**BAB IV**

**HASIL DAN ANALISA**

* 1. Integrasi Aplikasi

API merupakan perantara yang menghubungkan satu aplikasi dengan aplikasi lainya untuk saling berkomunikasi/terintegrasi. API berisi kumpulan prosedur, fungsi atau cara komunikasi untuk interaksi antar aplikasi.

Dalam pengembangkan aplikasi perangkat lunak sangat sering dilakukan integrasi antar aplikasi baik buatan sendiri maupun produk dari orang/perusahaan lain. Contohnya pada toko Online yang harus terintegrasi antara produk milik sendiri, sistem pembayaran hingga sistem pengiriman yang di buat oleh perusahaan lain.

Secara garis besar terdapat 4 metode yang digunakan dalam integrasi antar aplikasi (Enterprise integration pattern)

1. File sharing

Metode ini merupakan integrasi yang sangat mudah untuk dilakukan dimana file yang telah di buat akan dibagikan ke aplikasi lain, file sharing akan bermanfaat apabila integrasi aplikasi tidak terhubung secara langsung. kelemahan dalam metode ini adalah file yang dibagikan tidak dapat dibagikan secara realtime.

1. Database sharing

Metode ini merupakan integrasi antar aplikasi yang memanfaatkan system basis data yang berada dalam 1 jaringan yang sama antara aplikasi dengan jarinag basis data, sehingga aplikasi hanya perlu melakukan CRUD (Create, Read, Update, Delete) pada database dan dapat langsung diakses oleh aplikasi lain. Kelemahan dalam metode ini adalah apabila 2 aplikasi melakukan perubahan data yang sama maka akan menyebabkan tabrakan data.

1. Remote Proccedur invocation

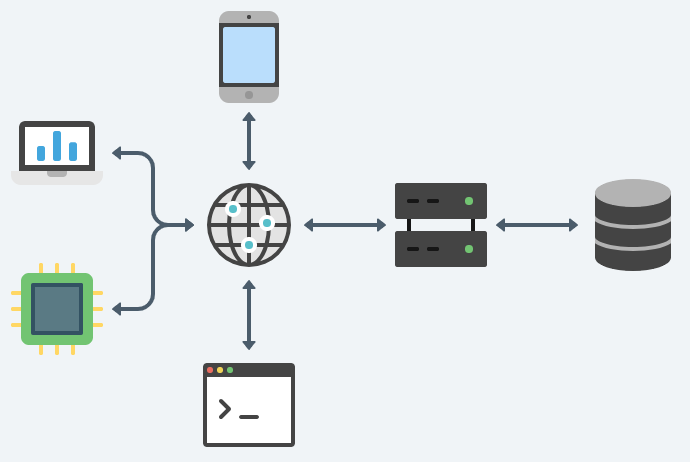
Metode ini juga dikenal dengan API (Aplication Programing Interface) dengan cara membuat yang dapat digunakan oleh aplikasi lain. Aplikasi utama yang memiliki data merpakan aplikasi utama yang akan menjadi API. Dan aplikasi yang membutuhkan data merupakan aplikasi yang akan menggunakan API. Kelebihan metode ini adalah integrasi dapat dilakukan secara realtime, dibandingkan dengan metode sharing basis data dimana jika terdapat perubahan pada data aplikasi perlu melakukan pengecekan secara manual pada database, sedangkan dengan metode ini API mengetahui semua perubahan data yang terjadi dalam system.

1. Messaging

Metode ini merupakan integrasi aplikasi yang memanfaatkan applikasi lain yang akan mengirimkan pesan yang disebut dengan message broker, metode ini cukup mirip dengan metode API dan Database sharing, namun metode ini tidak bersifat realtime dimana aplikasi menunggu pesan dari message broker untuk menerima update.

* 1. Hasil Perancangan API

RESTful API merupakan imlementasi API yang menggunakan HTTP sebagai protocol komunikasi. RESTful API dapat diadaptasi di semua Bahasa pemrograman yang dapat menggunakan HTTP seperti integrai aplikasi Web, aplikasi mobile dan aplikasi desktop. RESTful API juga dikenal sebagai Web Service dikarenakan metode ini menggunakan protocol HTTP yang biasa digunakan pada aplikasi web.



Gambar 4.1. Ilustrasi API

Pengembangan API setidaknya dilakukan dalam 4 tahapan yang akan dimulai dengan perancangan, pengembangan, pengujian dan implementasi. Dalam tahapan perancangan sangat di sarankan dengan menggunakan OpenAPI yang merupakan standarisasi untuk dokumentasi. Dalam penelitian ini penulis akan melakukan pengujian koneksi antar teknologi dan device untuk berkomunikasi sehingga akan dibuat beberapa endpoint sesuai kebutuhan.

Kebutuhan dasar yang akan di buat

1. Mendaftarkan perangkat API
2. Menghubungkan akun WA
3. Memutuskan Akun WA
4. Cek Status API
5. Membaca pesan
6. Mengirim pesan
7. Cek status pesan

Berdasarkan spesifikasi di atas akan di buat endpoint untuk setiap kebutuhan seperti pada tabel 4.1. berikut.

Tabel 4.1. Daftar Endpoint API

|  |  |  |
| --- | --- | --- |
| METHOD | ENDPOINT | KETERANGAN |
| POST | /devices | Mendaftarkan akun ke API |
| GET | /qr | Connect akun WA |
| DELETE | /devices | Disconnect akun WA |
| GET | /devices | Cek status API |
| GET | /messages | Membaca pesan masuk |
| POST | /messages | Kirim pesan pribadi |
| GET | /messages/{MESSAGE\_ID } | Cek status pesan |

Setiap endpoint harus memiliki respons untuk agar dapat di proses oleh client yang melakukan request ke API, dalam hal ini penulis melakukan dokumentasi menggunakan standar openAPI.

* + 1. Endpoint devices (/devices)

Pada endpoint ini dapat digunakan oleh 3 http method yaitu GET, POST, DELETE yang mana akan berfungsi sesuai dengan standar yang telah di buat. Sebelum dapat mengirim pesan menggunakan API maka perlu menambahkan device terlebih dahulu. Anda dapat memiliki lebih dari satu device pada API. Sebuah device harusnya terasosiasi dengan sebuah nomor WhatsApp.

POST Endpoint ini untuk menambahkan device baru yang bisa digunakan untuk pairing ke API.

GET Endpoint ini untuk melihat semua devices yang sudah dibuat oleh akun anda. Maksimum data yang dikembalikan per page adalah 10 item. Item terkahir yang dikembalikan per halaman ada pada properti last\_key. Properti ini kemudian dapat digunakan untuk merequest halaman berikutnya dengan menggunakan query string parameter start\_key. Jika nilai dari last\_key adalah null maka data sudah habis dan ini adalah halaman terakhir.

DELETE Endpoint ini digunakan untuk menghapus sebuah device berdasarkan DEVICE\_ID. Setelah device dihapus maka request berikutnya seperti mengirim pesan yang menggunakan DEVICE\_ID akan gagal.

* + 1. Endpoint QR (/qr)

Endpoint ini digunakan untuk mendapatkan QR code yang digunakan untuk melakukan pairing device ke AP. Terdapat dua atribut yang dikembalikan yaitu qr\_code dan image\_url. Untuk menampilkan gambar QR Code kunjungi link URL yang ada terdapat pada image\_url kemudian scan QR code tersebut dengan WhatsApp.

Jika terdapar WhatsApp Web maka session WhatsApp Web anda akan terputus setelah WhatsApp dipairing ke API. Hal ini karena server WhatsApp hanya memperbolehkan satu session WhatsApp Web yang aktif.

* + 1. Endpoint messages (/messages)

Saat ini API WhatsApp mendukung pengiriman pesan teks dan gambar. Pengiriman pesan pada API bersifat asynchronous. Artinya pesan yang dikirim akan masuk ke queue (antrian) server API Proses ini sangat cepat biasanya hanya memerlukan waktu beberapa detik untuk queue diproses. API mendukung pengiriman pesan ke nomor WhatsApp individu atau sebuah grup WhatsApp. Selain itu API juga dapat mengirimkan pesan secara terjadwal. Dimana pesan akan dikirim sesuai dengan tanggal dan waktu yang ditentukan.

POST /messages Endpoint ini digunakan untuk mengirimkan pesan baik teks, gambar, list, buttons atau dokumen (segala jenis file). Beberapa batasan untuk pengiriman pesan diantaranya: Pesan teks tidak lebih dari 8000 bytes atau 8KB. Untuk pesan bertipe image harus berformat JPEG, PNG atau GIF. Ukuran file gambar atau dokumen tidak boleh lebih dari 5.000.000 bytes atau 5MB. Timeout adalah 10 detik. Jika lebih dari itu maka pengiriman dianggap gagal dan koneksi akan diputus.

GET Endpoint ini digunakan untuk mendapatkan semua pesan yang dikirim dengan menggunakan endpoint POST /messages. Maksimum data yang dikembalikan per request adalah 10 item. Untuk merequest item berikutnya gunakan nilai dari atribut last\_key dan gunakan sebagai input untuk parameter start\_key. Jika nilai dari last\_key adalah null maka data sudah habis dan ini adalah halaman terakhir.

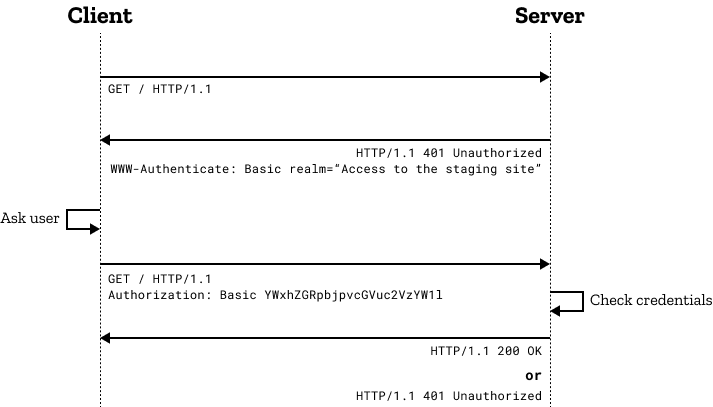
GET /messages/MESSAGE\_ID Endpoint ini digunakan untuk mendapatkan informasi sebuah message berdasarkan MESSAGE\_ID.

* + 1. Authorization

HTTP menyediakan kerangka kerja umum untuk kontrol akses dan autentikasi. Halaman ini adalah pengantar kerangka kerja HTTP untuk autentikasi, dan menunjukkan cara membatasi akses ke server Anda menggunakan skema "Basic" HTTP.

Alur tantangan dan respons bekerja seperti ini:

1. Server merespons klien dengan status respons 401 (Tidak Sah) dan memberikan informasi tentang cara mengotorisasi dengan header respons WWW-Authenticate yang berisi setidaknya satu tantangan.
2. Klien yang ingin mengautentikasi dirinya sendiri dengan server kemudian dapat melakukannya dengan menyertakan header permintaan Otorisasi dengan kredensial.
3. Biasanya klien akan memberikan permintaan kata sandi kepada pengguna dan kemudian akan mengeluarkan permintaan termasuk header Otorisasi yang benar.



Gambar 4.2. Flow HTTP Authorization

Header permintaan Otorisasi HTTP dapat digunakan untuk menyediakan kredensial yang mengautentikasi agen pengguna dengan server, memungkinkan akses ke sumber daya yang dilindungi.

Header Otorisasi biasanya, namun tidak selalu, dikirim setelah agen pengguna pertama kali mencoba meminta sumber daya yang dilindungi tanpa kredensial. Server merespons dengan pesan 401 Unauthorized yang menyertakan setidaknya satu header WWW-Authenticate. Header ini menunjukkan skema autentikasi apa yang dapat digunakan untuk mengakses sumber daya (dan informasi tambahan apa pun yang diperlukan oleh klien untuk menggunakannya). Agen pengguna harus memilih skema autentikasi paling aman yang didukungnya dari yang ditawarkan, meminta kredensial pengguna, lalu meminta ulang sumber daya (termasuk kredensial yang disandikan di header Otorisasi).

* 1. Hasil Peracangan Response API

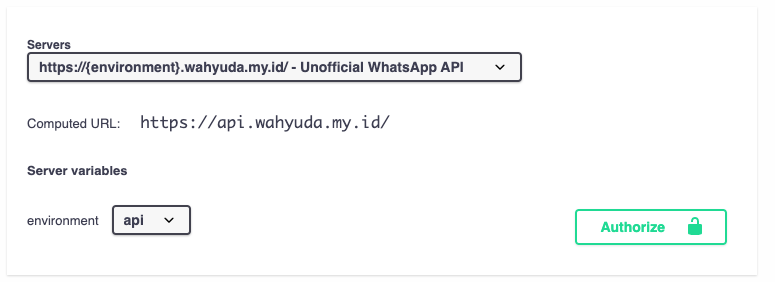
Setiap request yang berhasil dikirimkan akan mengembalikan response berupa json object yang akan di gunakan oleh client

* + 1. Dokumentasi Spesifikasi API

API yang akan di buat dokumentasi spesifikasi terlebih dahulu sehingga proses pengembangan akan lebih mudah dan dapat dikerjakan secara paralel.

1. API SERVER

|  |
| --- |
| "servers": [  {  "description": "Unofficial WhatsApp API",  "url": "https://{environment}.wahyuda.my.id",  "variables": {  "environment": {  "description": "Server Environment",  "default": "api",  "enum": ["dev", "test", "api"]  }  }  }  ], |

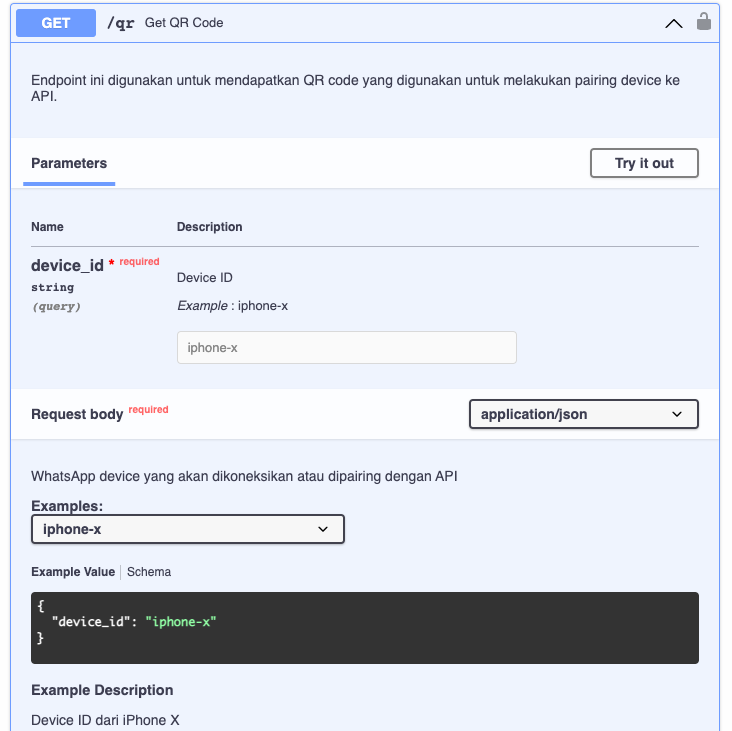


Gambar 4.3. URL Server API

Untuk dapat di akses oleh banyak penguna maka server harus terhubung ke internet, kemudian agar mempermudah dalam penggunaan server diberikan domain yang akan mengarahkan ke ip pada server.

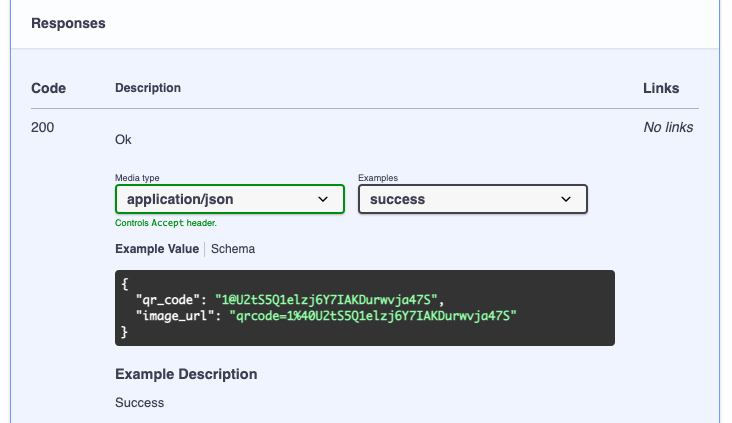
1. Endpoint QR

Pada endpoint ini memiliki beberapa parameter yang akan digunakan sebagai http request dan response, untuk malakukan pairing dengan akun whatsapp dipelukan QR Code dan untuk mendapatkannya pengguna harus melakukan http request dengan parameter *device\_id* pada bagian query url dengan contoh seperti berikut: *https://api.wahyuda.my.id/v1/qr?device\_id=DEVICE\_ID*



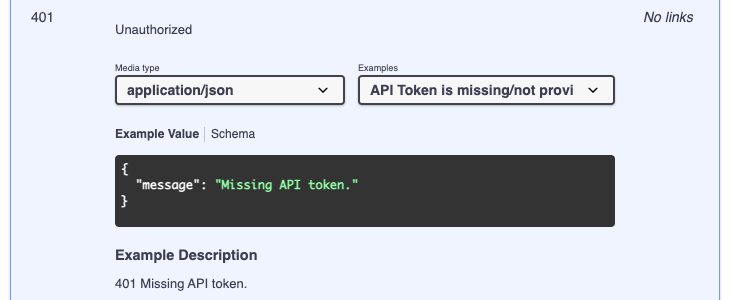
Gambar 4.4. Header parameter GET QR Code

Setelah mengirimkan request, server akan melakukan beberapa validasi berupa device ide yang di kirimkan, jika device id telah terdaftar maka server akan mengembalikan respons code 200 seperti gambar 4.5. di bawah.

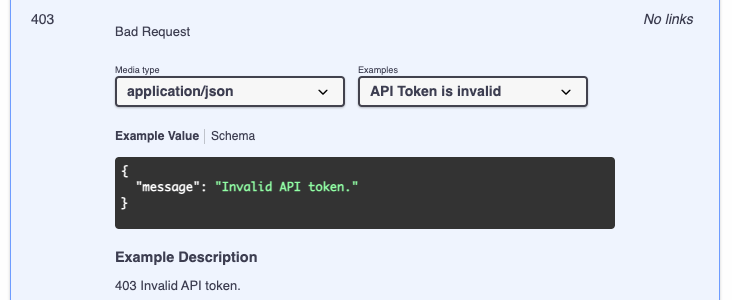


Gambar 4.5. Respose Code 200 GET QR Code

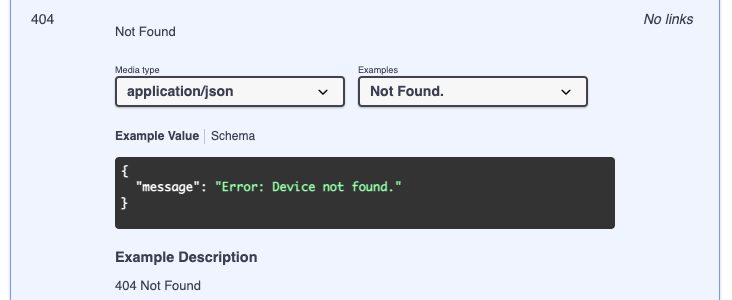
Server akan mengirimkan response code 4xx jika data yang di kirimkan pengguna tidak sesuai ataupun tidak ditemukan.



Gambar 4.6. GET QR Code Token tidak ditemukan



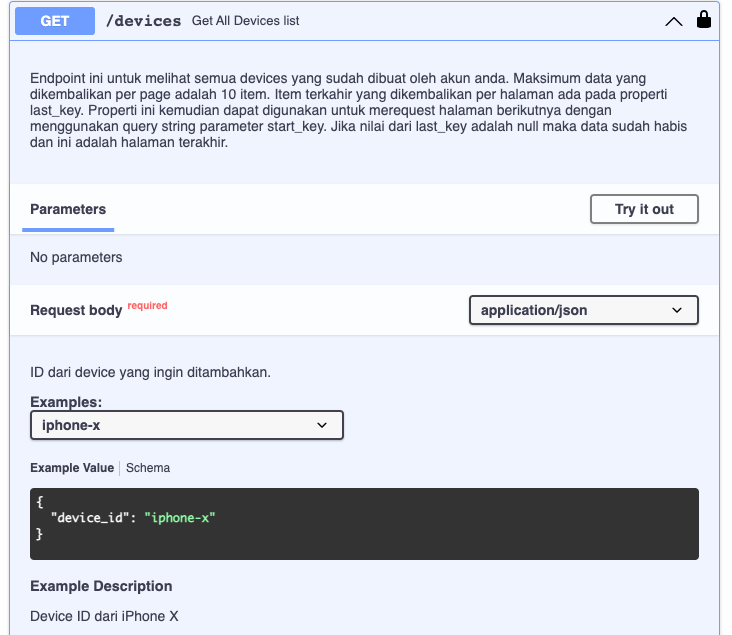
Gambar 4.7. GET QR Code Token yang dikirimkan Salah



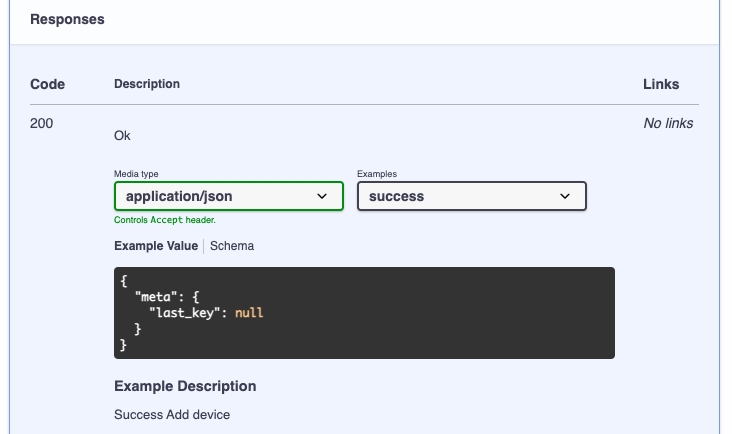
Gambar 4.8. GET QR Code device tidak ditemukan

1. DEVICES – GET

Untuk mendapatkan status dari device pengguna harus mengirimkan request ke endpoind device. Pada endpoint ini server akan mengembalikan status semua device yang telah didaftarkan.



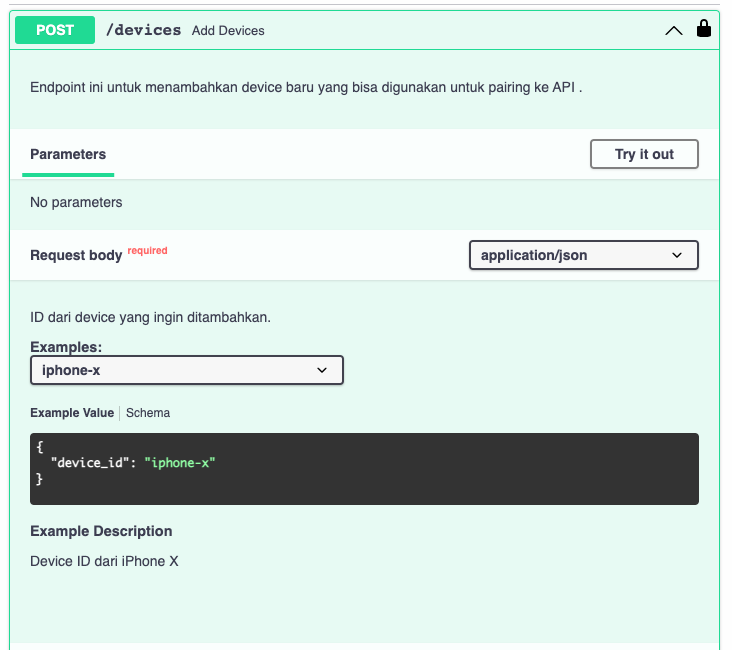
Gambar 4.9. GET DEVICES Request Parameter



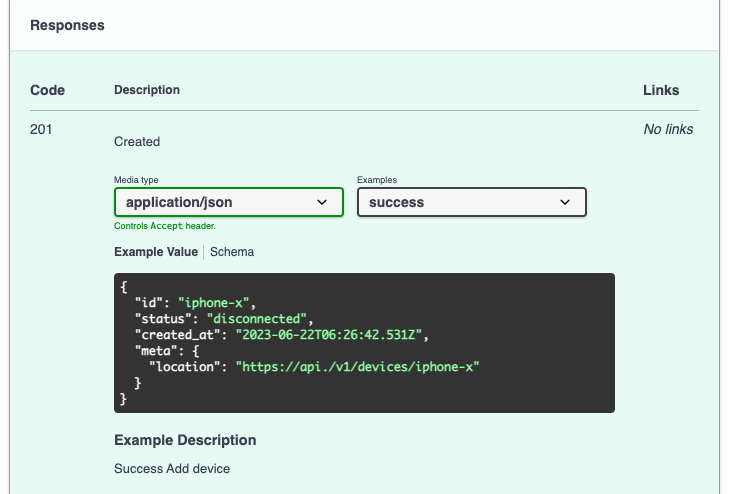
Gambar 4.10.GET DEVICE Response 200 OK

1. DEVICES – POST

Selain melakukan pairing akun whatsapp mengguna perlu mendaftarkan perangkat sebagai id/pengenal pada akun whatsapp nya. Pengguna diharuskan mengirimkan device\_id yang ingin di daftarkan.

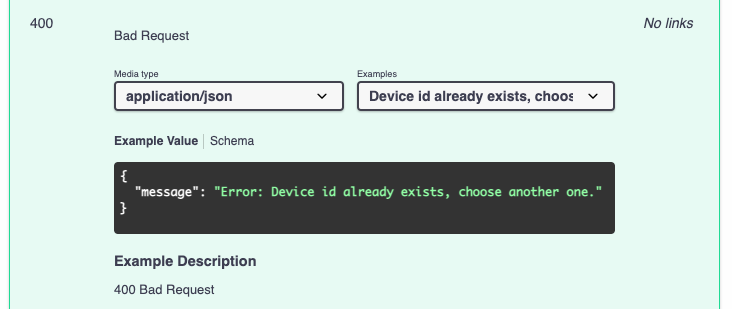


Gambar 4.11. POST DEVICES Request Parameter



Gambar 4.12. POST DEVICE Response 200 OK

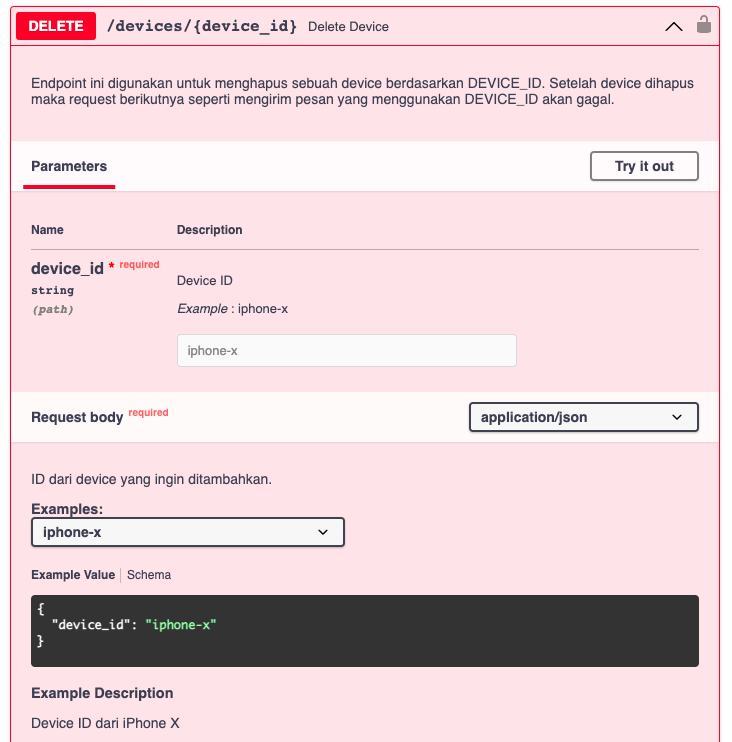
Apabila device\_id yang dikirimkan dalam http request telah terdaftar, maka server akan mengembalikan RC 400 bad request.



Gambar 4.13. POST DEVICE Response 400 Bad Request

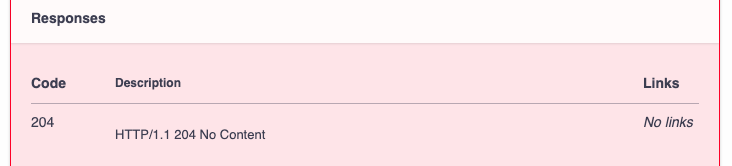
1. DEVICES – DELETE

Apabila pengguna memiliki kesalahan dalam mendaftarkan device ataupun ingin menghapus device nya karena sudah tidak digunakan maka method yang dikirimkan adalah DELETE dan dikuti oleh device\_id yang ingin di hapus.



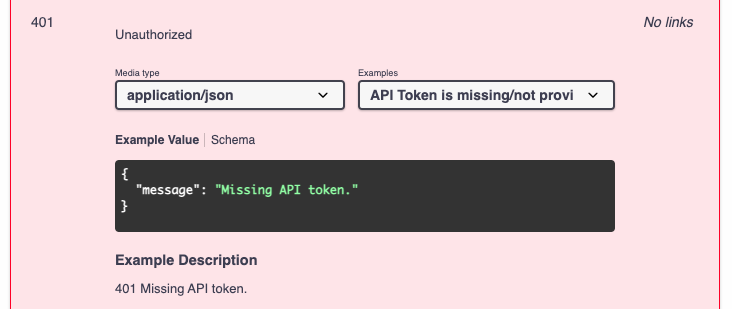
Gambar 4.14. DELETE DEVICES Request Paramater

Apabila device berhasil dihapus server akan mengembalikan RC 204 bukan 200. Hal ini dilakukan untuk membedakan response dari permintaan menambahkan device dan menghapus device.



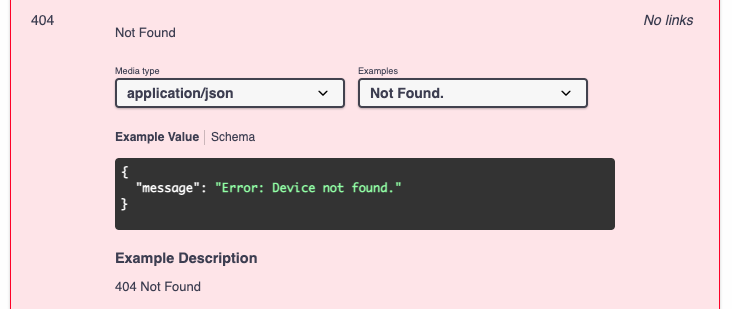
Gambar 4.15. DELETE DEVICES Response 204 No Content

Jika pengguna yang tidak memiliki hak untuk melakukan penghapusan device maka server akan mengembalikan RC 401 Unauthorize yang berarti HTTP menetapkan "tidak sah", secara semantik respons ini berarti "tidak diautentikasi". Artinya, klien harus mengautentikasi dirinya sendiri untuk mendapatkan respons yang diminta.



Gambar 4.16. DELETE DEVICE RC 401 Unauthorized

Namun jika device yang ingin di hapus tidak ditemukan Server akan mengembalikan RC 404 Not Found, hal ini terjadi jika device memang tidak terdaftar maupun sudah di hapus sebelumnya, Server tidak dapat menemukan sumber daya yang diminta. Dalam API, ini juga bisa berarti bahwa titik akhir valid tetapi sumber dayanya sendiri tidak ada. Kode respons ini mungkin yang paling terkenal karena sering muncul di web.



Gambar 4.17. DELETE DEVICE RC 404 Not Found

1. MESSAGES – POST
2. WEBHOOK – POST

Saat menerima pesan whatsapp, secara default api akan membuat beberapa parameter dari pesan yang di terima seperti pada tabel 4.2. di bawah ini untuk mengidentifikasi setiap pesan yang diterima.

Tabel 4.2. Parameter Webhook

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter |  | Tipe data | contoh |
| id | | String | 2605630cf636463da5ce2f4110a |
| Webhook\_type | | String | incoming\_message |
| payload | | Object |  |
|  | device\_id | String | iphone-x-pro |
| sender | String | 6281234567890 |
| from\_me | Boolean | False |
| message\_type | String | Text |
| text | String | Halo ini pesan dari api |
| timestamp | Int | 1631306956 |
| created\_at | | Timestamp | 2023-07-10T20:49:16.000Z |
| server\_time | | Timestamp | 2023-07-10T20:49:17.462Z |

Berikut ini merupakan contoh hasil dari tabel 4.2. dalam bentuk json yang di buat oleh server.

|  |
| --- |
| {  "id": "2605630cf636463da5ce2f4110a",  "webhook\_type": "incoming\_message",  "payload": {  "device\_id": "iphone-x-pro",  "sender": "6281234567890",  "from\_me": false,  "message\_type": "text",  "text": "Halo ini pesan dari api",  "timestamp": 1631306956  },  "created\_at": "2023-07-10T20:49:16.000Z",  "server\_time": "2023-07-10T20:49:17.462Z"  } |

* + 1. Response Code

Pada API setiap request yang berhasil akan mengembalikan response HTTP code 2XX. Sedangkan jika terjadi error maka HTTP code yang dikembalikan adalah 4XX atau 5XX.

Tabel 4.3. Success Code dasar

|  |  |
| --- | --- |
| Response  Code | Keterangan |
| 200 | OK/ Permintaan berhasil. "sukses" tergantung pada metode HTTP:  GET: Sumber daya telah diambil dan dikirim dalam isi pesan.  POST: Sumber daya yang menjelaskan hasil tindakan ditransmisikan dalam *message body*. |
| 201 | Created/Permintaan berhasil, dan sumber daya baru dibuat. respons ini dikirim setelah permintaan POST. |
| 204 | No Content/ Kode respons status keberhasilan HTTP 204 Tidak Ada Konten menunjukkan bahwa permintaan telah berhasil, tetapi klien tidak perlu keluar dari halamannya saat ini. |

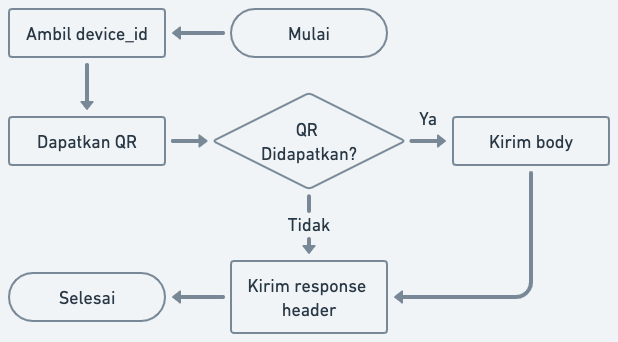
Tabel 4.4. Error Code dasar

|  |  |
| --- | --- |
| Response  Code | Keterangan |
| 400 | Bad Request/Request yang dilakukan tidak sesuai. |
| 401 | Unauthorized/API Token salah. |
| 403 | Forbidden/Tidak ada izin akses untuk URL yang diminta. |
| 404 | Not Found/Tidak ada halaman yang dimaksud. |
| 429 | Too Many Requests /Jumlah request yang dilakukan terlalu banyak. |
| 5xx | Internal Server Error/Terjadi kesalahan pada server API |

* 1. Pembuatan Code untuk API
     1. QR Code

Saat melakukan permintaan QR Code server menggunakan script seperti di bawah ini dengan alur di ilustrasikan pada gambar

|  |
| --- |
| import 'dotenv/config';  import { apiRequest } from '../services/services.js';  const getQRCode = async (req, res) => {  const device\_id = req.query.device\_id;  const url = new URL(process.env.API\_SERVER + '/qr?device\_id=' + device\_id);  const reqParams = {  token: process.env.API\_TOKEN,  url: url,  };  try {  const { body, response } = await apiRequest(reqParams);  console.log(body);  res.status(response.statusCode).json(JSON.parse(body));  } catch (error) {  res.status(error.response.statusCode).json(JSON.parse(error.body));  }  };  export default getQRCode; |



Gambar 4.22. Flowchart QR Code

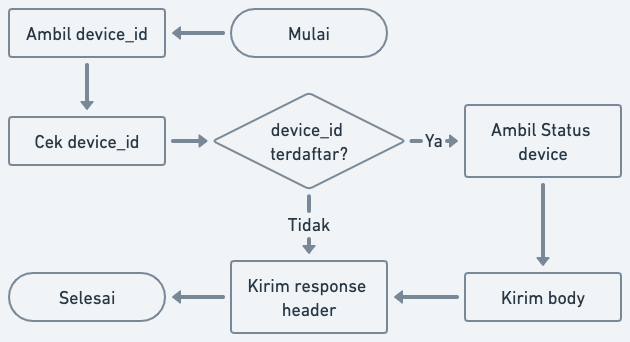
Ketika pengguna melakukan request untuk mendapatkan QR Code, server akan mengambil device\_id kemudian melakukan generate QR Code, apabila QR gagal di dapatkan server hanya akan mengirimkan response tanpa body yang berisi error code. Jika QR berhasil didapatkan server akan mengirimkan QR Code dalam encode base64, hal ini dilakukan karena transmisi menggunakan json yang memiliki keterbatasan yang hanya dapat mengirimkan data alfanumerik dan simbol. Namun server juga mengirimkan url qrcode yang dapat di scan oleh pengguna nantinya.

* + 1. Devices

Pada endpoint device akan digunakan untuk segala hal yang berhubungan dengan device pada api ini, setiap device hanya dapat terhubung ke satu akun whatsapp. Nama device bersifat unik sehingga setiap pengguna harus menentukan nama device yang berbeda saat ingin menambahkan device menggunakan method POST. Device juga dapat di hapus apabila sudah tidak di gunakan lagi ataupun ingin menonaktifkannya.

1. GET

|  |
| --- |
| const getDevices = async (req, res) => {  const device\_id = req.params.device\_id;  const url = new URL(process.env.API\_SERVER + '/devices/' + device\_id);  const reqParams = {  token: process.env.API\_TOKEN,  url: url,  };  try {  const { body, response } = await apiRequest(reqParams);  res.status(response.statusCode).json(JSON.parse(body));  } catch (error) {  console.error('Something went wrong');  res.status(error.response.statusCode).json(JSON.parse(error.body));  }  }; |

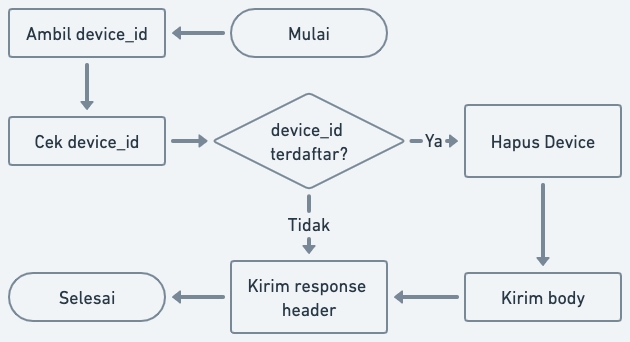


Gambar 4.23. Flowchart GET Device Status

Ketika pengguna melakukan request untuk melihat apakah device terdaftar, server akan mengambil *device\_id* kemudian melakukan generate Device Status, apabila device gagal di dapatkan server hanya akan mengirimkan response tanpa body yang berisi error code. Jika berhasil didapatkan maka server akan mengembalikan RC 400 bad request, hal ini dilakukan karena transmisi menggunakan json yang memiliki keterbatasan yang hanya dapat mengirimkan data alfanumerik dan simbol.

1. DELETE

|  |
| --- |
| const deleteDevices = async (req, res) => {  const device\_id = req.params.device\_id;  const url = new URL(process.env.API\_SERVER + '/devices/' + device\_id);  const reqParams = {  token: process.env.API\_TOKEN,  url: url,  method: 'DELETE',  };  try {  const { response } = await apiRequest(reqParams);  console.log(device\_id + ' Deleted');  res.status(response.statusCode).end();  } catch (error) {  console.error('Something went wrong');  res.status(error.response.statusCode).json(JSON.parse(error.body));  }  }; |

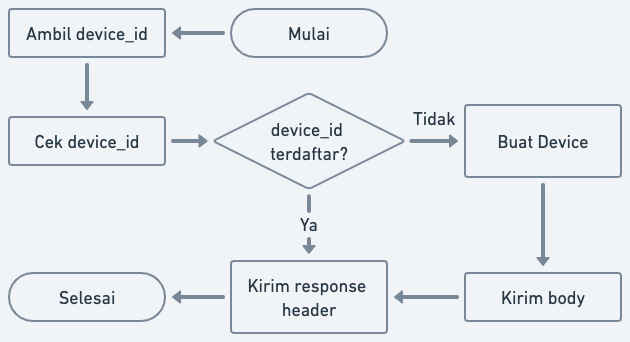


Gambar 4.24. Flowchart Hapus Device

Ketika pengguna melakukan request untuk menghapus device, server akan mengambil device\_id kemudian melakukan generate Hapus Device, apabila hapus device gagal di dapatkan server hanya akan mengirimkan response tanpa body yang berisi error code. Jika berhasil dihapus server akan mengembalikan RC 204 bukan 200, hal ini dilakukan karena transmisi menggunakan json yang memiliki keterbatasan yang hanya dapat mengirimkan data alfanumerik dan simbol.

1. POST

|  |
| --- |
| const addDevices = async (req, res) => {  const device\_id = req.body.device\_id;  const url = new URL(process.env.API\_SERVER + '/devices');  const reqParams = {  token: process.env.API\_TOKEN,  url: url,  method: 'POST',  payload: JSON.stringify({  device\_id: device\_id,  }),  };  try {  const { body, response } = await apiRequest(reqParams);  console.log(body);  res.status(response.statusCode).json(JSON.parse(body));  } catch (error) {  console.error('Something went wrong');  res.status(error.response.statusCode).json(JSON.parse(error.body));  }  };  export { addDevices, getAllDevices, getDevices, deleteDevices }; |

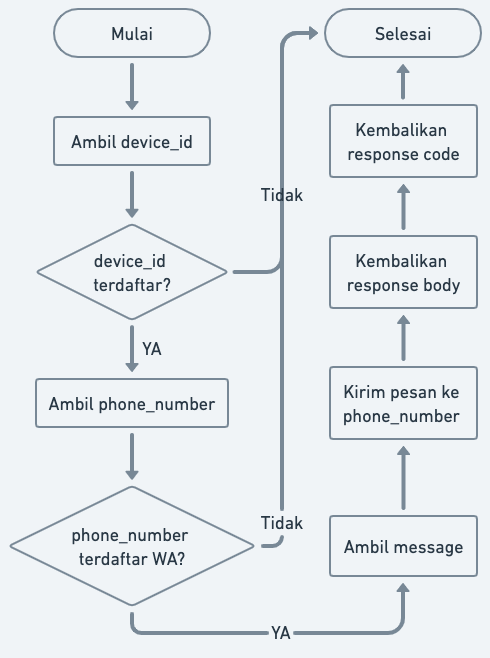


Gambar 4.25. Flowchart Tambah Device Baru

Ketika pengguna melakukan request untuk daftar Device, server akan mengambil device\_id kemudian melakukan generate Buat Device, apabila Buat Device gagal atau telah terdaftar di dapatkan server hanya akan mengirimkan response tanpa body yang berisi error RC 400 bad Request. Jika zbuat Device berhasil didaftarkan server akan mengirimkan Response 200 OK, hal ini dilakukan karena transmisi menggunakan json yang memiliki keterbatasan yang hanya dapat mengirimkan data alfanumerik dan simbol.

* + 1. Message

|  |
| --- |
| import { apiRequest } from '../services/services.js';  const sendMessage = async (req, res) => {  const url = new URL(process.env.API\_SERVER + '/messages');  const message = {  message: 'Halo ini adalah pesan dari api',  phone\_number: '6281234567890',  message\_type: 'text',  device\_id: 'iphone-x-pro',  };  const reqParams = {  token: process.env.API\_TOKEN,  url: url,  method: 'POST',  payload: JSON.stringify(message),  };  try {  const { body, response } = await apiRequest(reqParams);  res.status(response.statusCode).json(JSON.parse(body));  } catch (error) {  res.status(error.response.statusCode).json(JSON.parse(error.body));  }  };  export { sendMessage }; |

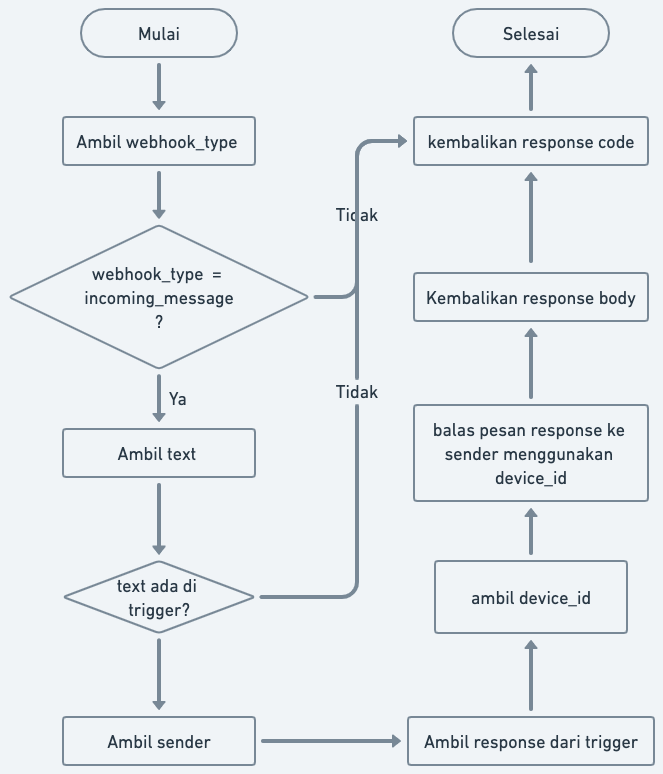


Gambar 4.26. Flowchart Kirim Pesan

Ketika pengguna melakukan request maka yang dilakukan pengecekan device terftar atau tidak, jika tidak maka langsung mengembalikan response, jika terdaftar maka lanjut pengecekan nomor apakah terdaftar jika tidak server akan mengembalikan RC 400 atau 404, apabila terdaftar maka server ngambil pesan dan mengirim response ke nomor telepon dalam bentuk response body lalu response code dan selesai. Hal ini dilakukan karena transmisi menggunakan json yang memiliki keterbatasan yang hanya dapat mengirimkan data alfanumerik dan simbol.

* + 1. Webhook

|  |
| --- |
| import 'dotenv/config';  import { logger } from '../app/logging.js';  import { prismaClient } from '../app/database.js';  import { apiRequest } from '../services/services.js';  const webhookIncomingMessage = async (req, res) => {  const body = req.body;  const payload = body.payload;  const text = payload.text;  const sender = payload.sender;  const device\_id = payload.device\_id;  const timestamp = payload.timestamp;  const webhook\_type = body.webhook\_type;  const message\_type = payload.message\_type;  if (webhook\_type === 'incoming\_message') {  const check = await prismaClient.message.findFirst({  where: {  trigger: text,  },  select: {  state: true,  response: true,  hardwareId: true,  },  });  if (check === null) {  res.status(204).end();  } else {  const data = {  phone\_number: sender,  message: check.response,  device\_id: device\_id,  message\_type: message\_type,  };  await prismaClient.log.create({  data: {  text: text,  sender: sender,  time: timestamp,  device\_id: device\_id,  },  });  if (check.hardwareId != null && check.state != null) {  await prismaClient.hardware.update({  where: {  id: check.hardwareId,  },  data: {  state: check.state,  },  });  }  const url = new URL(process.env.API\_SERVER + '/messages');  const reqParams = {  token: process.env.API\_TOKEN,  url: url,  method: 'POST',  payload: JSON.stringify(data),  };  try {  const { body, response } = await apiRequest(reqParams);  res.status(response.statusCode).json(JSON.parse(body));  } catch (error) {  res.status(400).json(JSON.parse(error.body));  }  }  } else {  logger.error({ message: 'Unknown webhook type of ' + webhook\_type });  res.status(400).end();  }  };  export { webhookIncomingMessage }; |



Gambar 4.26. Flowchart pesan masuk

Ketika API menerima pesan maka yang dilakukan identifikasi melalui webhook jika tidak sesuai server akan mengembalikan RC 400 atau 404, jika sesuai maka pesan akan di proses apakah text terdafrar dalam database jika tidak maka server akan mengembalikan RC 400 atau 400, jika terdaftar maka lanjut ambil response sesuai permintaan text dan balas pesan menggunakan device\_id dan dikembalikan dalam bentuk response body. Hal ini dilakukan karena transmisi menggunakan json yang memiliki keterbatasan yang hanya dapat mengirimkan data alfanumerik dan simbol.

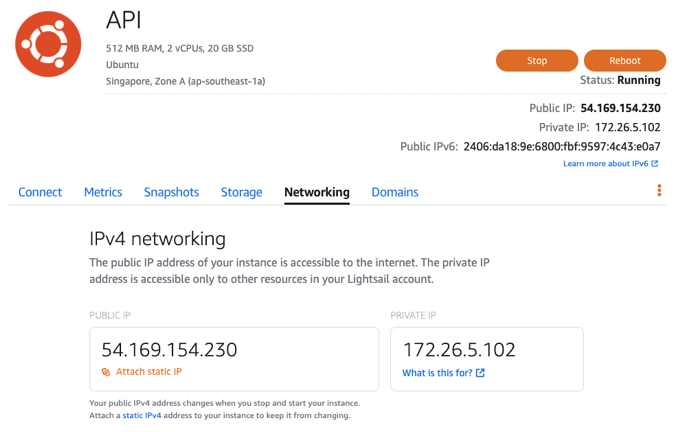
* 1. Konfigurasi Server API

API yang di buat tidak akan berjalan tanpa adanya server, oleh karena itu penulis akan menjalankan API pada server yang terhubung ke internet dengan spesifikasi sebagai berikut.

Tabel 4.x. Spesifikasi Server

|  |  |
| --- | --- |
| Keterangan | Spesifikasi |
| Sistem Operasi | Ubuntu 22.04 |
| Memory | 512 MB |
| Processing | 2 vCPUs |
| Penyimpanan | 20 GB |
| NodeJS | v18.16.1 |
| NPM | v9.5.1 |
| expressJS | v4.18.2 |
| Database | MySQL v8.0.31 |

Server yang terhubung memiliki alamat IP public seperti gambar 4.x. dibawah, penulis juga akan menghubungkan alamat IP tersebut ke domain agar lebih mudah digunakan.

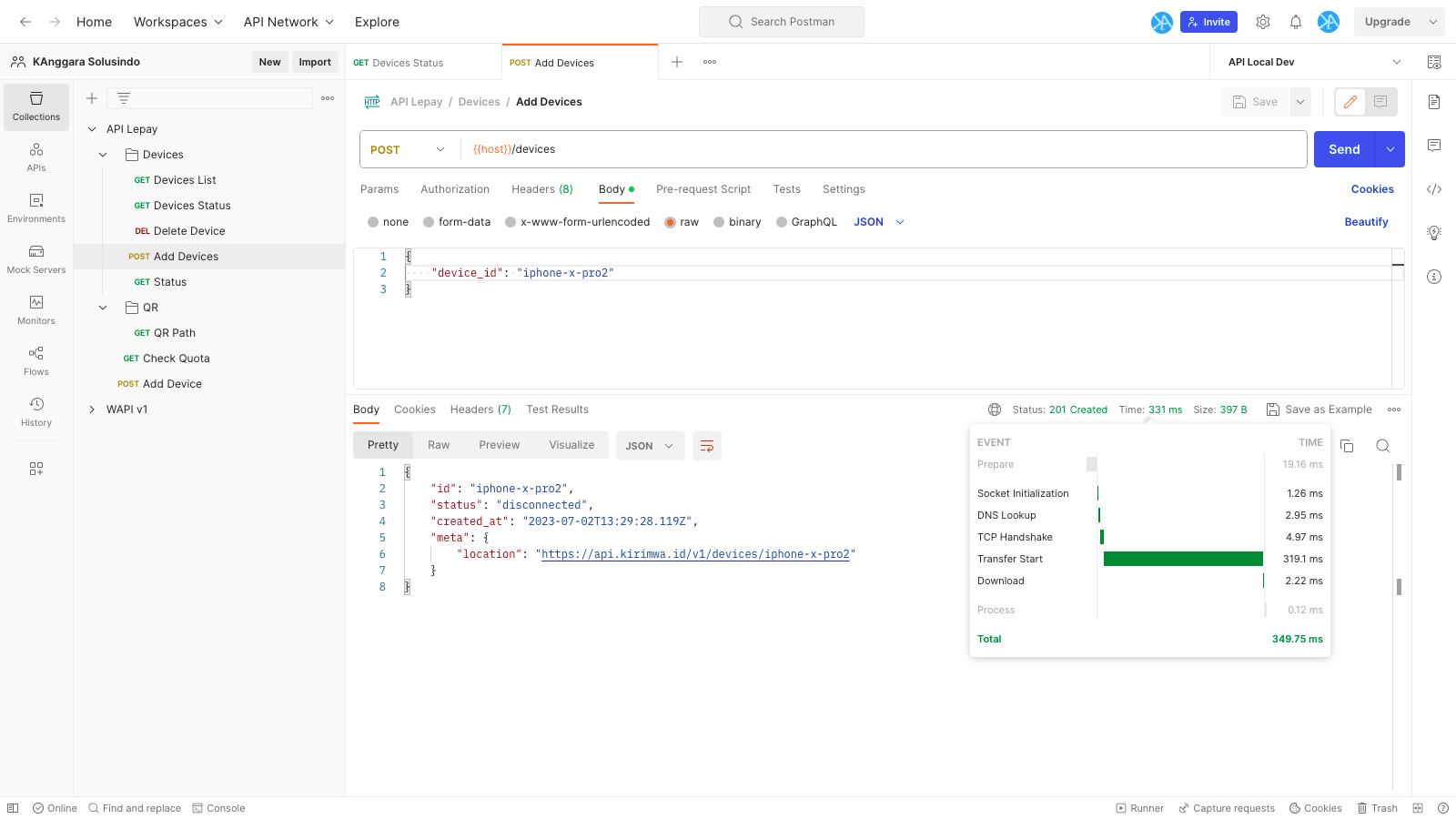


Gambar 4.x. Spesifikasi Server API

* 1. Pengujian API

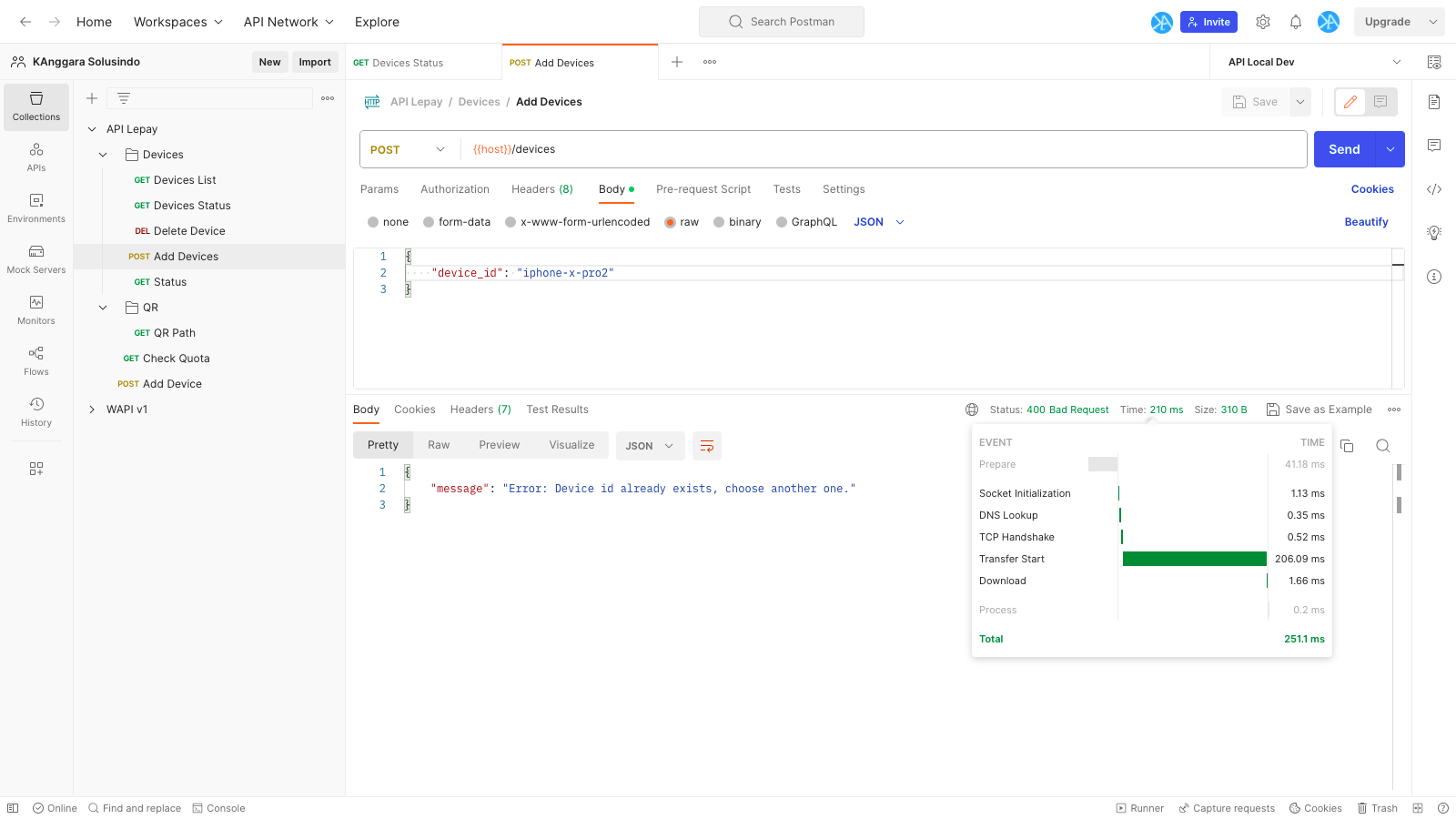
Setelah API berhasil di buat dan dijalankan di server maka akan dilakukan proses pengujian untuk mendapatkan data berupa hasil dari pengujian yang dilakukan. Pengujian yang dilakukan untuk setiap endpoin sebanyak 30 kali dengan response code berhasil atau 2xx.

* + 1. Add Device



Gambar 4.x. Pengujian POST Device RC 201

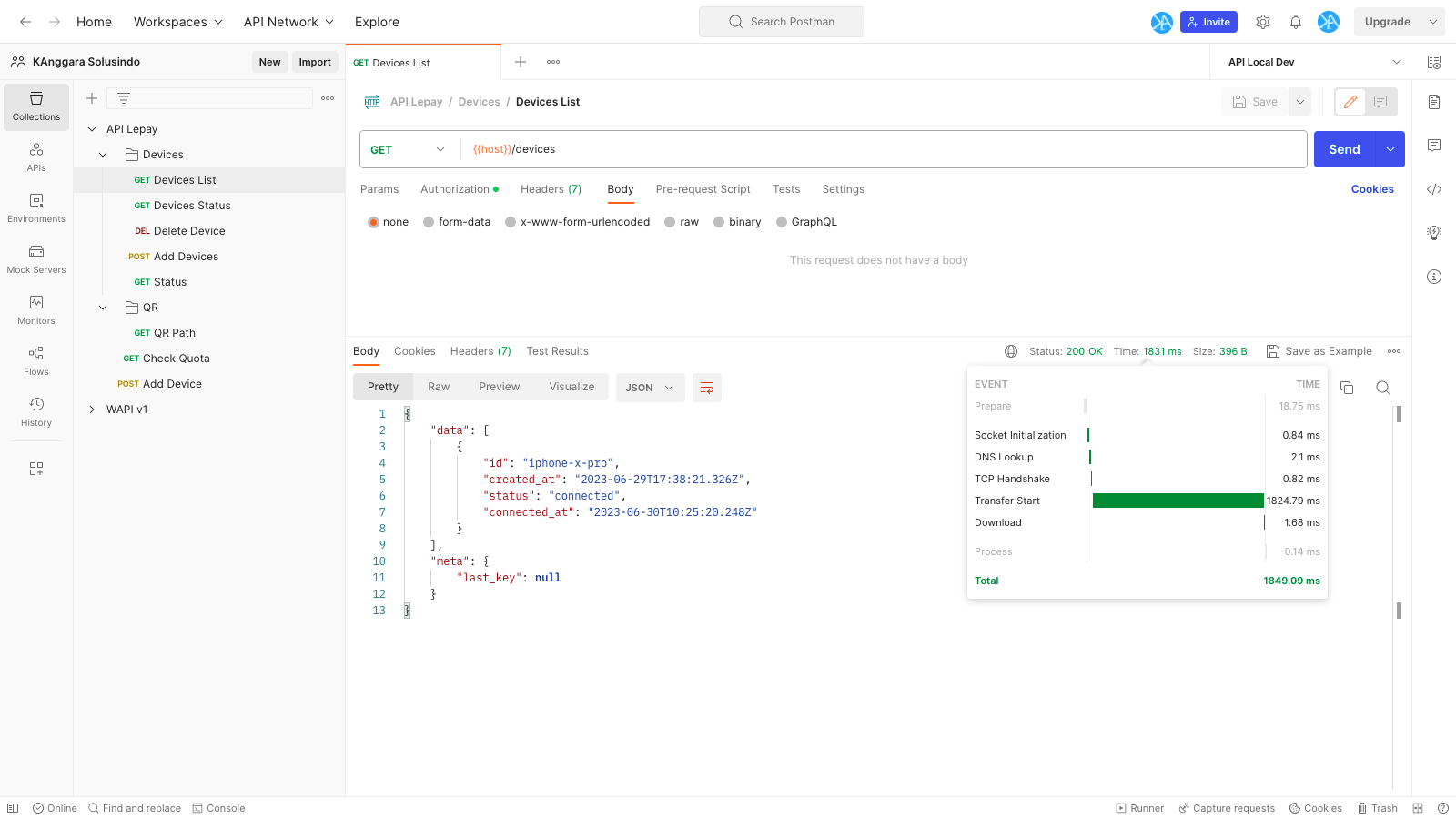
Jika device\_id yang akan di daftarkan belum tersedia server akan mengirimkan RC 201yang berarti permintaan yang dikirimkan berhasil dan dibuat.



Gambar 4.x. Pengujian POST Device RC 400

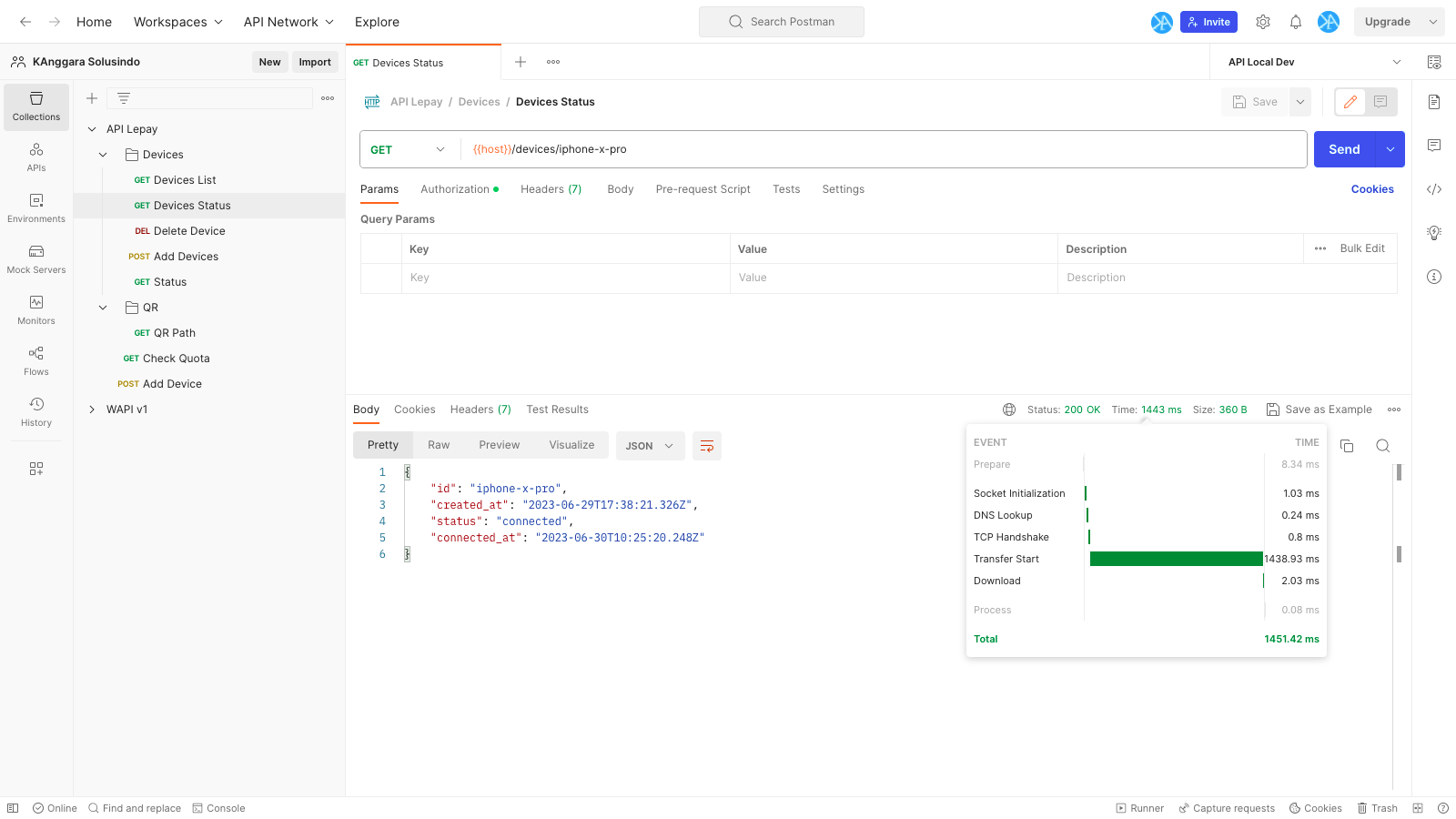
Saat melakukan request yang sama dengan request sebelumnya yang mendapatkan RC 201 server akan mengirimkan RC 400 karena device\_id yang ingin didaftarkan telah ada.

* + 1. Get Device List

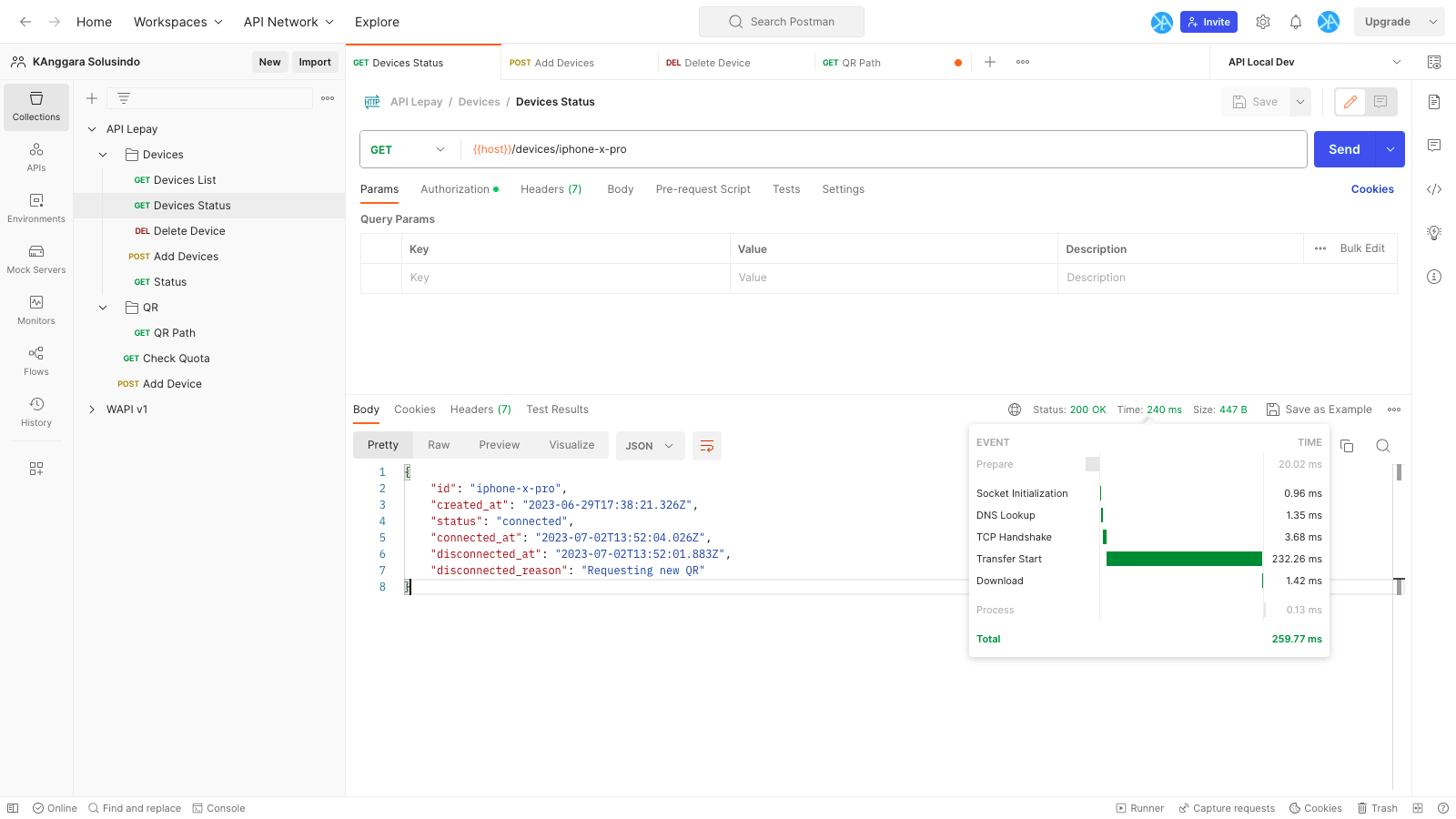


Gambar 4.x. Pengujian GET Device list RC 200

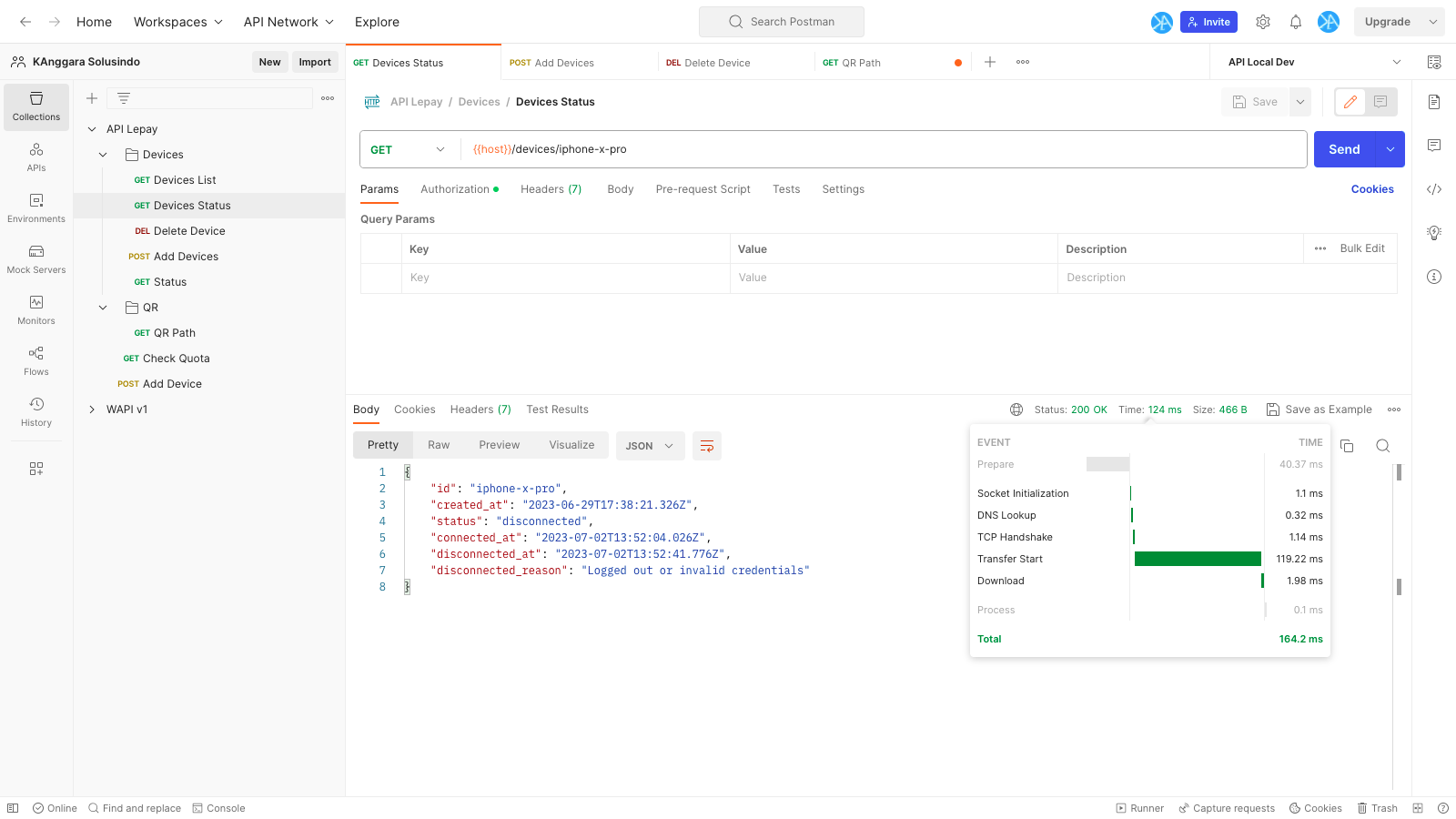
* + 1. Get Device Status



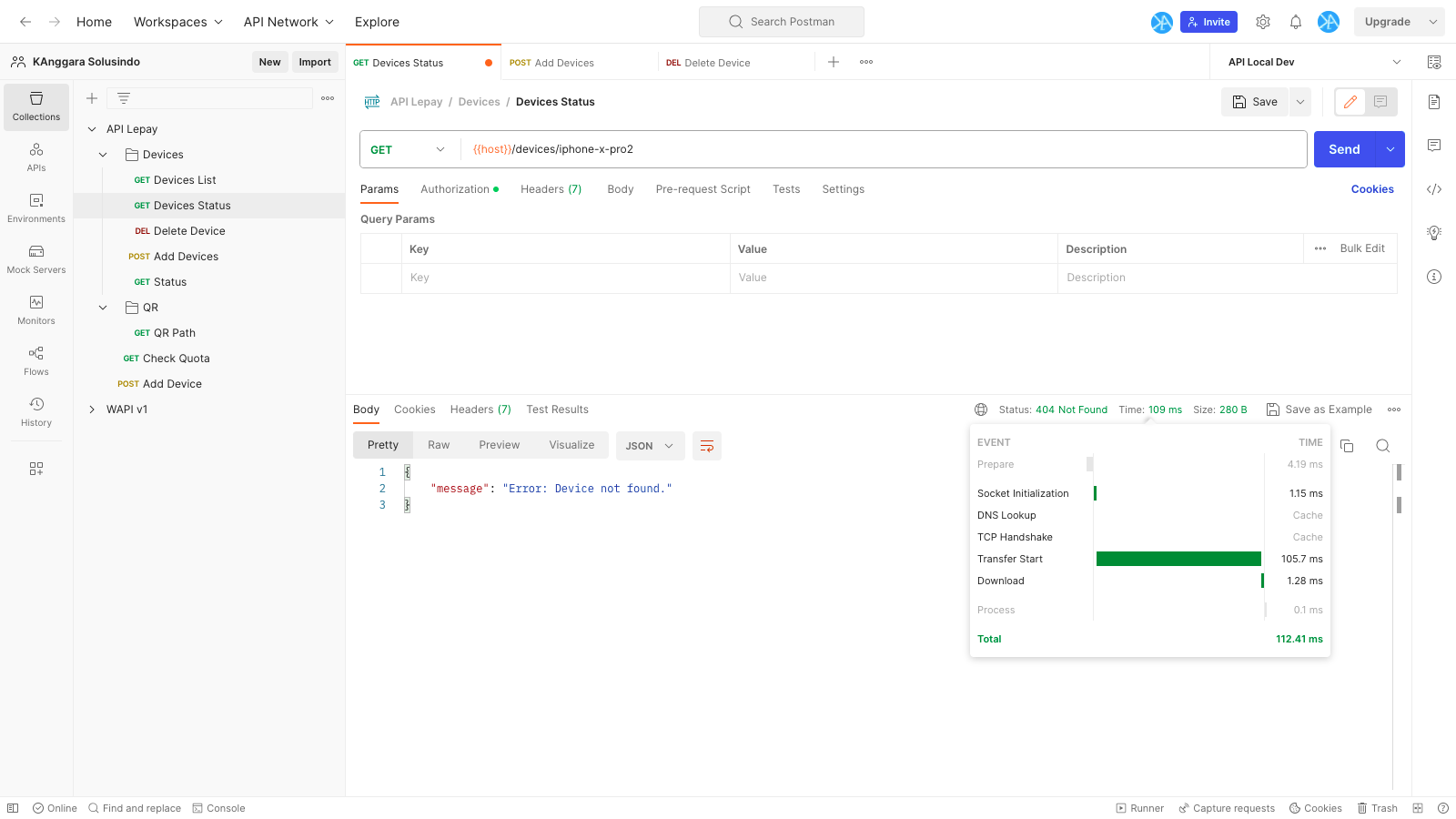
Gambar 4.x. Pengujian GET Device Status connected RC 200



Gambar 4.x. Pengujian GET Device Status RC 200 request new QR

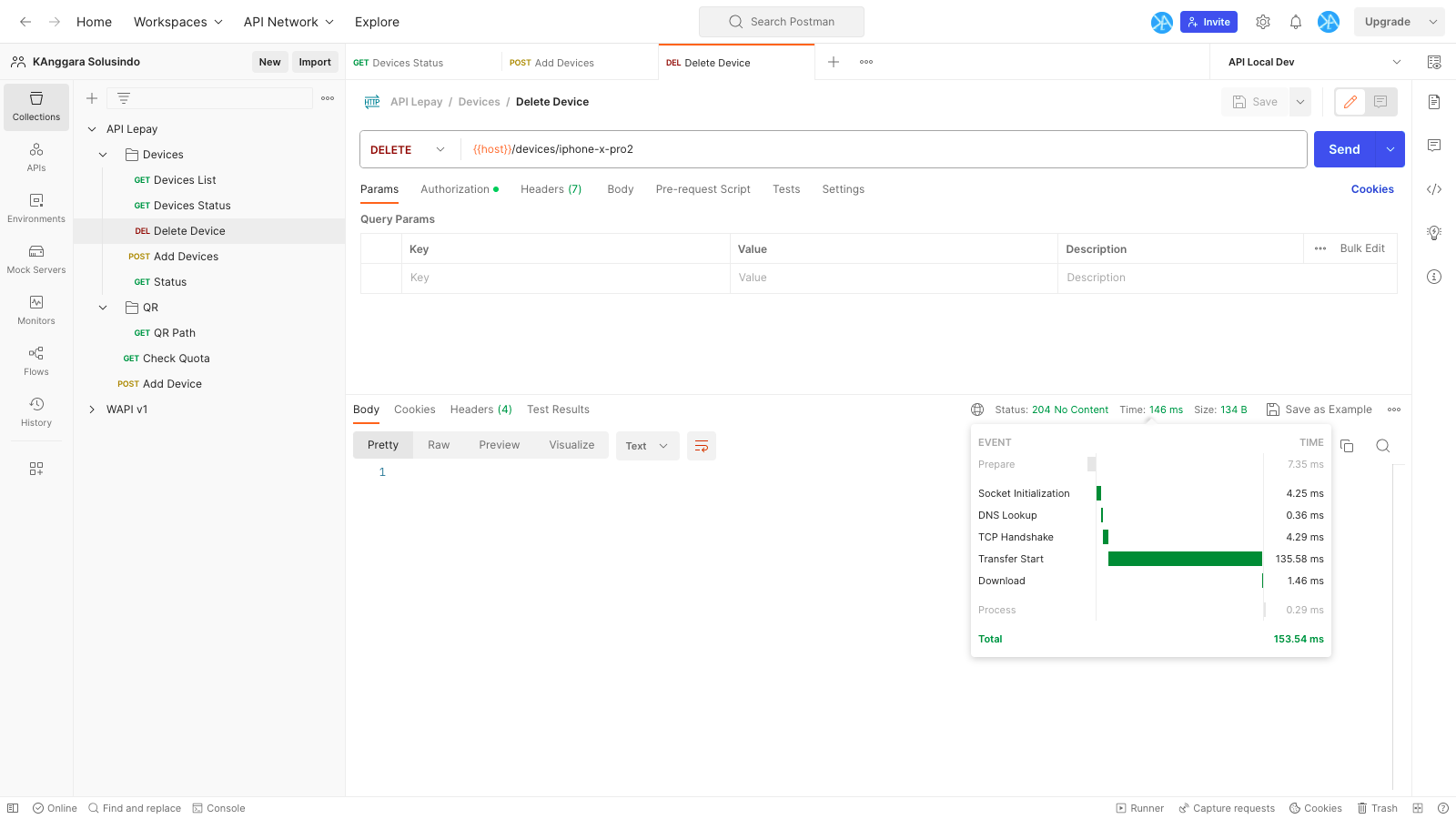


Gambar 4.x. Pengujian GET Device Status disconnect RC 200

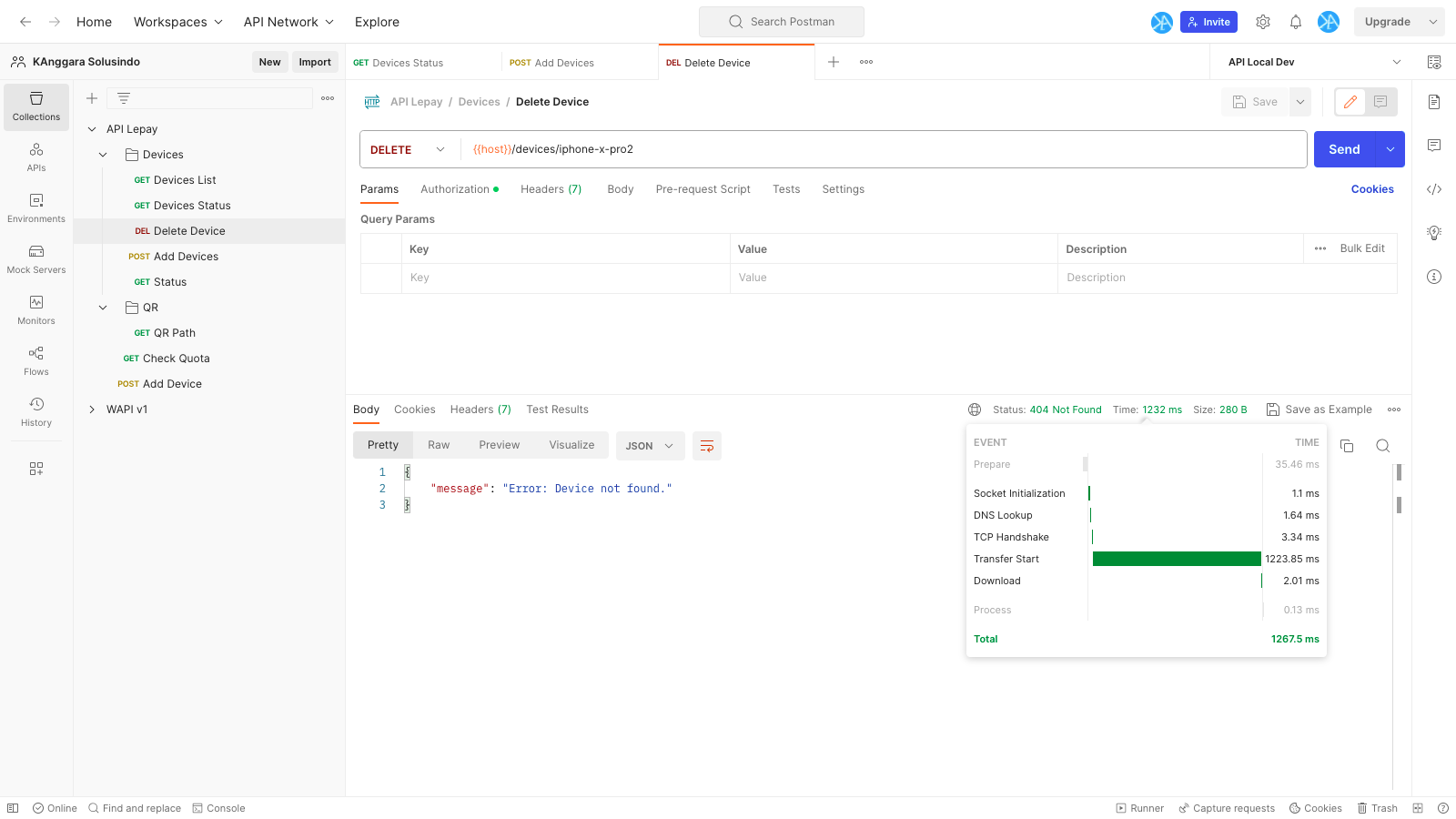


Gambar 4.x. Pengujian GET Device Status RC 404 Not Found

* + 1. Delete Device

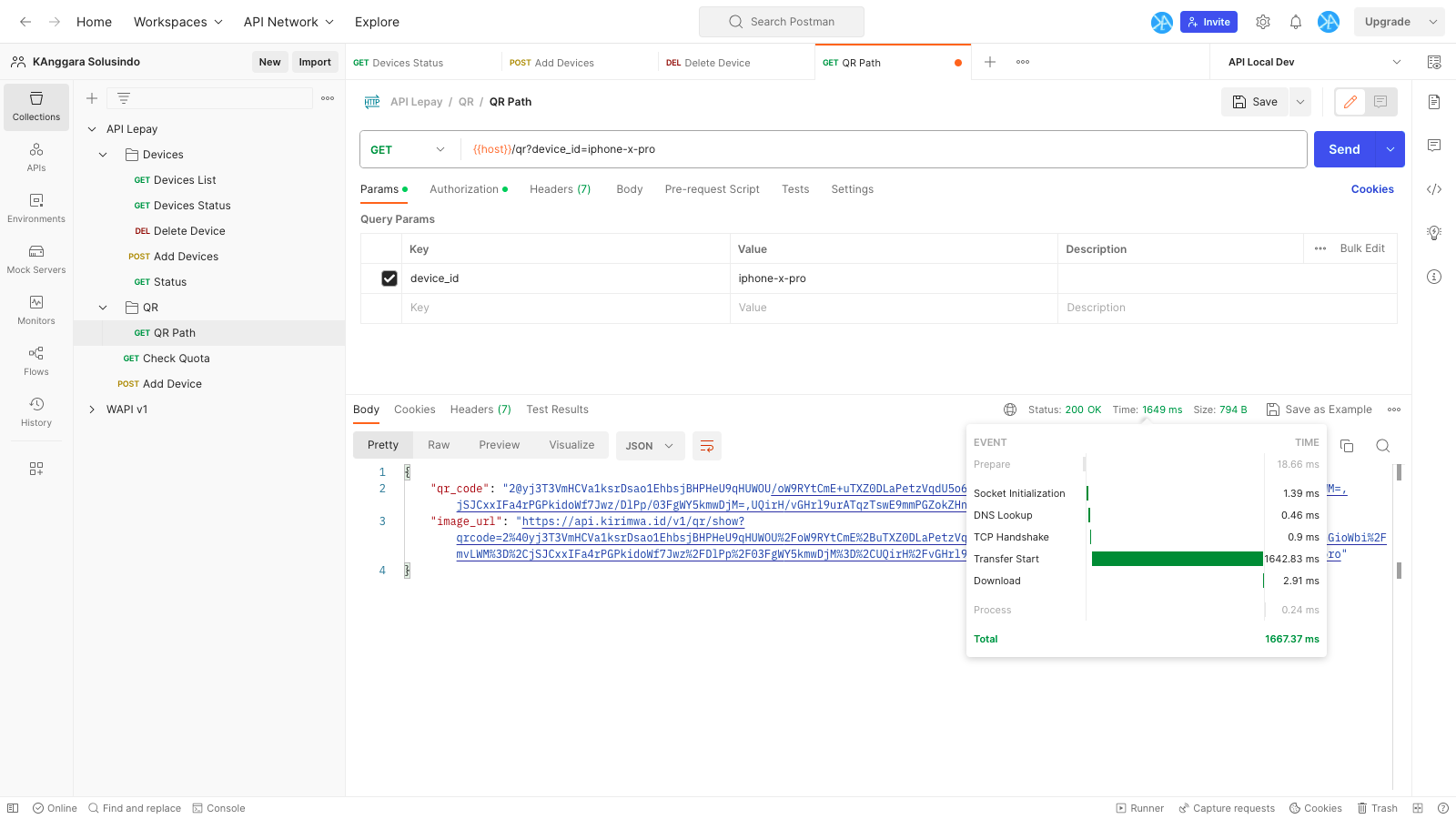


Gambar 4.x. Pengujian Delete Device RC 204 No Content



Gambar 4.x. Pengujian Delete Device RC 404 Not Found

* + 1. Get QR Code



Gambar 4.x. Pengujian Get QR RC 200



Gambar 4.x. Tampilan hasil GET QR pada browser

* + 1. Pengujian Respons time API

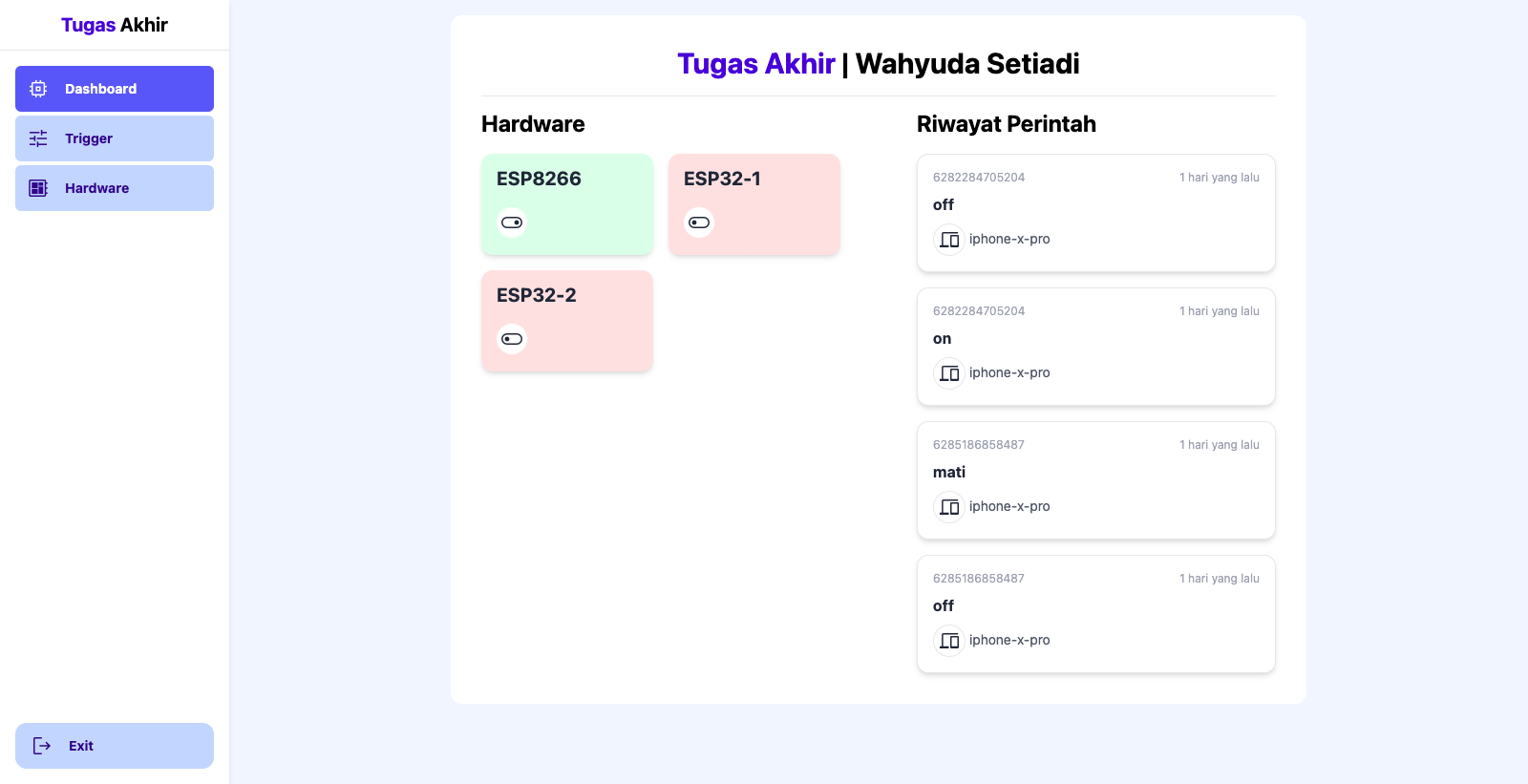
Untuk mengetahui kinerja dari API pada setiap request yang dikirimkan oleh pengguna penulis menggunakan aplikasi Postman, hal ini dilakukan karena pada browser hanya mendukung method GET. Dalam melakukan pengujian ini melakukan permintaan ke server sebanyak 30 kali untuk setiap endpoint.

Tabel 4.x. Hasil pengujian Response Time

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Method | Endpoint | Response Time | | | |
| Min (ms) | Max (ms) | Average (ms) | Total (s) |
| GET | /qr |  |  |  |  |
| POST | /devices |  |  |  |  |
| GET | /devices |  |  |  |  |
| DELETE | /devices |  |  |  |  |
| POST | /messages |  |  |  |  |
| POST | /webhook |  |  |  |  |

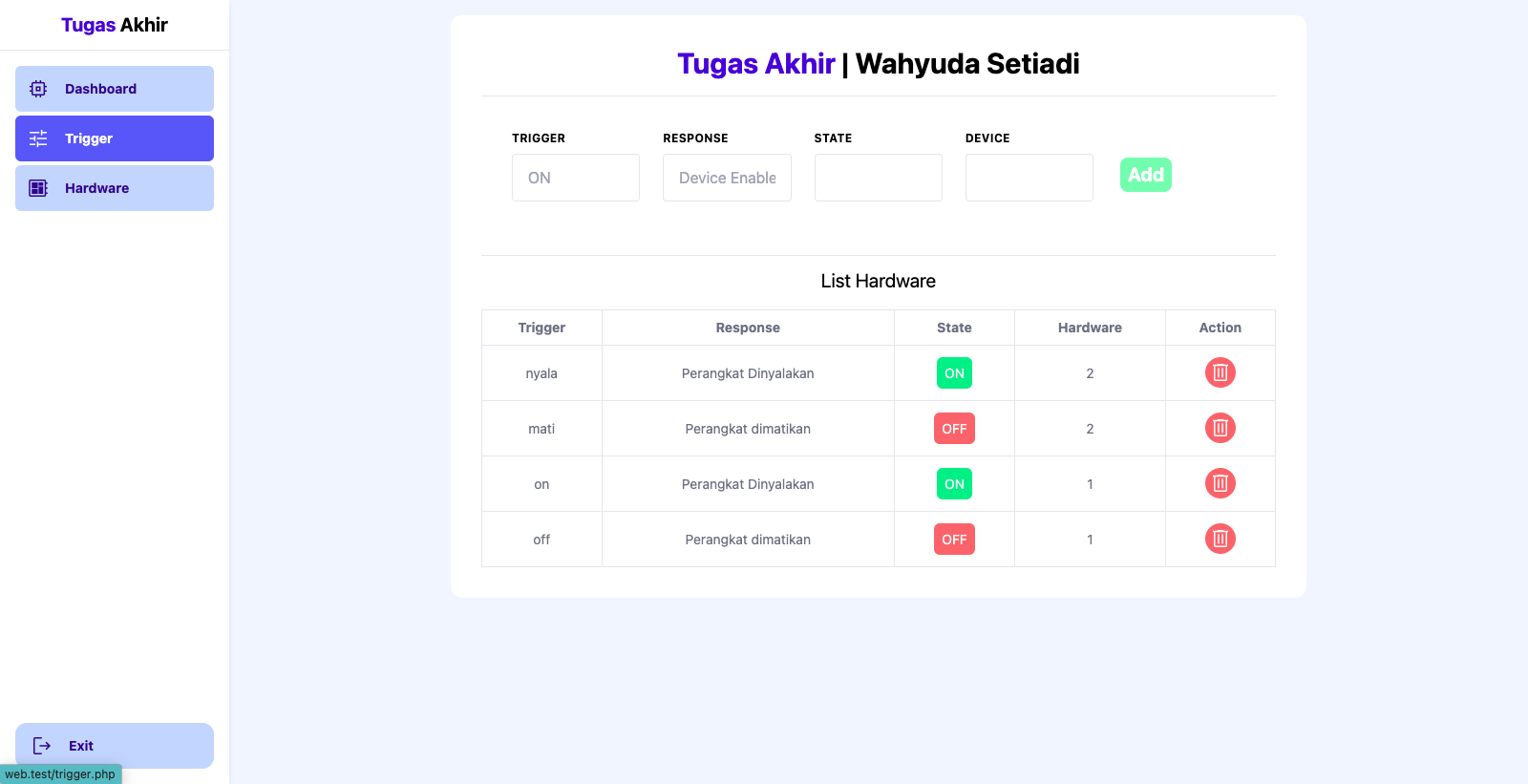
* + 1. Kdn
  1. Implementasi pada Web

Tujuan Awal pembuatan API adalah untuk melakukan integrasi antar aplikasi sehingga juga perlu dilakukan pengujian dengan cara melakukan integrasi menggunakan web. Penulis membuat web dengan bahasa pemrograman PHP, pada gambar 4.x. dibawah ini web akan menampilkan pesan masuk yang diterima oleh API.

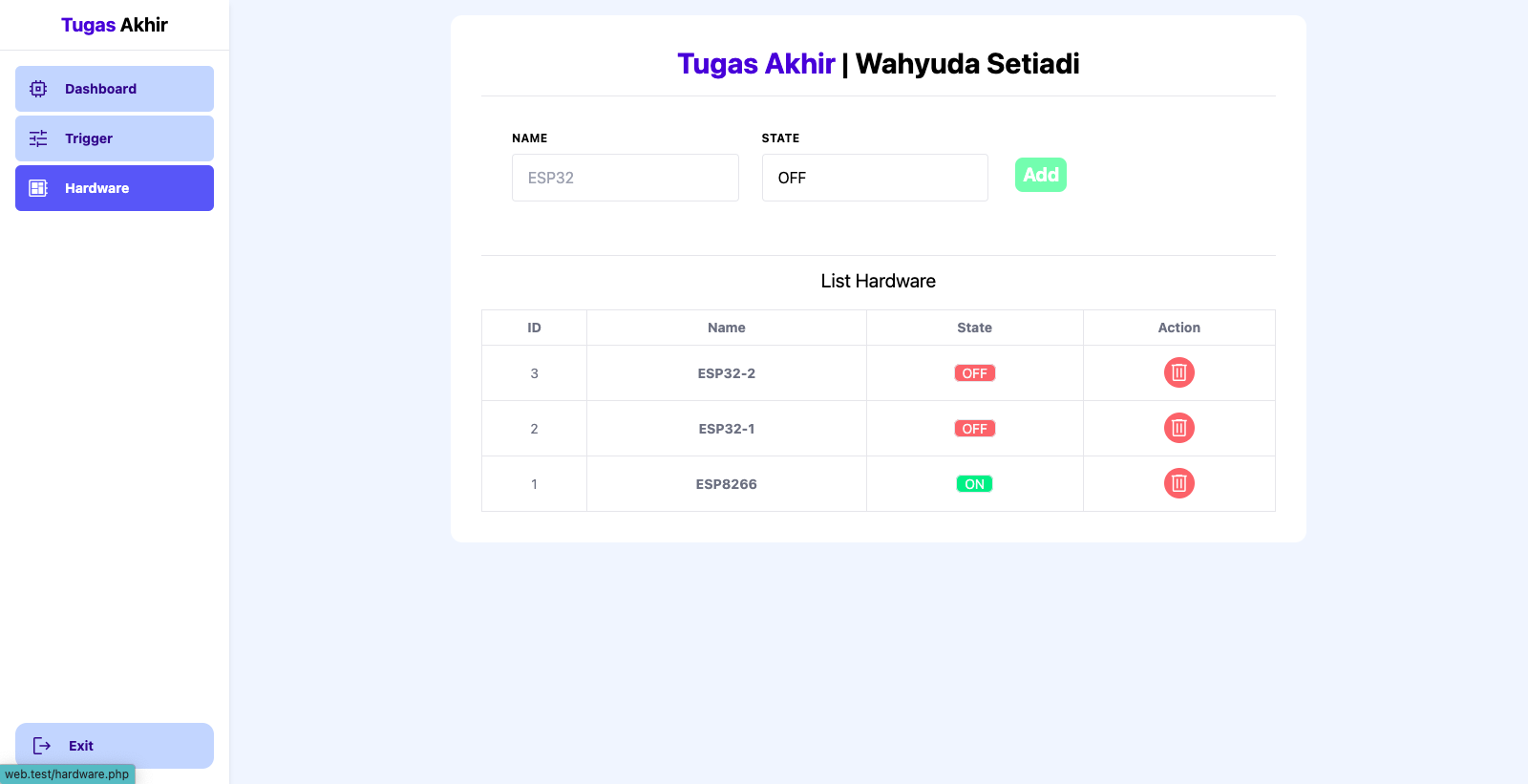


Gambar 4.x. Implementasi pada web menampilkan riwayat pesan masuk

Penulis juga menerapkaan fitur trigger sehingga tidak semua pesan masuk akan di proses oleh API, dibagian ini penulis melakukan implementasi trigger dan response dimana saat API menerima pesan yang telah didaftarkan pada trigger maka akan melakukan response dengan membalas pesan secara otomatis dan manjalankan logic sesuai yang di tentukan.



Gambar 4.x. Implementasi pada web menampilkan response berdasarkan trigger



Gambar 4.x. Implementasi pada web menampilkan hardware yang dikontrol

Untuk menambahkan itegrasi antar aplikasi pengguna juga membuat halaman web untuk menambahkan perangkat hardware yang dapat di kontrol pada sistem ini seperti ditunjukan pada gambar di atas.

* 1. Implementasi pada ESP32
  2. Sfdssd