FINAL PROJECT REPORT

REST-API DEVELOPMENT USING JAVA SPRINGBOOT
(BUS RESERVATION APPLICATION)

Oleh:

M. MACHRUSH ALIY SIROJJAM MUSHLICH

NIK 095061

BFI Technology Development Program (BTDP)

PT BFI Finance Indonesia Tbk

Tangerang Selatan

2022

CONTENT

PRELIMINARY

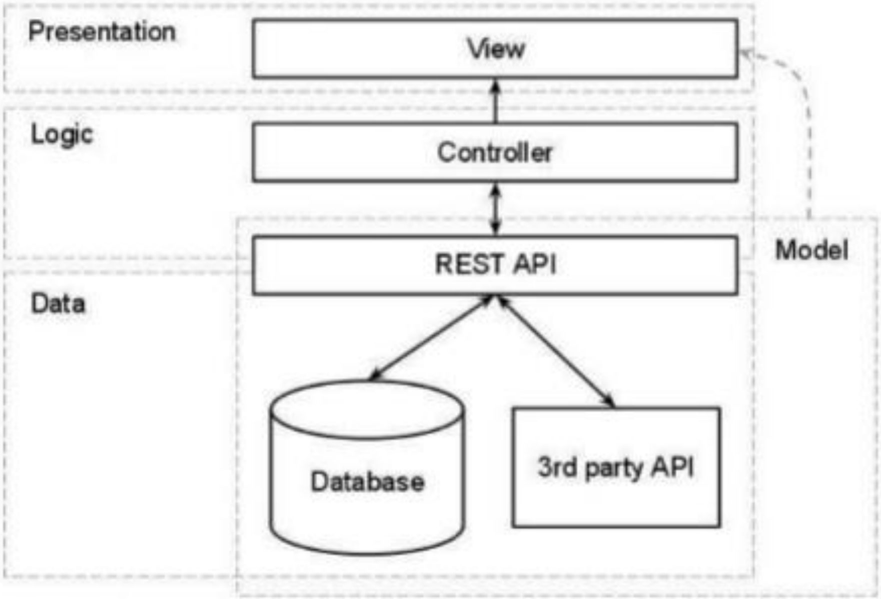
APPLICATION REQUIREMENTS

- Sign-up Modul
- Authentification (Log-in) Modul
- User Modul
- Agency Modul
- Bus Modul
- Reservation Modul
- Stop Modul
- Ticket Modul
- Trip Modul
- Trip Schedule Modul

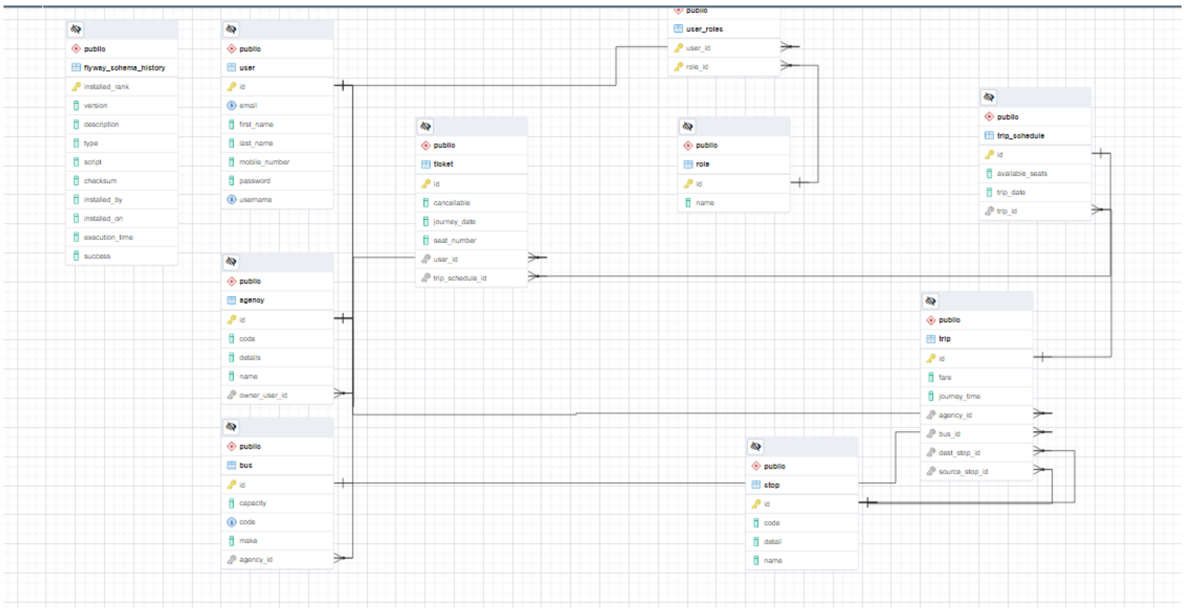
BUILD RESOURCES

- JDK 1.8 (Java 8)
- Swagger UI V2
- Eclipse IDE 2022-06
- Windows 11 Pro
- Spring Boot Framework
- PostgreSQL 14

ARCHITECTURE APPLICATION DIAGRAM



ENTITY RELATIONAL DATABASE



DATABASE SCHEMA

https://raw.githubusercontent.com/jampirojam/FinalProjectOjam/master/db/migration/V1.1__database_schema.sql

RAW DATA

https://raw.githubusercontent.com/jampirojam/FinalProjectOjam/master/db/migration/V1.2__data_raw.sql

PROJECT

https://github.com/jampirojam/FinalProjectOjam

APPLICATION GUIDE

HOW TO RUN

Create Database

- Login to PostgreSQL Management Database Admin (PgAdmin 4)
- Create database, in this section we use “datafinal”
- Migrate “database schema” and “data raw” using import database or SQL query editor

Run on IDE

- Open Eclipse IDE
- Open Project and wait until build project completed
- Move data on db/migration into resources:

  ```
  src
  ----- main
  ---------- resources
  --------------- db
  -------------------- migration
  ------------------------- V1.1__database_schema.sql
  ------------------------- V1.2__database_raw.sql
  ```
- Right click then run as Maven Clean
- After completed run as Maven Install
- After completed run as Spring Boot App

Run on Browser using Localhost

- Open Browser
- Type localhost:4596

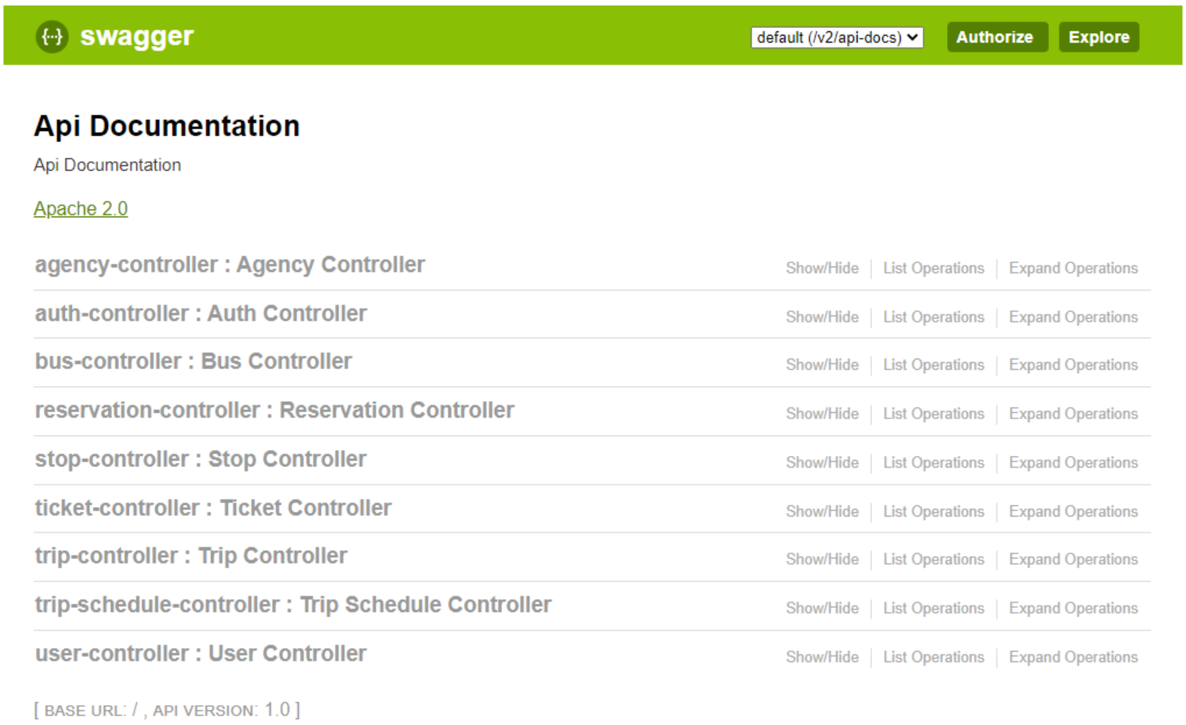
Run on Browser using Heroku

- Open Browser
- Type https://final-project-ojam.herokuapp.com/

Note**:** this action run for swagger display only, because database management is not free.

MODUL

SWAGGER UI



Api Documentation

Api Documentation

Apache 2.0

agency-controller : Agency Controller — Show/Hide | List Operations | Expand Operations

auth-controller : Auth Controller — Show/Hide | List Operations | Expand Operations

bus-controller : Bus Controller — Show/Hide | List Operations | Expand Operations

reservation-controller : Reservation Controller — Show/Hide | List Operations | Expand Operations

stop-controller : Stop Controller — Show/Hide | List Operations | Expand Operations

ticket-controller : Ticket Controller — Show/Hide | List Operations | Expand Operations

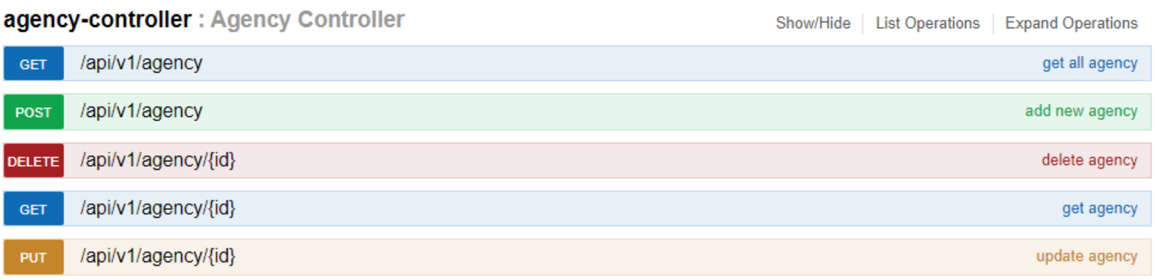
trip-controller : Trip Controller — Show/Hide | List Operations | Expand Operations

trip-schedule-controller : Trip Schedule Controller — Show/Hide | List Operations | Expand Operations

user-controller : User Controller — Show/Hide | List Operations | Expand Operations

[BASE URL: / , API VERSION: 1.0]

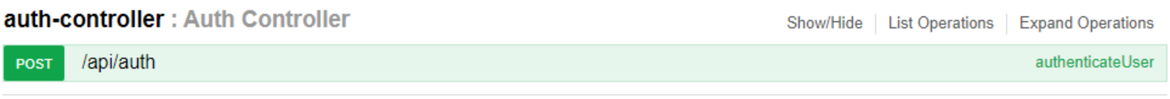
AGENCY MODUL



agency-controller : Agency Controller

GET	/api/v1/agency	get all agency
POST	/api/v1/agency	add new agency
DELETE	/api/v1/agency/{id}	delete agency
GET	/api/v1/agency/{id}	get agency
PUT	/api/v1/agency/{id}	update agency

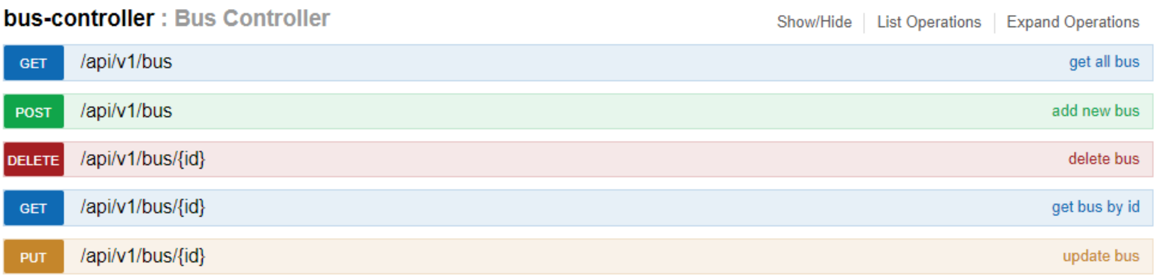
AUTHENTICATION MODUL



auth-controller : Auth Controller

POST	/api/auth	authenticateUser

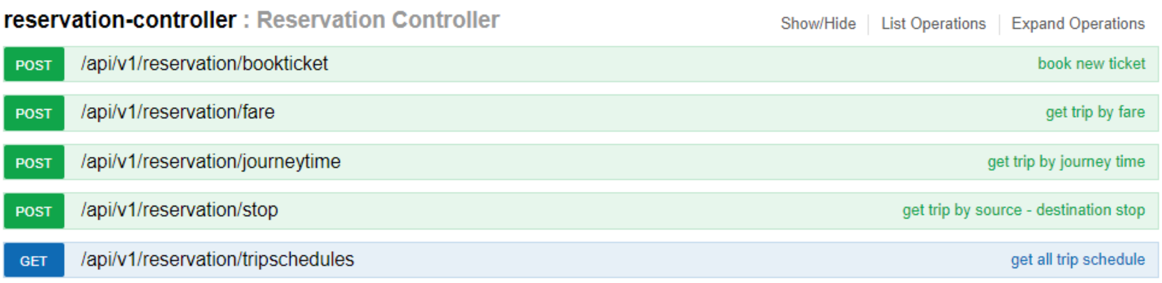
BUS MODUL



bus-controller : Bus Controller

GET	/api/v1/bus	get all bus
POST	/api/v1/bus	add new bus
DELETE	/api/v1/bus/{id}	delete bus
GET	/api/v1/bus/{id}	get bus by id
PUT	/api/v1/bus/{id}	update bus

RESERVATION MODUL

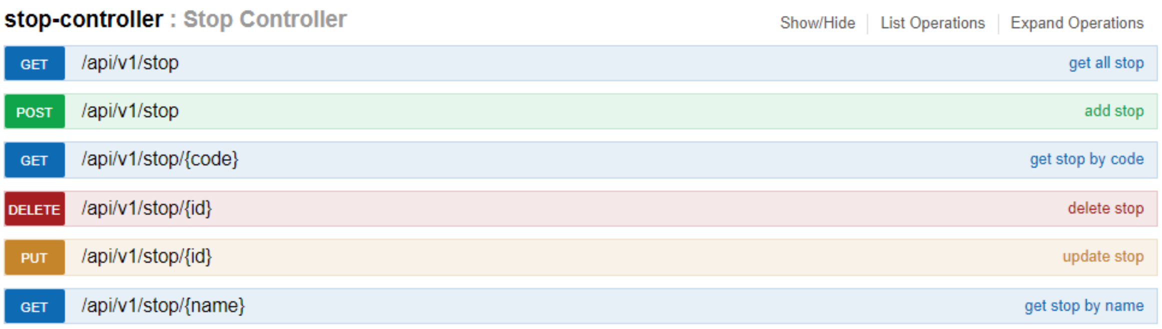


reservation-controller : Reservation Controller

Show/Hide | List Operations | Expand Operations

POST	/api/v1/reservation/bookticket	book new ticket
POST	/api/v1/reservation/fare	get trip by fare
POST	/api/v1/reservation/journeytime	get trip by journey time
POST	/api/v1/reservation/stop	get trip by source - destination stop
GET	/api/v1/reservation/tripschedules	get all trip schedule

STOP MODUL

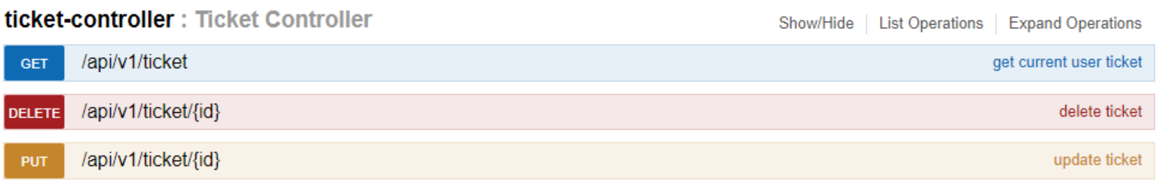


stop-controller : Stop Controller

Show/Hide | List Operations | Expand Operations

GET	/api/v1/stop	get all stop
POST	/api/v1/stop	add stop
GET	/api/v1/stop/{code}	get stop by code
DELETE	/api/v1/stop/{id}	delete stop
PUT	/api/v1/stop/{id}	update stop
GET	/api/v1/stop/{name}	get stop by name

TICKET MODUL

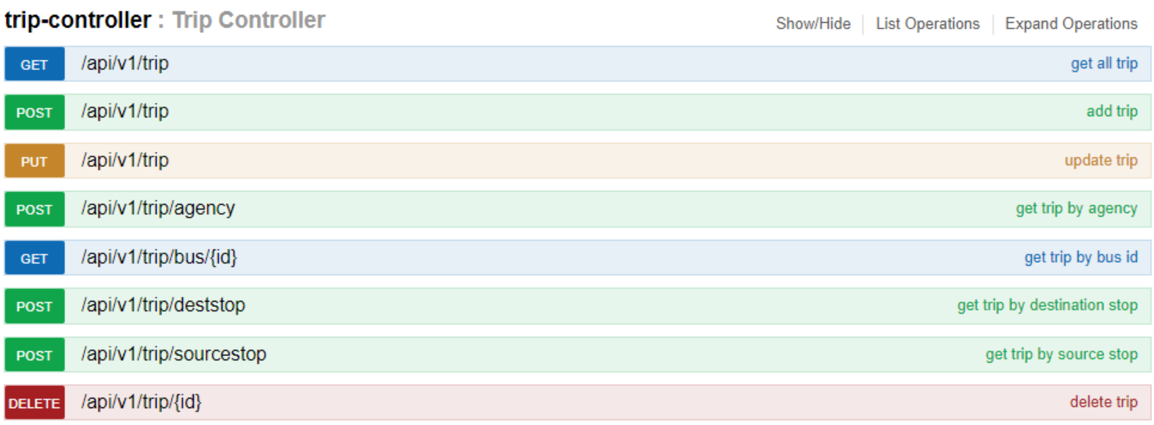


ticket-controller : Ticket Controller

Show/Hide | List Operations | Expand Operations

GET	/api/v1/ticket	get current user ticket
DELETE	/api/v1/ticket/{id}	delete ticket
PUT	/api/v1/ticket/{id}	update ticket

TRIP MODUL

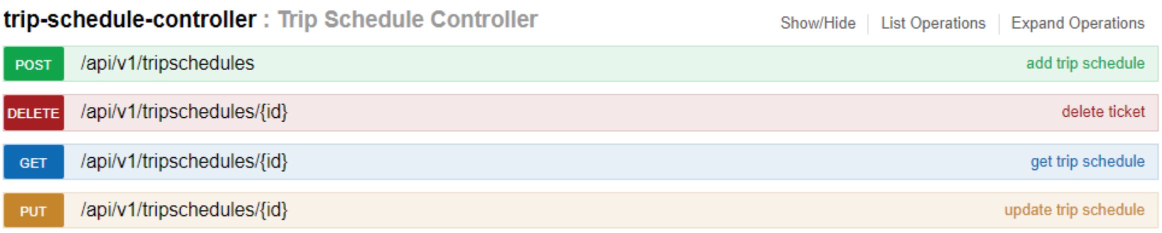


trip-controller : Trip Controller

Show/Hide | List Operations | Expand Operations

GET	/api/v1/trip	get all trip
POST	/api/v1/trip	add trip
PUT	/api/v1/trip	update trip
POST	/api/v1/trip/agency	get trip by agency
GET	/api/v1/trip/bus/{id}	get trip by bus id
POST	/api/v1/trip/deststop	get trip by destination stop
POST	/api/v1/trip/sourcestop	get trip by source stop
DELETE	/api/v1/trip/{id}	delete trip

TRIP SCHEDULE MODUL

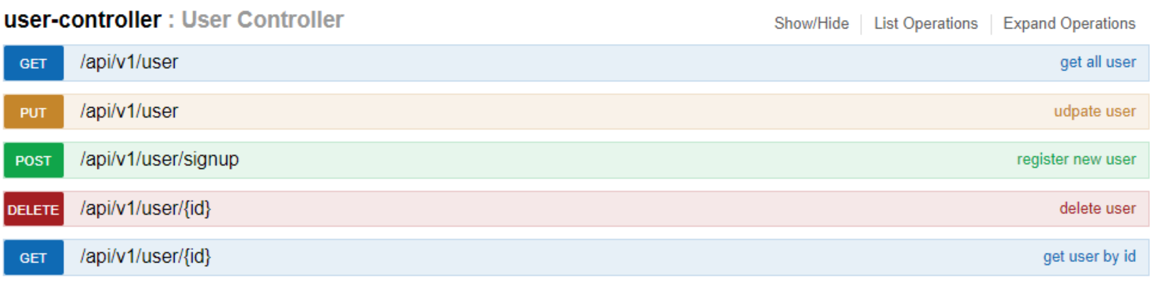


trip-schedule-controller : Trip Schedule Controller

Show/Hide | List Operations | Expand Operations

POST	/api/v1/tripschedules	add trip schedule
DELETE	/api/v1/tripschedules/{id}	delete ticket
GET	/api/v1/tripschedules/{id}	get trip schedule
PUT	/api/v1/tripschedules/{id}	update trip schedule

USER MODUL



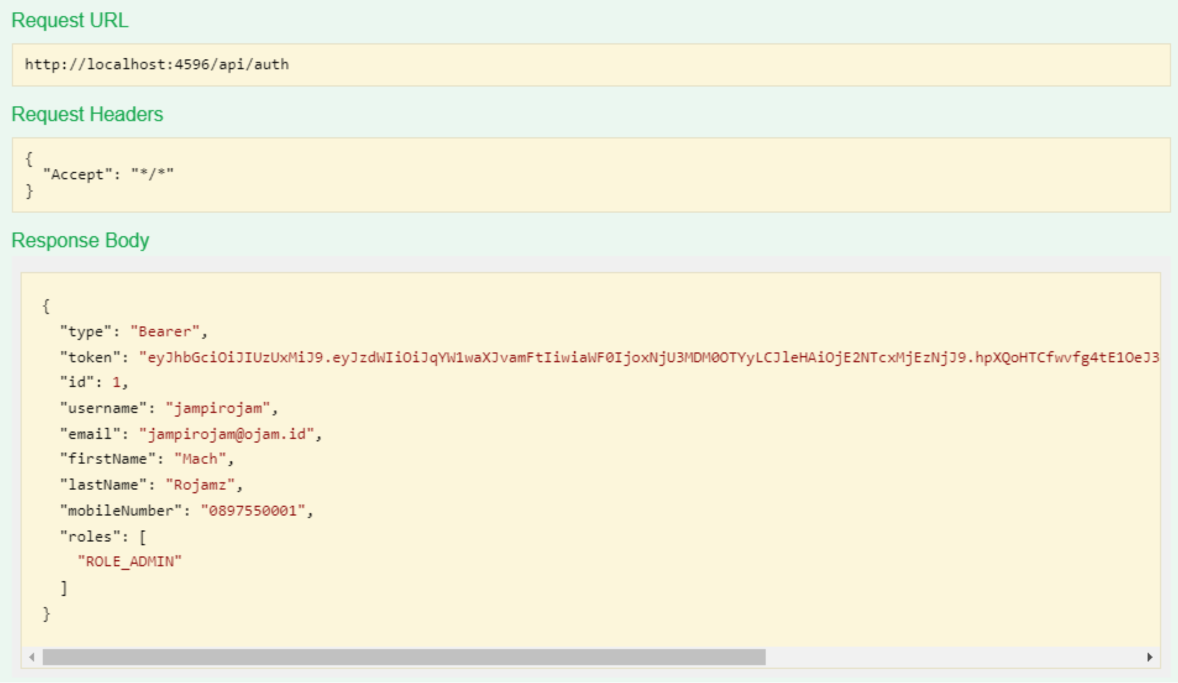
user-controller : User Controller

GET	/api/v1/user	get all user
PUT	/api/v1/user	udpate user
POST	/api/v1/user/signup	register new user
DELETE	/api/v1/user/{id}	delete user
GET	/api/v1/user/{id}	get user by id

SUCCESS AND ERROR RESPONSE ENDPOINT

PRELIMINARY

Before check the success and error response, first time we should to get Token for authentification (login) using username or password like below.



Request URL

```
http://localhost:4596/api/auth
```

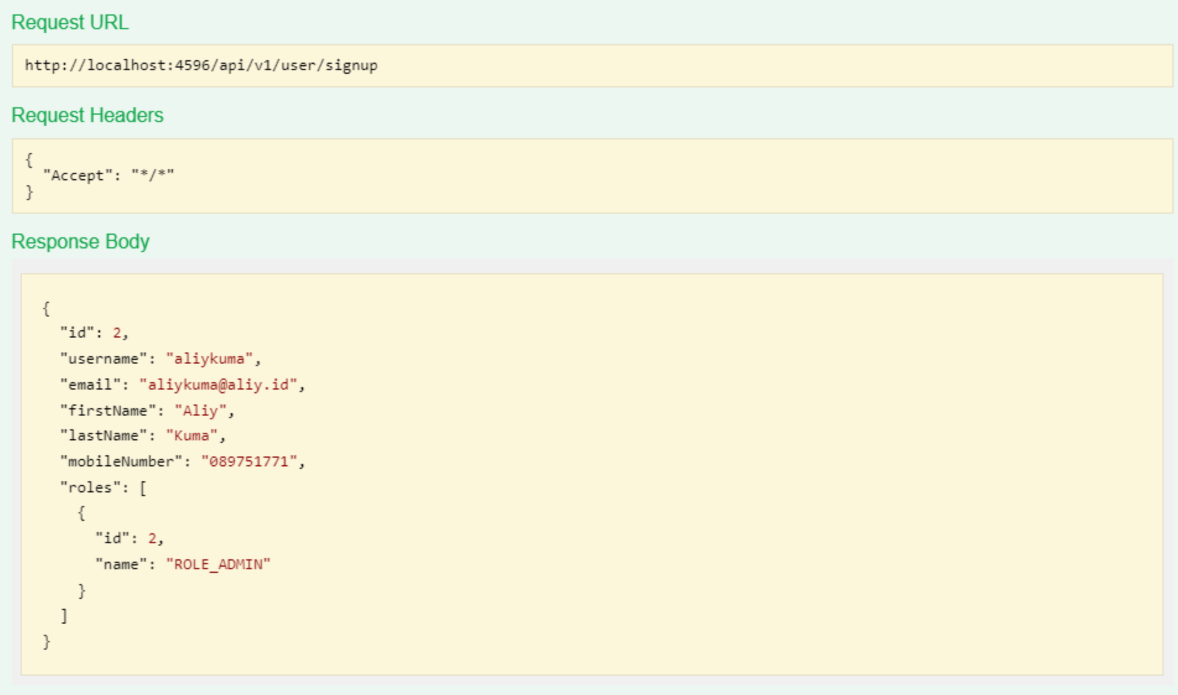
Request Headers

```
{
  "Accept": "*/*"
}
```

Response Body

```
{
  "type": "Bearer",
  "token": "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJqYW1waXJvamFtIiwiaWF0IjoxNjU3MDM0OTYyLCJleHAiOjE2NTcxMjEzNjJ9.hpXQoHTCfwvfg4tE1OeJ3
  "id": 1,
  "username": "jampirojam",
  "email": "jampirojam@ojam.id",
  "firstName": "Mach",
  "lastName": "Rojamz",
  "mobileNumber": "0897550001",
  "roles": [
    "ROLE_ADMIN"
  ]
}
```

Or, if we don't have an account should create the account for authentification, and this also for check the success response.



Request URL

```
http://localhost:4596/api/v1/user/signup
```

Request Headers

```
{
  "Accept": "*/*"
}
```

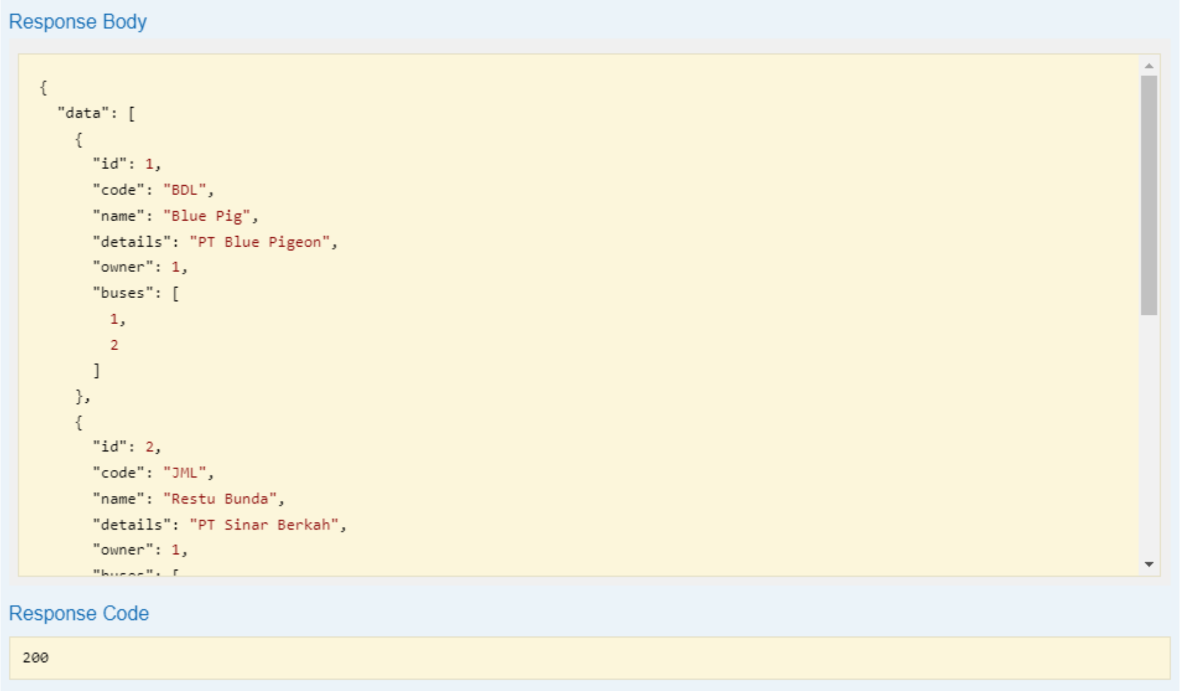
Response Body

```
{
  "id": 2,
  "username": "aliykuma",
  "email": "aliykuma@aliy.id",
  "firstName": "Aliy",
  "lastName": "Kuma",
  "mobileNumber": "089751771",
  "roles": [
    {
      "id": 2,
      "name": "ROLE_ADMIN"
    }
  ]
}
```

After that, we authentification using token that we get from the authentification modul. To use the token, go to Authorize then input "Type + Token", example below

Bearer 76aSuwdjYS6KJ97/g76GhsbgTT7879gFHghdr.aGTY86712d3d3OgR686defh

SUCCESS REPONSE GET



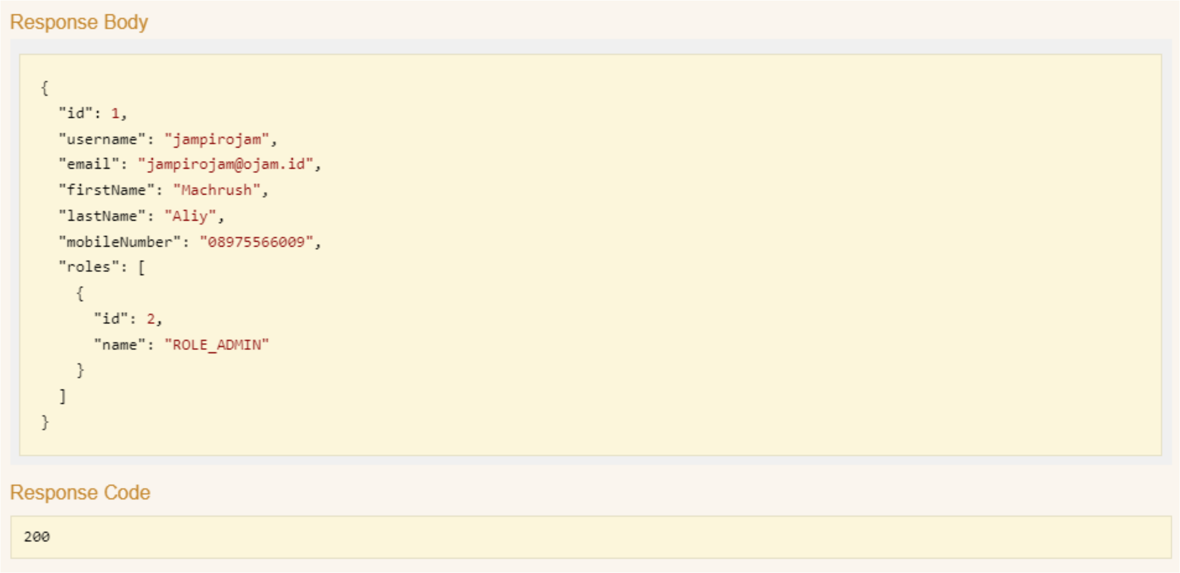
Response Body

```
{
  "data": [
    {
      "id": 1,
      "code": "BDL",
      "name": "Blue Pig",
      "details": "PT Blue Pigeon",
      "owner": 1,
      "buses": [
        1,
        2
      ]
    },
    {
      "id": 2,
      "code": "JML",
      "name": "Restu Bunda",
      "details": "PT Sinar Berkah",
      "owner": 1,
      "buses": [
```

Response Code

```
200
```

SUCCESS REPONSE PUT



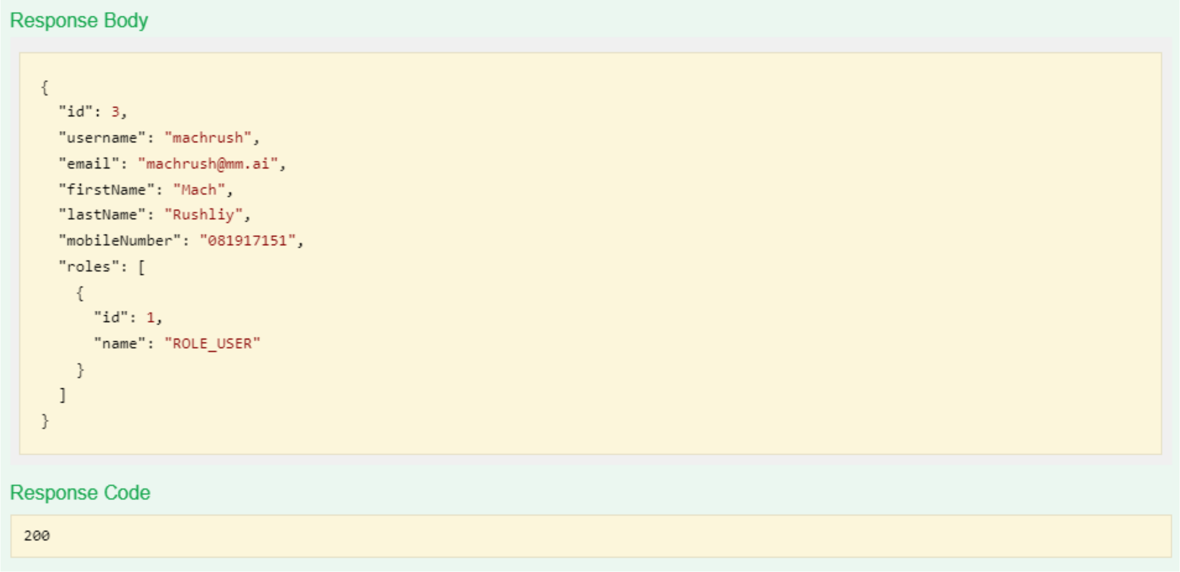
Response Body

```
{
  "id": 1,
  "username": "jampirojam",
  "email": "jampirojam@ojam.id",
  "firstName": "Machrush",
  "lastName": "Aliy",
  "mobileNumber": "08975566009",
  "roles": [
    {
      "id": 2,
      "name": "ROLE_ADMIN"
    }
  ]
}
```

Response Code

```
200
```

SUCCESS REPONSE POST



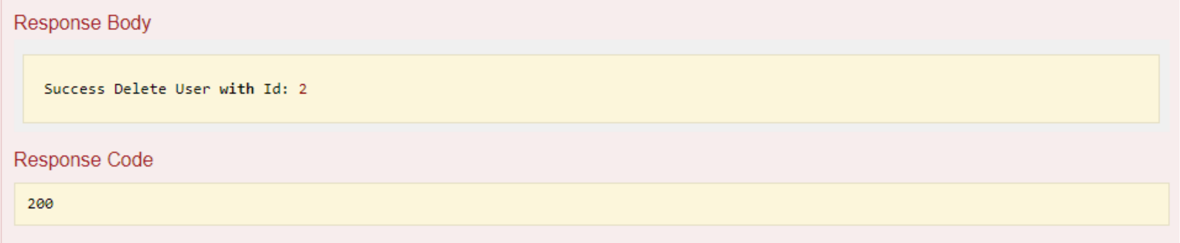
Response Body

```
{
  "id": 3,
  "username": "machrush",
  "email": "machrush@mm.ai",
  "firstName": "Mach",
  "lastName": "Rushliy",
  "mobileNumber": "081917151",
  "roles": [
    {
      "id": 1,
      "name": "ROLE_USER"
    }
  ]
}
```

Response Code

```
200
```

SUCCESS REPONSE DELETE



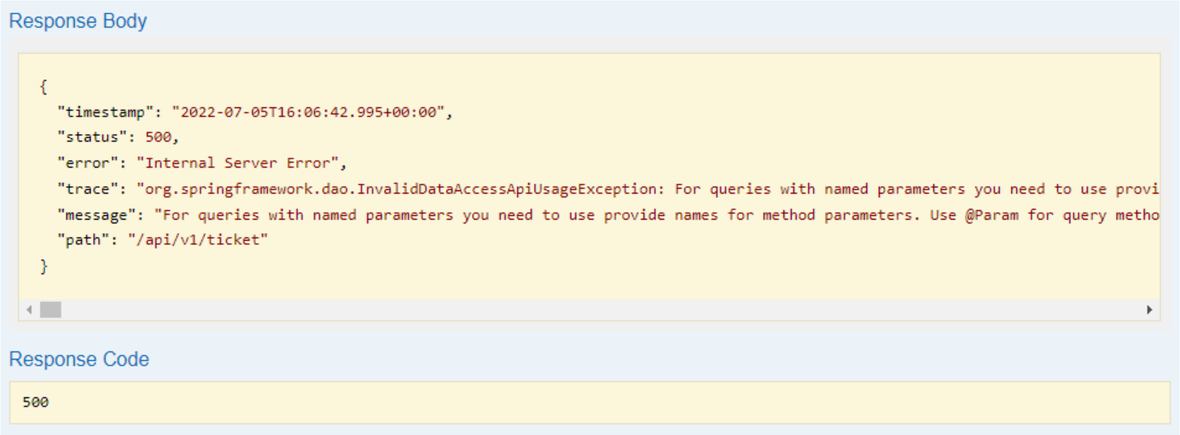
Response Body

```
Success Delete User with Id: 2
```

Response Code

```
200
```

ERROR REPONSE GET



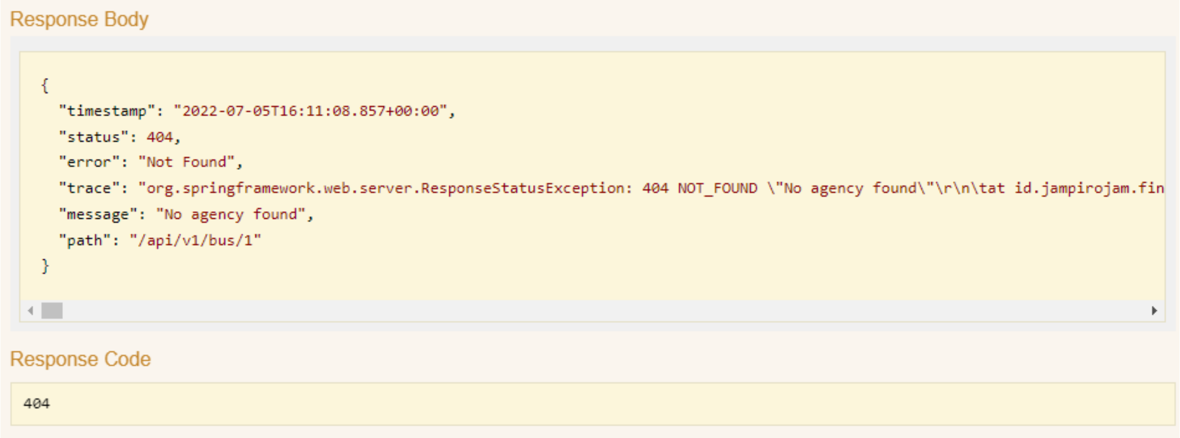
Response Body

```
{
  "timestamp": "2022-07-05T16:06:42.995+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "trace": "org.springframework.dao.InvalidDataAccessApiUsageException: For queries with named parameters you need to use provi
  "message": "For queries with named parameters you need to use provide names for method parameters. Use @Param for query metho
  "path": "/api/v1/ticket"
}
```

Response Code

```
500
```

ERROR REPONSE PUT



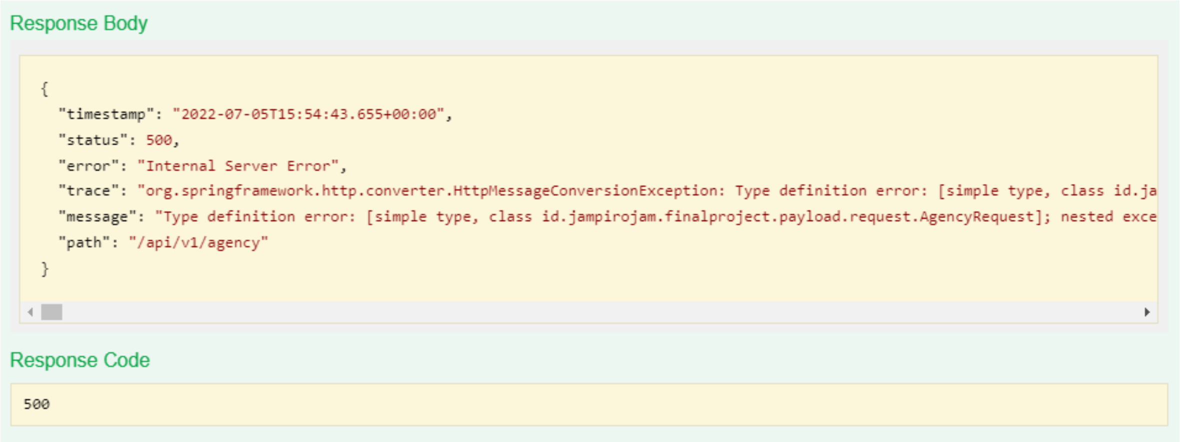
Response Body

```
{
  "timestamp": "2022-07-05T16:11:08.857+00:00",
  "status": 404,
  "error": "Not Found",
  "trace": "org.springframework.web.server.ResponseStatusException: 404 NOT_FOUND \"No agency found\"\r\n\tat id.jampirojam.fin
  "message": "No agency found",
  "path": "/api/v1/bus/1"
}
```

Response Code

```
404
```

ERROR REPONSE POST



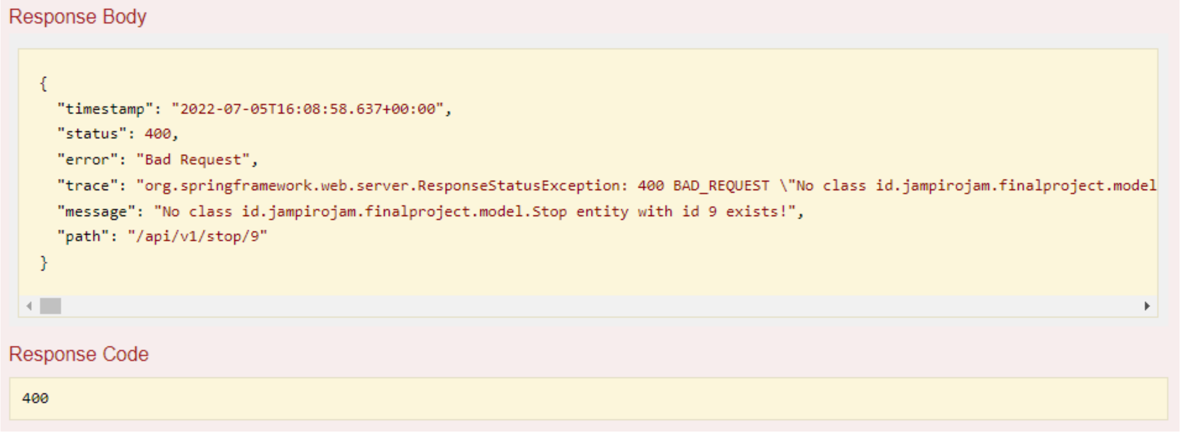
Response Body

```
{
  "timestamp": "2022-07-05T15:54:43.655+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "trace": "org.springframework.http.converter.HttpMessageConversionException: Type definition error: [simple type, class id.ja
  "message": "Type definition error: [simple type, class id.jampirojam.finalproject.payload.request.AgencyRequest]; nested exce
  "path": "/api/v1/agency"
}
```

Response Code

```
500
```

ERROR REPONSE DELETE



Response Body

```
{
  "timestamp": "2022-07-05T16:08:58.637+00:00",
  "status": 400,
  "error": "Bad Request",
  "trace": "org.springframework.web.server.ResponseStatusException: 400 BAD_REQUEST \"No class id.jampirojam.finalproject.model
  "message": "No class id.jampirojam.finalproject.model.Stop entity with id 9 exists!",
  "path": "/api/v1/stop/9"
}
```

Response Code

```
400
```