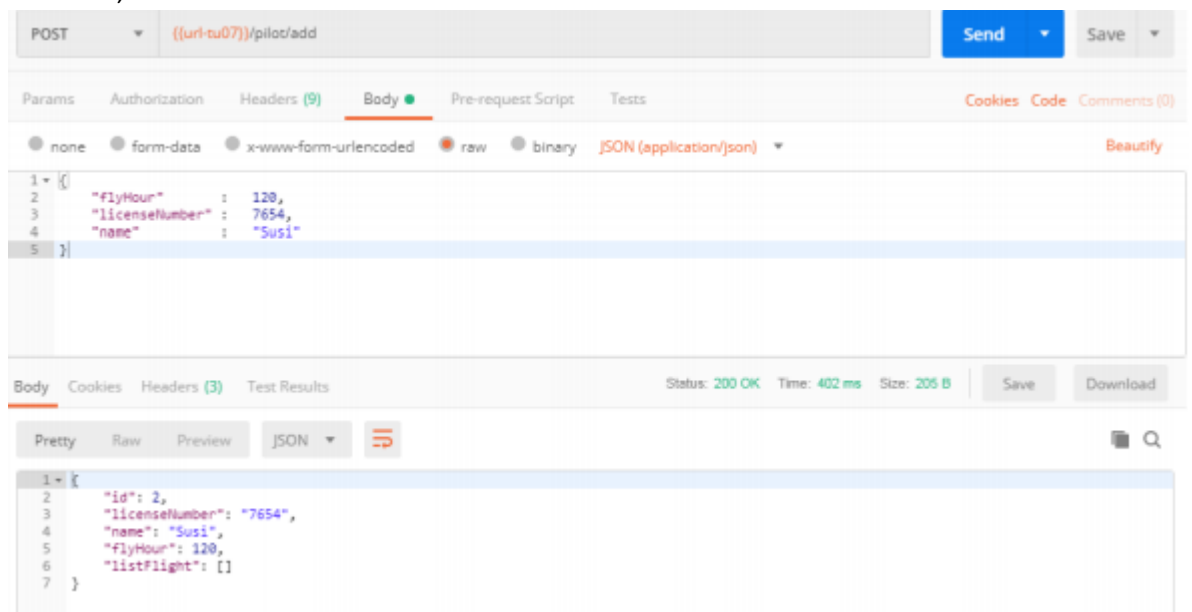


4. Mengakses method POST

- a. Mengakses request post yang telah dibuat untuk “Add a Pilot.” Lalu tambahkan request body.



- b. Mengirim *request* dengan menekan tombol “Send”. Akan ada pesan pada response body di bawah, sbb :



- c. Mengecek pada dB.

	Ubah	Salin	Hapus	2	120	7654	Susi
--	------	-------	-------	---	-----	------	------

- d. Menambahkan paramater pada Postman untuk *update* dan *delete*.

- *update*



- *delete*

Sary Yuni A
1706106980

Params ● Authorization Headers (8) Body Pre-request Script Tests Cookies Code Comments (0)				
Query Params				
	KEY	VALUE	DESCRIPTION	*** Bulk Edit
<input checked="" type="checkbox"/>	name	HasilUpdate		
<input checked="" type="checkbox"/>	flyHour	321		
	Key	Value	Description	

5. Mockserver

- Membuat Collection baru dengan nama “Tutorial-07_Mock-Server”.

CREATE A NEW COLLECTION

Name

Tutorial-07_Mock-Server

Description

Authorization

Pre-request Scripts

Tests

Variables

This description will show in your collection's documentation, along with the descriptions of its folders and requests.

Adding a description makes your docs better

Descriptions support Markdown

Cancel

Create

- Membuat request baru pada Collection tersebut dengan nama “Get Status” lalu Save.

SAVE REQUEST

Requests in Postman are saved in collections (a group of requests).
[Learn more about creating collections](#)

Request name

Get Status

Request description (Optional)

Adding a description makes your docs better

Descriptions support Markdown

Select a collection or folder to save to:

Q Search for a collection or folder

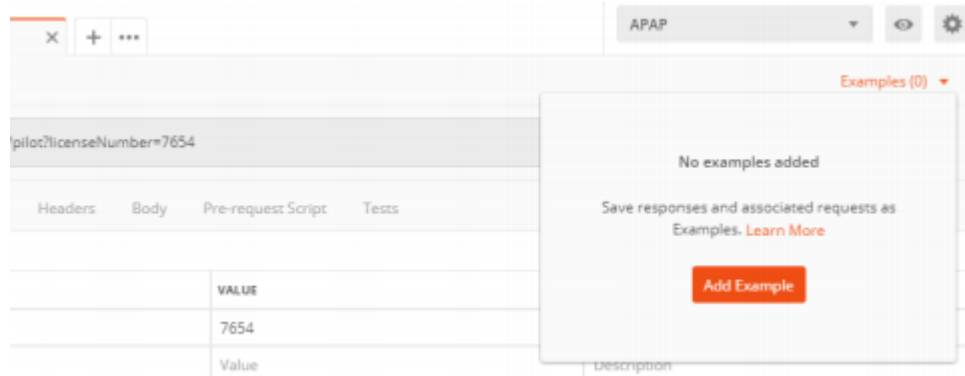
Tutorial-07_Mock-Server + Create Folder

Cancel

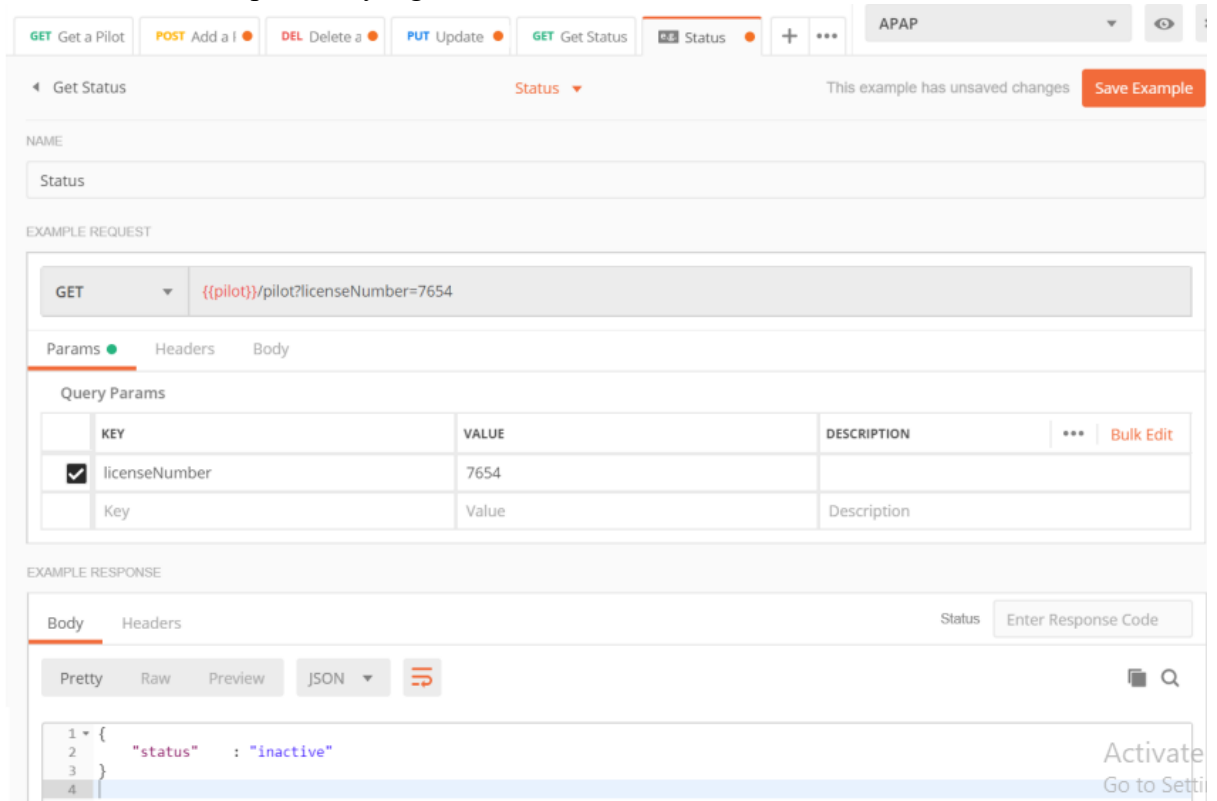
Save to Tutorial-07_Mock-Server

Sary Yuni A
1706106980

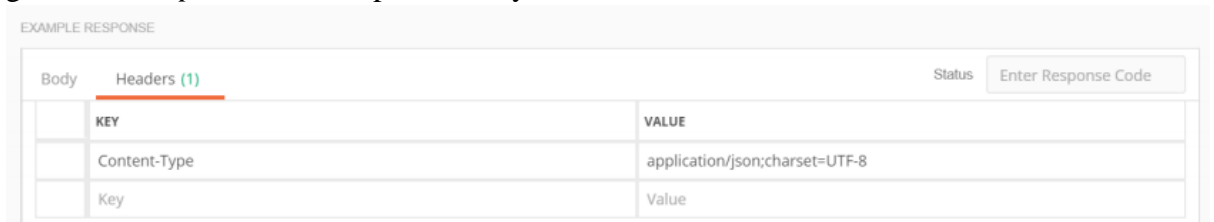
- e. Klik “Examples (0)” >> “Add Example”.



- f. Membuat example baru yang bernama “Status” sbb.:

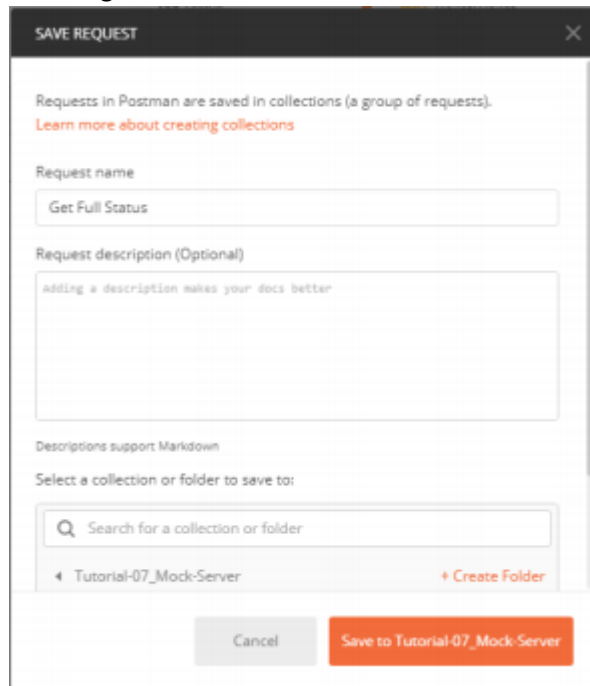


- g. Untuk *response header* spesifikasinya sbb.:



Sary Yuni A
1706106980

- h. Membuat request lain dengan nama “Get Full Status” lalu Save.



SAVE REQUEST

Requests in Postman are saved in collections (a group of requests).
[Learn more about creating collections](#)

Request name
Get Full Status

Request description (Optional)
Adding a description makes your docs better

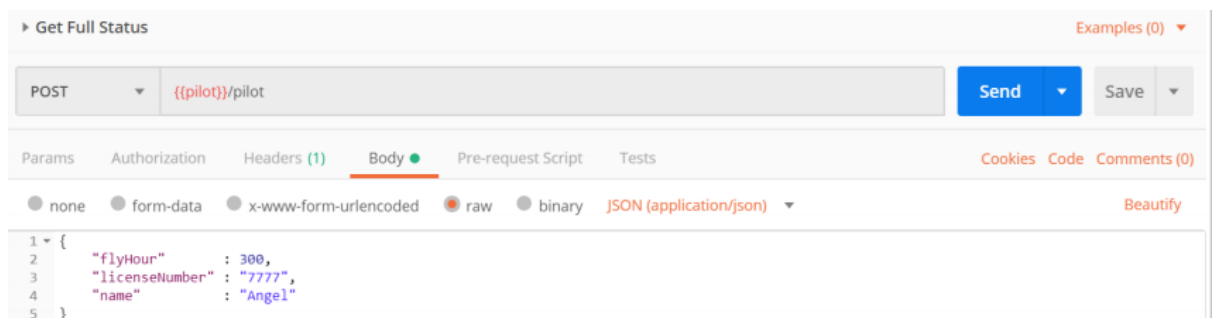
Descriptions support Markdown

Select a collection or folder to save to:

Search for a collection or folder

Tutorial-07_Mock-Server + Create Folder

Cancel Save to Tutorial-07_Mock-Server



Get Full Status Examples (0)

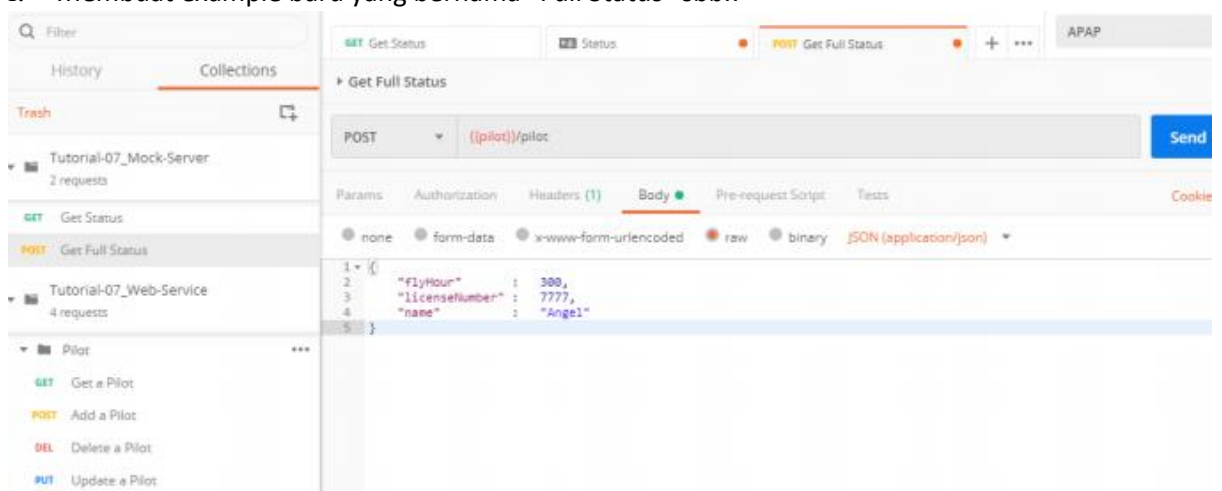
POST {{pilot}}/pilot Send Save

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Code Comments (0)

none form-data x-www-form-urlencoded raw binary JSON (application/json) Beautify

```
1 {  
2   "flyHour" : 300,  
3   "licenseNumber" : "7777",  
4   "name" : "Angel"  
5 }
```

- i. Membuat example baru yang bernama “Full Status” sbb.:



Filter History Collections

Tutorial-07_Mock-Server 2 requests

GET Get Status

POST Get Full Status

Tutorial-07_Web-Service 4 requests

Pilot

GET Get a Pilot

POST Add a Pilot

DEL Delete a Pilot

PUT Update a Pilot

GET Get Status Status POST Get Full Status + ... APAP

Get Full Status

POST {{pilot}}/pilot Send

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

none form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {  
2   "flyHour" : 300,  
3   "licenseNumber" : 7777,  
4   "name" : "Angel"  
5 }
```

- c. Membuat *response header* spesifikasinya sbb.:

Body	Headers (1)	Status	Enter Response Code
	KEY	VALUE	
	Content-Type	application/json; charset=UTF-8	
	Key	Value	

- d. Klik New >> Mock Server kemudian Pilih “Use collection from this workspace” >> “Tutorial-07_Mock-Server. Klik Create.

1. Select requests to mock 2. Configure mock server 3. Next steps

Create a new API Use collection from this workspace

Enter the requests and their expected responses below. Add headers to these requests by clicking on the (***) icon.

Tutorial-07_Web-Service
4 requests

Tutorial-07_Mock-Server
2 requests

- e. Akan terlihat URL untuk mengakses server baru.

✓ Select requests to mock ✓ Configure mock server 3. Next steps

Tutorial-07_Mock-Server mock server created
A mock server responds with sample responses, without an actual backend

NEXT STEPS

- Send a request to this mock URL
<https://ab65aa04-b05c-48c6-ae6b-b22dcf1f733a.mock.pstmn.io/> followed by the request path
- Modify the response of the mock server
Editing the Example of the request in the collection would change the response served by the mock server. [Learn more](#)

- f. Copy url yang di generate pada environment “Tutorial-07_Mock-Server” lalu simpan pada environment “APAP” dengan nama variable “pilot.”

<input checked="" type="checkbox"/> pilot	https://ab65aa04-b05c-48c6-ae6b-b22dcf1f733a.mock.pstmn.io/
Add a new variable	

- g. Pada collection “Tutorial-07_Web-Service”, buat folder dengan nama “Mock” di dalam folder “Pilot”. Lalu duplicate request “Get Status” dan “Get Full Status” pada collection “Tutorial07_Mock-Server” dan pindahkan hasil duplicate tersebut ke dalam folder “Mock” yang baru saja dibuat (tip: rename agar konsisten).

ADD FOLDER TO PILOT

Name

Mock

- h. Menjalankan kedua request tersebut pada environment “APAP”, response yang diberikan seharusnya sesuai dengan example yang sudah dibuat.

GET Get Status

GET `{{pilot}}/pilot?licenseNumber=7654`

Params Authorization Headers (7) Body Pre-request Script Tests

Query Params

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> licenseNumber	7654	
Key	Value	Description

Body Cookies Headers (13) Test Results Status: 200 OK Time: 2001 ms Size: 45

Pretty Raw Preview JSON

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

Get Full Status

POST `{{pilot}}/pilot`

Params Authorization Headers (9) Body Pre-request Script Tests

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (13) Test Results Status: 200 OK Time: 435 ms Size: 48

Pretty Raw Preview JSON

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

6. Membuat Service Consumer

- a. Menambahkan dependency berikut pada pom.xml.

```
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
</dependency>
```

- b. Buatlah package baru dengan nama com.apap.tu07.res.

Java Package

Create a new Java package.



Creates folders corresponding to packages.

Source folder:

Name:

- c. Membuat class PilotDetail.java.

```
package com.apap.tu07.rest;

import java.util.Date;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonProperty;;

/**
 * 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;
    }
}
```

- d. Membuat class bernama Setting.java pada package rest.

```
package com.apap.tu07.rest;

public class Setting {
    final public static String pilotUrl = "https://52d2de8e-5965-4472-83d8-e153b63dd3ae.mock.pstmn.io";
}
```

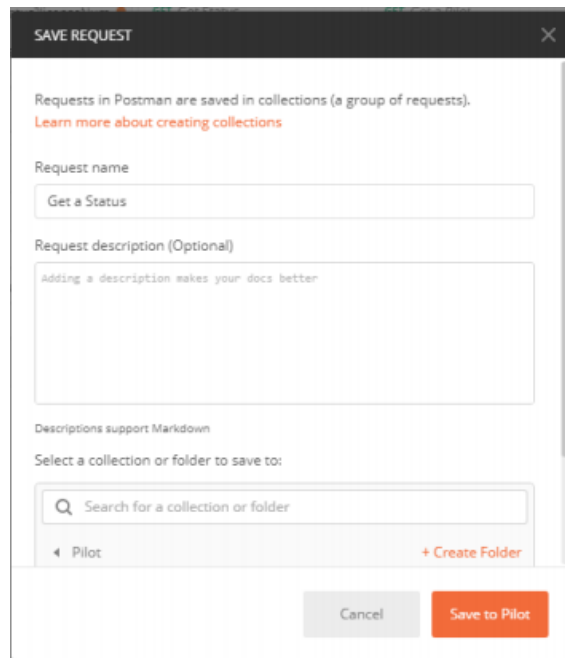
- e. Pada PilotController tambahkan kode di bawah ini.

```
@Autowired
RestTemplate restTemplate;

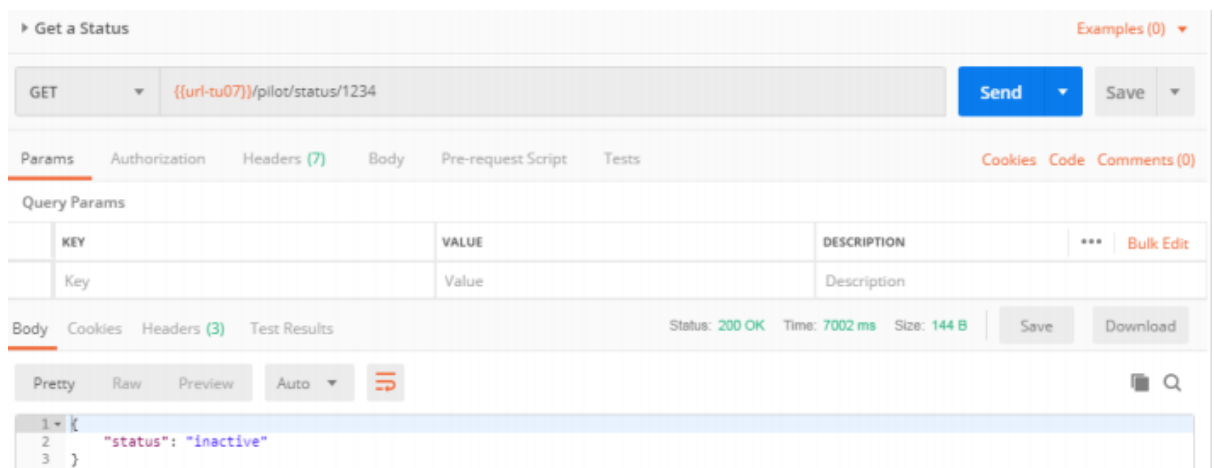
@Bean
public RestTemplate rest() {
    return new RestTemplate();
}

@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).get();
    PilotDetail detail = restTemplate.postForObject(path, pilot, PilotDetail.class);
    return detail;
}
```



The 'SAVE REQUEST' dialog box in Postman. It contains a title bar 'SAVE REQUEST' with a close button. The main text says 'Requests in Postman are saved in collections (a group of requests). Learn more about creating collections'. There is a 'Request name' field with 'Get a Status' entered. Below it is a 'Request description (Optional)' text area with the placeholder 'Adding a description makes your docs better'. A note says 'Descriptions support Markdown'. Then, 'Select a collection or folder to save to:' is followed by a search bar 'Search for a collection or folder' and a list showing 'Pilot' with a '+ Create Folder' button. At the bottom are 'Cancel' and 'Save to Pilot' buttons.



The Postman interface for a GET request. The top bar shows 'Get a Status' and 'Examples (0)'. The request method is 'GET' and the URL is '{{url-tu07}}/pilot/status/1234'. There are 'Send', 'Save', and 'Dropdown' buttons. Below the request bar are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', and 'Tests'. The 'Params' tab is active, showing a table for 'Query Params' with columns 'KEY', 'VALUE', 'DESCRIPTION', and 'Bulk Edit'. The table has one row with 'Key' and 'Value'. Below the table are tabs for 'Body', 'Cookies', 'Headers (3)', and 'Test Results'. The 'Body' tab is active, showing a JSON response:

```
{  "status": "inactive"}
```

. The status bar at the bottom shows 'Status: 200 OK', 'Time: 7002 ms', 'Size: 144 B', and 'Save' and 'Download' buttons.

7. Latihan

- Buatlah FlightController menjadi RestController sesuai spesifikasi.

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

    @Autowired
    private PilotService pilotService;
```

```
@PostMapping(value = "/add/{licenseNumber}")
private FlightModel addFlight(@PathVariable(value = "licenseNumber") String licenseNumber,
    @RequestBody FlightModel flight, Model model) {
    PilotModel pilot = pilotService.getPilotDetailByLicenseNumber(licenseNumber).get();
    pilot.setListFlight(new ArrayList<FlightModel>(){
        private ArrayList<FlightModel> init(){
            this.add(new FlightModel());
            return this;
        }
    }).init());

    model.addAttribute("pilot", pilot);
    return flightService.addFlight(flight);
}

@RequestMapping(value = "/flight/add/{licenseNumber}", method = RequestMethod.POST, params={
private String addRow(@ModelAttribute PilotModel pilot, Model model) {
    pilot.getListFlight().add(new FlightModel());
    model.addAttribute("pilot", pilot);
    return "add-flight";
}

@RequestMapping(value="/flight/add/{licenseNumber}", method = RequestMethod.POST, params={"
public String removeRow(@ModelAttribute PilotModel pilot, Model model, HttpServletRequest req) {
    Integer rowId = Integer.valueOf(req.getParameter("removeRow"));
    pilot.getListFlight().remove(rowId.intValue());

    model.addAttribute("pilot", pilot);
    return "add-flight";
}

@GetMapping(value = "/view/{licenseNumber}")
private FlightModel flightView(@PathVariable("flightNumber") String flightNumber) {
    FlightModel flight = flightService.getFlightDetailByFlightNumber(flightNumber).get();
    return flight;
}

@GetMapping(value = "/all")
private FlightModel flightViewAll(@RequestBody FlightModel flight) {
    return (FlightModel) flightService.getAllFlight();
}

@RequestMapping(value = "/flight/view/", method = RequestMethod.GET)
public String viewAll(@RequestParam("licenseNumber") String licenseNumber, Model model) {
    model.addAttribute("flist", flightService.getAllFlight());
    model.addAttribute("found", true);
    model.addAttribute("licenseNumber", licenseNumber);
    return "view-flight";
}

@DeleteMapping(value = "/delete")
private String deleteFlight(@RequestParam("flightNumb") String flightNumb) {
    FlightModel flight = flightService.getFlightDetailByFlightNumber(flightNumb);
    flightService.deleteFlight(flight);
    return "success deleting flight";
}
```



```
@PutMapping(value = "/update/{flightNum}")
private String updateFlight(@PathVariable("flightNum") String flightNum,
    @RequestParam("destination") String dest,
    @RequestParam("origin") String ori,
    @RequestParam("time") Date time) {
    FlightModel flight = flightService.getFlightDetailByFlightNumber(flightNum).get();
    if(flight.equals(null)) {
        return "Couldn't find your flight";
    }
    flight.setDestination(dest);
    flight.setOrigin(ori);
    flight.setTime(time);
    flightService.updateFlight(flightNum, flight);
    return "update-flight";
}
```

- b. Kemudian, add folder dengan nama flight.
- c. Pada flight, add request, dengan nama Get a Flight

The screenshot shows a Postman request for 'Get a Flight' using the GET method. The URL is `{{url-tu07}}/flight/view/1131`. The 'Params' tab is active, showing a table with one parameter: 'Key' with value 'Value' and description 'Description'. The 'Body' tab is also active, showing a JSON response in 'Pretty' format. The status is 200 OK, with a time of 78 ms and size of 308 B.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
1 {
2   "id": 1,
3   "flightNumber": "1131",
4   "origin": "BDO - Bandar Udara Internasional Husein Sastranegara",
5   "destination": "AMQ - Bandar Udara Internasional Pattimura, Ambon",
6   "time": "1970-07-07"
7 }
```

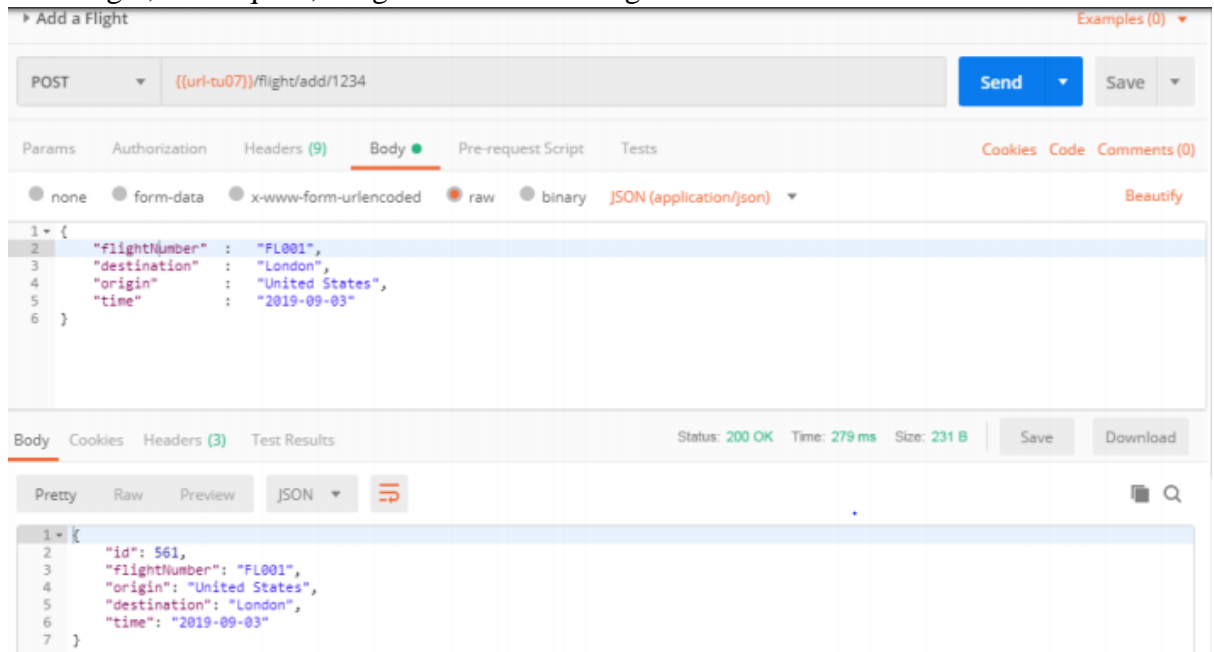
- d. Pada flight, add request, dengan nama Get all Flight

The screenshot shows a Postman request for 'Get All Flight' using the GET method. The URL is `{{url-tu07}}/flight/all`. The 'Params' tab is active, showing a table with one parameter: 'Key' with value 'Value' and description 'Description'. The 'Body' tab is also active, showing a JSON response in 'Pretty' format. The status is 200 OK, with a time of 262 ms and size of 98.74 KB.

KEY	VALUE	DESCRIPTION
Key	Value	Description

```
1 [
2   {
3     "id": 1,
4     "flightNumber": "1131",
5     "origin": "BDO - Bandar Udara Internasional Husein Sastranegara",
6     "destination": "AMQ - Bandar Udara Internasional Pattimura, Ambon",
7     "time": "1970-07-07"
8   },
9   {
10    "id": 2,
11    "flightNumber": "1151",
12    "origin": "SRG - Bandar Udara Internasional Achmad Yani, Sema",
13    "destination": "AMQ - Bandar Udara Internasional Pattimura, Ambon",
14    "time": "1961-11-10"
15  },
16  {
17    "id": 3,
18    "flightNumber": "1176",
19    "origin": "TIM - Bandar Udara Internasional Mozes Kilangin, M",
```

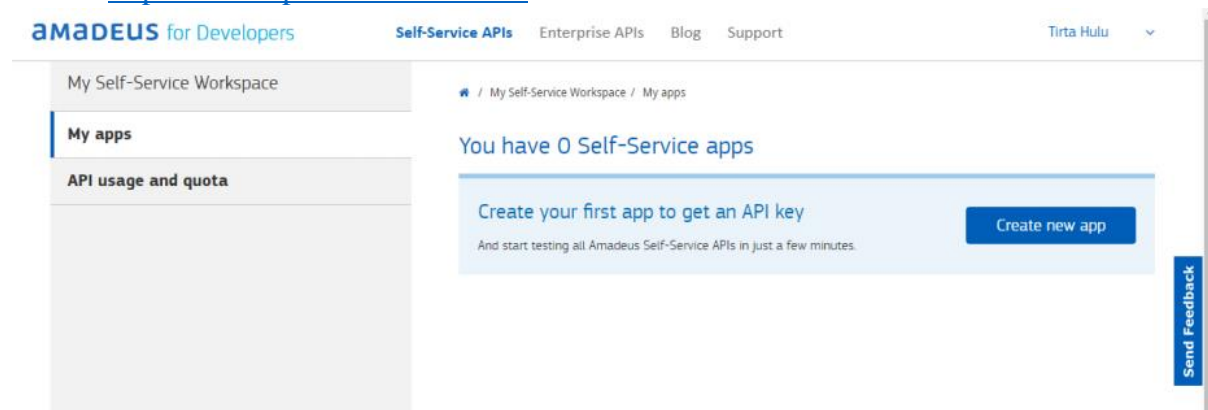

- e. Pada flight, add request, dengan nama Add a Flight



- f. Pada flight, add request, dengan nama delete a Flight
g. Pada flight, add request, dengan nama update a Flight

2. Coba pelajari dokumentasi API yang ada pada link berikut:

<https://developers.amadeus.com/>



amadeus for Developers

Self-Service APIsEnterprise APISupport

Tirta Hulu

My Self-Service Workspace

My Self-Service Workspace / My apps

Create new app

App name*

tutorial-07

Targeted to (optional)

☐ Business to consumer ☐ Business to business

Description (optional)

Enter a description

CancelCreate

Send Feedback

Activat

```
<dependencies>
  <dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.5</version>
  </dependency>
  <dependency>
    <groupId>com.amadeus</groupId>
    <artifactId>amadeus-java</artifactId>
    <version>3.1.0</version>
  </dependency>
```

```
1 package com.apap.tu07.controller;
2
3 import com.amadeus.Amadeus;
4 import com.amadeus.Params;
5
6 import com.amadeus.exceptions.ResponseException;
7 import com.amadeus.referenceData.Locations;
8 import com.amadeus.resources.Location;
9
10 public class AmadeusExample {
11     public static void main(String[] args) throws ResponseException {
12         Amadeus amadeus = Amadeus
13             .builder("jdPZPAFmKq2iRyQ2uUWhq8AjpPoCi1wk", "PSQALPuZLIgW55W9")
14             .build();
15
16         Location[] locations = amadeus.referenceData.locations.get(Params
17             .with("keyword", "LON")
18             .and("subType", Locations.ANY));
19
20         System.out.println(locations);
21     }
22 }
```

Sary Yuni A
1706106980

SAVE REQUEST

Requests in Postman are saved in collections (a group of requests).
[Learn more about creating collections](#)

Request name

Get Airport

Request description (Optional)

Adding a description makes your docs better

Descriptions support Markdown

Select a collection or folder to save to:

Search for a collection or folder

◀ Tutorial-07_Mock-Server

+ Create Folder

Cancel

Save to Tutorial-07_Mock-Server