

**UNIVERSITAS DIPONEGORO**

**PERANCANGAN SISTEM *DATABASE* DAN *SERVER* PADA SISTEM KEAMANAN KUNCI PINTU GEDUNG DENGAN *ACCESS CONTROL***

**TUGAS AKHIR**

**MUHAMMAD KHOIRIL WAFI**

**21060119140133**

**FAKULTAS TEKNIK**

**DEPARTEMEN TEKNIK ELEKTRO**

**PROGRAM STUDI SARJANA**

**SEMARANG**

**JUNI 2023**



**UNIVERSITAS DIPONEGORO**

**PERANCANGAN SISTEM *DATABASE* DAN *SERVER* PADA SISTEM KEAMANAN KUNCI PINTU GEDUNG DENGAN *ACCESS CONTROL***

**TUGAS AKHIR**

**Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik**

**MUHAMMAD KHOIRIL WAFI**

**21060119140133**

**FAKULTAS TEKNIK**

**DEPARTEMEN TEKNIK ELEKTRO**

**PROGRAM STUDI SARJANA**

**SEMARANG**

**JUNI 2023**

HALAMAN PERNYATAAN ORISINALITAS

**Tugas Akhir ini adalah karya saya sendiri,**

**dan semua sumber baik yang dikutip maupun yang dirujuk**

**telah saya nyatakan dengan benar**

|  |  |  |
| --- | --- | --- |
| NAMA | : | MUHAMMAD KHOIRIL WAFI |
| NIM | : | 21060119140133 |
| Tanda Tangan | : |  |
| Tanggal | : | 23 Juni 2023 |

HALAMAN PENGESAHAN

**HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI**

**TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademika Universitas Diponegoro, saya yang bertanda tangan di bawah ini:

Nama : Muhammad Khoiril Wafi

NIM : 21060119140133

Program Studi : SARJANA (S1)

Departemen : TEKNIK ELEKTRO

Fakultas : TEKNIK

Jenis Karya : TUGAS AKHIR

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Diponegoro **Hak Bebas Royalti Noneksklusif** (*Non-exclusive Royalty Free Right)* atas karya ilmiah saya yang berjudul: PERANCANGAN SISTEM *DATABASE* DAN *SERVER* PADA SISTEM KEAMANAN KUNCI PINTU GEDUNG DENGAN *ACCESS* *CONTROL.* Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti/Noneksklusif Universitas Diponegoro berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Semarang

Pada Tanggal : 23 Juni 2023

Yang menyatakan,

(Muhammad Khoiril Wafi)

NIM. 21060119140133

ABSTRAK

*Diisin nanti setelah perancangan selesai. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed euismod posuere urna, in finibus nisi dignissim sed. Proin in justo eu turpis interdum condimentum. Quisque eu mauris sed massa feugiat dapibus vitae sit amet elit. Donec lacinia neque non magna pellentesque, sed venenatis enim accumsan. Fusce sit amet dui magna. Vivamus fringilla lobortis vulputate. Curabitur id consequat nisl. Aliquam erat volutpat. Nulla facilisi. Vestibulum scelerisque euismod tellus nec eleifend. Nulla eu quam elit. Vivamus non lacus ligula. Aenean eu congue tellus. In suscipit risus ac elit pulvinar, in congue ex pulvinar. Nullam in felis tincidunt, tincidunt ligula et, rutrum velit. Aenean iaculis lobortis nulla nec auctor. Sed luctus elementum purus id dictum.*

***Kata Kunci****: Lorem ipsum dolor sit amet, consectetur adipiscing elit*

ABSTRACT

*Diisin nanti setelah perancangan selesai. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed euismod posuere urna, in finibus nisi dignissim sed. Proin in justo eu turpis interdum condimentum. Quisque eu mauris sed massa feugiat dapibus vitae sit amet elit. Donec lacinia neque non magna pellentesque, sed venenatis enim accumsan. Fusce sit amet dui magna. Vivamus fringilla lobortis vulputate. Curabitur id consequat nisl. Aliquam erat volutpat. Nulla facilisi. Vestibulum scelerisque euismod tellus nec eleifend. Nulla eu quam elit. Vivamus non lacus ligula. Aenean eu congue tellus. In suscipit risus ac elit pulvinar, in congue ex pulvinar. Nullam in felis tincidunt, tincidunt ligula et, rutrum velit. Aenean iaculis lobortis nulla nec auctor. Sed luctus elementum purus id dictum.*

***Key Words****:*

.

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah Subhanahu Wa Ta’ala atas rahmat, karunia, taufik, dan hidayah-Nya, sehingga penulis dapat menyelesaikan tugas akhir dan penyusunan laporan ini. Tugas Akhir berjudul “PERANCANGAN SISTEM *DATABASE* DAN *SERVER* PADA SISTEM KEAMANAN KUNCI PINTU GEDUNG DENGAN *ACCESS CONTROL*” ini diajukan sebagai syarat akhir untuk menyelesaikan program Sarjana di Departemen Teknik Elektro, Fakultas Teknik, Universitas Diponegoro, Semarang. Penyusunan dan penyelesaian Tugas Akhir ini tidak lepas dari bantuan dan dukungan dari semua pihak, baik secara langsung maupun tidak langsung. Oleh karena itu, pada kesempatan kali ini penulis mengucapkan terima kasih kepada:

1. Bapak Aghus Sofwan, S.T., M.T., Ph.D., selaku Ketua Departemen Teknik Elektro Fakultas Teknik Universitas Diponegoro Semarang.
2. Bapak Dr. Munawar Agus Riyadi, S.T., M.T., selaku Ketua Program Studi S1 Teknik Elektro Universitas Diponegoro Semarang.
3. Bapak Yuli Christiyono, S.T., M.T., selaku Sekretaris Program Studi S1 Teknik Elektro Universitas Diponegoro.
4. Bapak M. Arfan, S. Kom., M. Eng., selaku Dosen Pembimbing Utama yang selalu memberikan bimbingan, arahan, dan motivasi dalam menyelesaikan tugas akhir ini.
5. Bapak Imam Santoso, S.T., M.T., selaku Dosen Pendamping yang selalu memberikan motivasi, bimbingan, serta arahan yang membangun kepada penulis.
6. Bapak Yuli Christiyono, S.T., M.T., selaku dosen wali penulis yang selalu memberikan semangat dan mendengarkan dengan baik permasalahan penulis selama menempuh studi di Departemen Teknik Elektro Universitas Diponegoro.
7. Orang tua dan keluarga besar penulis yang selalu memberikan kasih sayang, doa, motivasi, dan semangat kepada penulis.
8. Sahabat kelompok tugas akhir Henric Dhiki Wicaksono dan Novi Dianasari yang banyak membantu penulis dalam menyelesaikan tugas akhir ini.
9. Keluarga Eternity S1 Teknik Elektro Universitas Diponegoro Angkatan 2019 khususnya konsentrasi Teknologi Informasi, terimakasih atas semuanya selama penulis menimba ilmu di S1 Teknik Elektro Universitas Diponegoro.
10. Dan semua pihak yang tidak dapat penulis sebutkan satu per satu yang telah membantu dari awal hingga akhir.

Penulis menyadari bahwa dalam penulisan laporan Tugas Akhir ini tentunya ada kekurangan. Oleh karena itu, kritik dan saran yang dapat membangun diperlukan demi kebaikan dan kesempurnaan penyusunan laporan di masa yang akan datang. Semoga Tugas Akhir ini dapat memberikan manfaat dan menambahkan pengetahuan bagi kita.

Semarang, 23 Juni 2023

(Muhammad Khoiril Wafi)

NIM. 21060119140133

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

# **BAB I**

**PENDAHULUAN**

1. Latar Belakang

Keamanan menjadi hal yang harus diperhatikan dalam sebuah gedung atau bangunan. Pada saat ini sistem penguncian masih banyak menggunakan penguncian tradisional dengan menggunakan kunci fisik yang tidak efisien mengingat jumlah ruangan yang banyak, kunci fisik juga mempunyai tingkat keamanan yang kurang dikarenakan kunci rentan untuk dicuri atau diduplikasi[1].

Mengelola sejumlah besar ruangan menjadi tantangan tersendiri. Seiring bertambahnya ruangan yang harus dikelola maka bertambah juga konpleksitas tugas yang harus dikerjakan seperti mengatur siapa saja yang dapat mengakses ruangan tersebut, bertambahnya ruangan juga meningkatkan resiko kunci hilang atau tertukar sehingga ruangan tidak bisa diakses[2].

Masalah keamanan ruangan dalam sebuah gedung dan efektifitas dapat diselesaikan dengan menggunakan sebuah sistem penguncian cerdas yang terorganisasi dan kerkoneksi ke sebuah sistem manajemen kunci pintu (*access control*) yang memiliki tingkat keamana yang lebih tinggi, adaptif dan fleksibel[3], [4].

Untuk mendukung kinerja dari sistem penguncian yang terorganisasi maka diperlukan sebuah *server* dan penyimpanan data. sebuah *server* akan menjalankan kode program yang bertugas untuk mengatur dan mengawasi semua aktifitas sistem penguncian dan sebuah penyimpanan data digunakan untuk menyimpan data-data seperti data pengguna, kunci, dan *backup*[5].

Dengan demikian, perancangan sistem *database* dan *server* pada sistem keamanan kunci pintu gedung dengan *access control* diharapkan dapat memberikan solusi yang efektif dalam meningkatkan keamanan kunci pintu gedung dan memberikan kenyamanan serta kemudahan dalam pengelolaannya.

1. Tujuan

Tujuan tugas akhir ini adalah untuk merancang sistem *database* dan *server* pada sistem keamanan kunci pintu gedung dengan *access control.*

1. Batasan Masalah

Dalam tugas akhir ini, telah ditentukan batasan-batasan masalah sebagai berikut:

1. Perancangan sistem *database* dan *server* akan difokuskan pada aplikasi *access control* untuk kunci pintu gedung. Hal ini meliputi perancangan struktur *database* untuk menyimpan informasi pengguna, akses pintu gedung, dan riwayat aktivitas akses serta perancangan server untuk menjalankan sistem penguncian.
2. Perancangan sistem *server* akan berfokus pada pemrosesan data dan komunikasi antara sistem *access control* dan perangkat keras.
3. Sistem keamanan kunci pintu gedung dengan *access control* akan mencakup teknologi seperti perangkat lunak *access control* untuk mengatur akses pengguna.
4. Sistem keamanan akan diimplementasikan pada gedung yang memiliki pintu masuk terbatas dengan akses terbatas pada karyawan atau pihak tertentu.
5. Penyimpanan data pengguna dan aktivitas akses akan disimpan dengan cara yang aman dan terenkripsi untuk memastikan keamanan informasi.
6. Pengembangan sistem *access control* harus memenuhi standar keamanan dan regulasi yang berlaku dalam industri atau lingkungan operasional gedung.
7. Sistem keamanan *access control* harus mudah dioperasikan dan dikelola oleh administrator gedung dengan tingkat keamanan dan kontrol yang optimal.
8. Sistematika Penulisan

Sistematika penulisan dalam laporan Tugas Akhir ini adalah sebagai berikut:

**BAB I PENDAHULUAN**

Bab ini berisi latar belakang, tujuan, batasan masalah, dan sistematika penulisan.

**BAB II DASAR TEORI**

Bab ini berisi teori yang melandasi dalam perancangan.

**BAB III PERANCANGAN SISTEM**

Bab ini berisi tentang perancangan sistem *database* dan *server* serta sistem keamanan kunci pintu gedung dengan *access control*.

**BAB IV PENGUJIAN DAN ANALISIS**

Pada bab ini dilakukan pengujian dan analisis hasil pengujian.

**BAB V PENUTUP**

Bab ini berisi kesimpulan dan saran terhadap hasil pengujian dan analisis yang telah dibuat.

**BAB II**

**DASAR TEORI**

1. Kendali Akses (Access Control)

Kendali akses atau *access control* merupakan sebuah mekanisme pengaturan kebijakan yang digunakan untuk membatasi dan mengatur hak akses pengguna terhadap suatu sumber dayaatau fasilitas tertentu. Kendali akses dapat diaplikasikan secara *hardware* maupun *software* yang akan menjadi pintu masuk pengguna sebelum mengakses suatu fasilitas. Dengan menggunakan kendali akses kita bisa mengatur dan membatasi akses pengguna sehingga hanya pengguna tertentu yang diijinkan yang bisa mengakses sumber daya yang dilindungi[6].

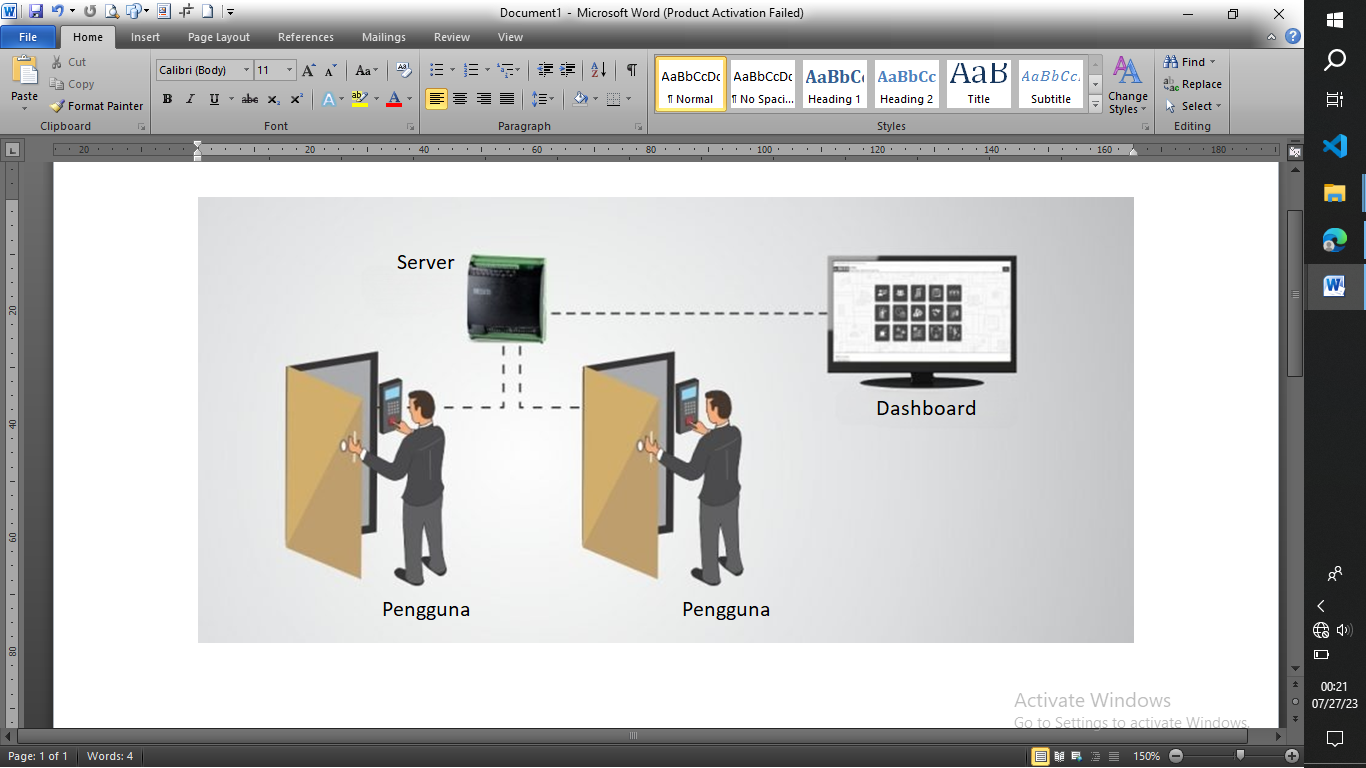
Terdapat beberapa pendekatan yang biasanya digunakan untuk membatasi akses pengguna untuk mengakses suatu sumber daya yang dimiliki, diantaranya yaitu :

1. *Access Control List*

ACL atau *Access Control List*  merupakan sebuah daftar yang berisi aturan-aturan atau daftar pengguna yang diijinkan untuk mengakses sebuah sumber daya yang dilindungi[7]. Dengan menggunakan ACL kita dapat mambatasi hak akses dengan hanya mengijinkan pengguna yang terdapat didalam daftar untuk mengakses sumber daya atau fasilitas tertentu. ACL akan memberikan pembatasan akses pada setiap pengguna secara ketat dan hanya mengijinkan akses jika pengguna memiliki hak akses yang tersimpan didalam daftar akses.

1. *Role-based Access Control*

RBAC atau *Role-based Access Control* merupakan pendekatan kendali akses yang tidak memberikan hak akses langsung ke pengguna melainkan memberikan hak akses berdasarkan peran yang dimiliki oleh pengguna seperti *admin*, operator, *supervisor*, dan lain sebagainya. Metode pengendalian akses berbasis peran akan memberikan aturan-aturan terkait akses berdasarkan peran pengguna tanpa mengetahui identitas pengguna secara spesifik dan hanya menggunakan peran pengguna sebagai parameter yang digunakan untuk melakukan pengendalian akses sumber daya.

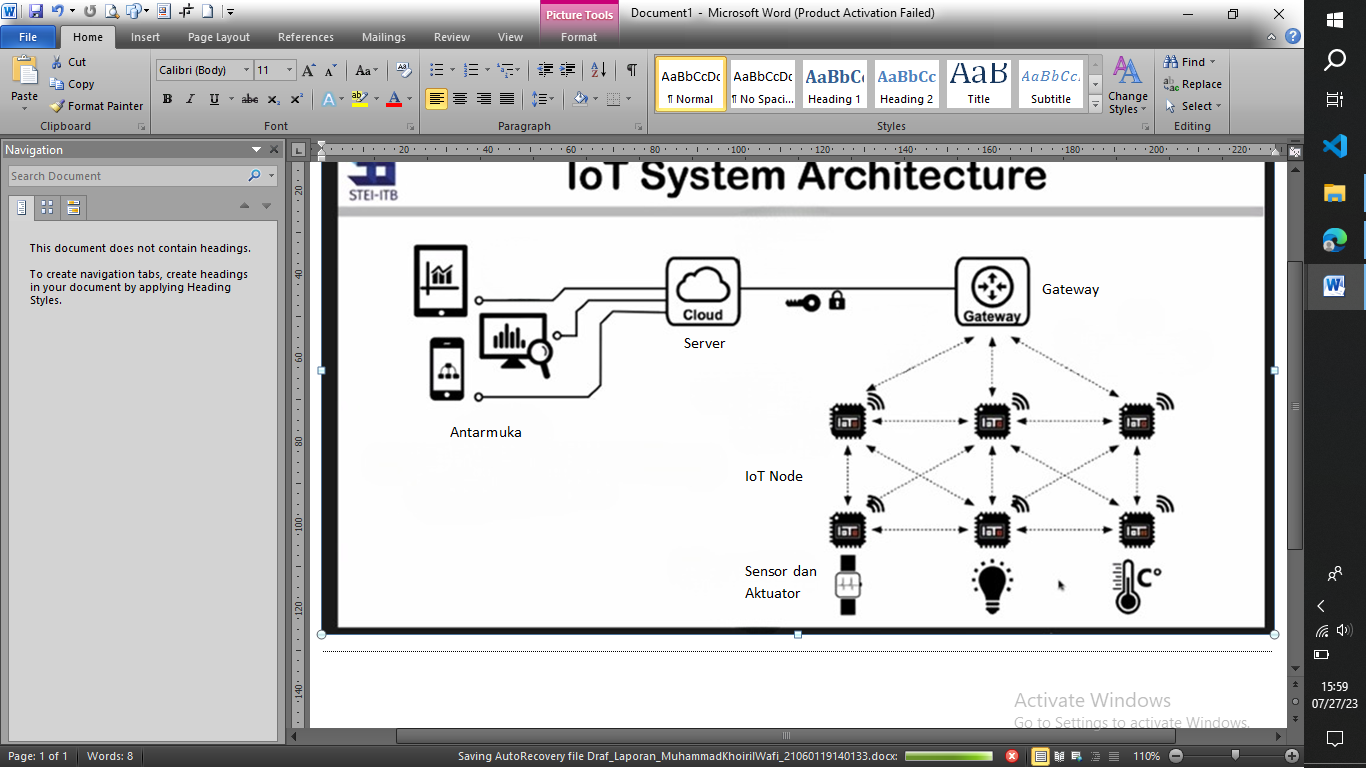


**Gambar 2.1** Contoh Pengendalian Akses

Gambar 2.1 merupakan contoh pengendalian akses pada pintu untuk membatasi akses pengguna ke suatu ruangan. Setiap pengguna memiliki identitas unik yang digunakan untuk menentukan tindakan apakah pengguna tersebut diijinkan atau tidak, proses perijinan dilakukan secara otomatis dan dikendalikan oleh sebuah *server* sebagai pusat kendali dan pusat data. Pada sistem pengendalian akses juga terdapat sebuah *sofware* yang digunakan sebagai antar muka untuk melakukan pengelolaan hak akses seperti mengijinkan pengguna baru atau menghapus ijin pengguna dan juga melihat riwayat akses pengguna.

1. Internet of Things

*Internet of Things* atau IoT merupakan sebuah konsep yang digunakan untuk mengembangkan koneksifitas internet, IoT juga memberikan gambaran mengenai kemampuan dari berbagai perangkat elektronik yang saling terhubung dengan membentuk sebuah jaringan komunikasi baik melalui internet maupun komunikasi lainnya seperti bluetooth. Dengan menggunakan konsep IoT kita dapat menghubungkan peralatan seperti sensor dan aktuator yang terhubung ke ke sebuah jaringan menjadi sebuah kesatuan sistem yang dapat dikendalikan secara efektif dan efisien dengan tingkat kerumitan yang rendah[8].



**Gambar 2.2** Arsitektur Sistem IoT

Gambar 2.2 menjelaskan arsitektur dari sistem IoT, sebuah sistem IoT dibangun dari beberapa bagian yang membentuk suatu sistem yang dapat berkomunikasi antara perangkat IoT dan juga server. Bagian-bagian dari sistem IoT diantaranya :

* 1. Sensor dan Aktuator

Sensor dan aktuator akan menjadi penghubung yang menjembatani interaksi antara sistem dengan dunia luar. Sebuah sensor akan melakukan akuisisi data yang merupakan sumber data dari sebuah sistem IoT sedangkan aktuator akan menerima data dan memberikan keluaran mekanis seperti menyalakan lampu, membuka kunci dan lain sebagainya.

* 1. IoT *Node*

IoT *node* bertindak sebagai penghubung antara perangkat keras seperti mikrokontroler kedalam sebuah jaringan IoT dengan adanya IoT *node* memungkinkan antar perangkat IoT untuk saling berkomunikasi satu dengan lainnya secara independen[8].

* 1. *Gateway*

Pada sistem IoT umumnya akan terhubung ke sebuah server sebagai pusat kendali dan pusat data. *Gateway* menjadi penghubung antara IoT *node* dengan jaringan internet global, sebuah *gateway* misalnya *router* dan modem akan menyediakan koneksi internet yang dapat digunakan oleh perangkat IoT untuk terhubung ke *server* melalui koneksi internet[8].

* 1. *Server*

*Server* akan menjadi pusat kendali dan penyimpanan data terpusat pada sistem IoT. Dengan adanya sebuah *server* memungkinkan perangkat IoT untuk saling saling berkoordinasi dengan *server* sebagai pengaturnya, dengan adanya *server* juga memungkinkan perangkat IoT melakukan penyimpanan data dan informasi secara *online* pada *database*[8].

* 1. Antarmuka

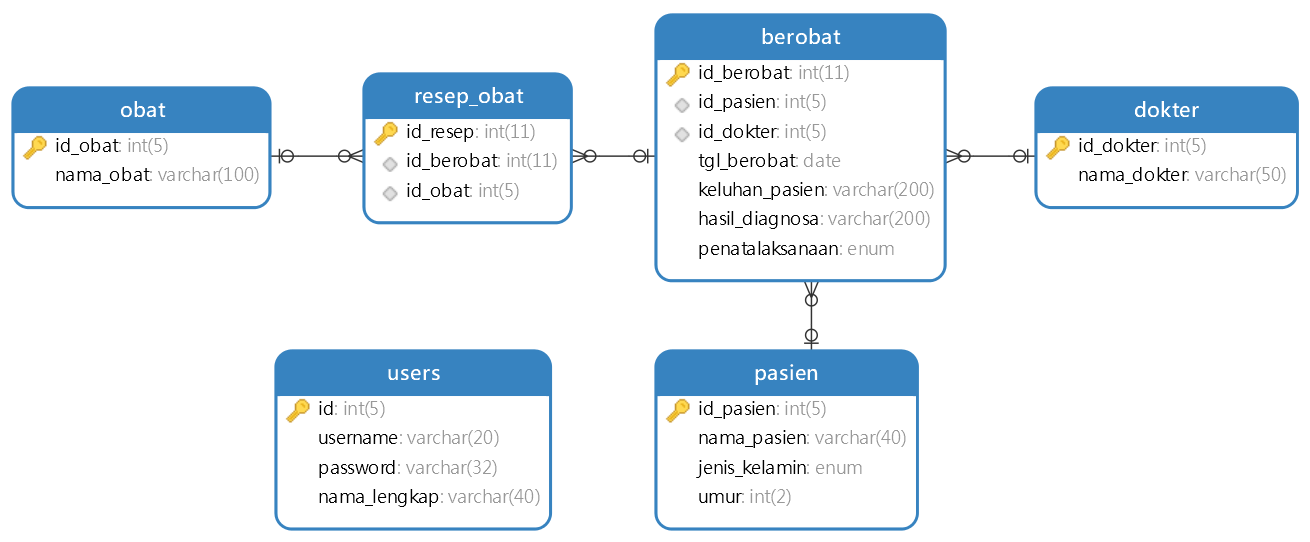
Antarmuka pada sistem IoT digunakan untuk melakukan proses pemantauan aktifitas dan data yang terjadi didalam sistem IoT. Antarmuka akan memberikan akses kepada pengguna untuk berinteraksi dengan sistem IoT seperti melihat data dan memberikan perintah ke perangkat IoT.

1. *Database* MySQL

*Database* atau basisdata merupakan sekumpulan data yang terintegrasi dan diatur sedemikian rupa sehingga data tersebut dapat dicari, diambil, ditambahkan, dan diolah dengan tepat[6]. Selain berisikan data, *database* juga berisikan informasi lainnya seperti tipe data, nama-nama kolom dan baris. Adapun fungsi lain dari *database* yaitu[9] :

1. Mempermudah identifikasi data dengan cara pengelompokkan data, salah satu contohnya dengan pembuatan beberapa tabel atau *field* yang berbeda-beda.
2. Meminimalisir suatu data ganda.
3. Penyimpanan secara digital.
4. Menjadi alternatif lain terkait masalah penyimpanan ruang dalam suatu aplikasi.

MySQL merupakan singkatan dari *Structured Query Language*. SQL merupakan bahasa terstruktur yang khusus digunakan untuk mengolah database. MySQL merupakan sistem manajemen *database* yang bersifat relational. Artinya, data yang dikelola dalam *database* akan diletakkan pada beberapa tabel yang terpisah sehingga manipulasi data akan jauh lebih cepat[10].

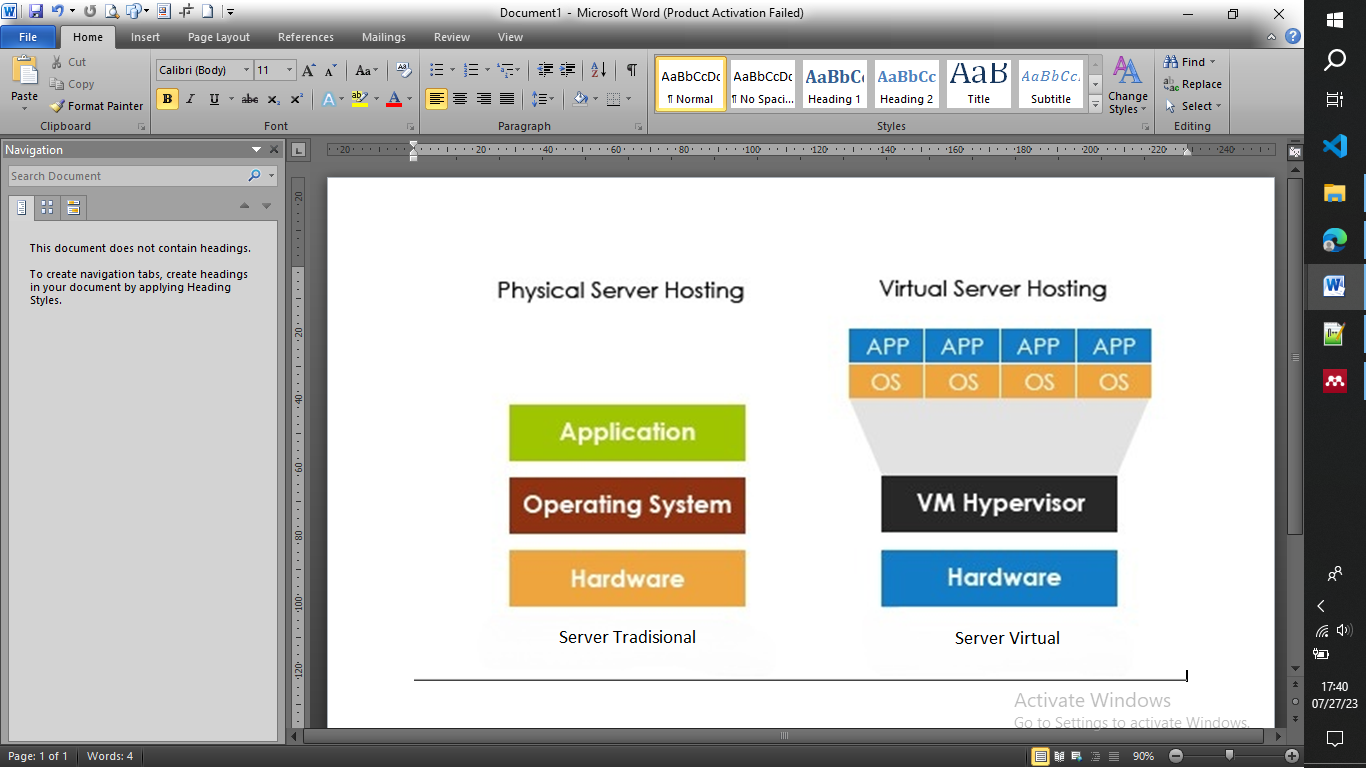


**Gambar 2.3** Struktur Tabel MySQL

MySQL merupakan sistem database relasional yang berarti pada MySQL data akan disimpan dalam tabel-tabel yang terdiri dari beberapa kolom, gambar 2.3 menjelaskan strukur data yang tersimpan didalam *database* MySQL, data yang disimpan dapat berupa teks, angka, *boolean*, waktu, hari dan lain sebagainya tergantung dengan tipe data pada setiap kolomnya. Dalam MySQL terdapat relasi antar tabel yang memberikan informasi hubungan antara tabel satu dengan tabel lainnya serta perilakunya terhadap perubahan yang terjadi pada tabel induknya.

1. *Virtual Private Server*

*Virtual Private Server* atau VPS adalah jenis server yang secara eksklusif diperuntukkan bagi satu pengguna, sehingga seluruh sumber daya yang ada di dalamnya tidak dipengaruhi atau dibagi dengan pengguna lain. Dengan menggunakan teknologi VPS, sebuah mesin fisik dapat menjalankan beberapa sistem operasi secara bersamaan. Pengguna VPS memiliki kendali penuh untuk mengatur seluruh konfigurasi sesuai kebutuhan. Teknologi yang digunakan dalam VPS adalah virtualisasi hardware pada server fisik yang memungkinkan pembagian sumber daya menjadi beberapa bagian yang berbeda, sehingga setiap VPS berfungsi seperti server pribadi yang terisolasi dari pengguna lainnya[11].

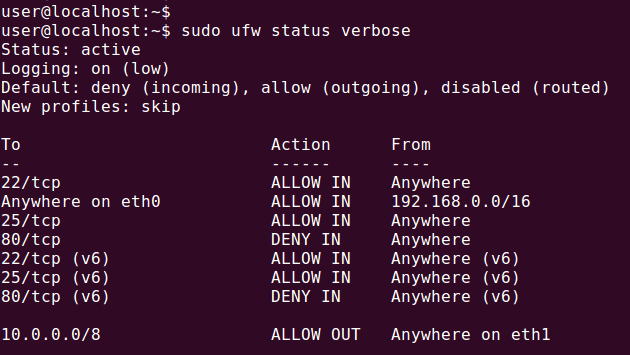


**Gambar 2.4** Perbedaan Server Tradisional dan Virtual

Gambar 2.4 memperlihatkan perbedaan antara *server* tradisional dengan *server* virtual, pada *server* tradisional satu *server* fisik hanya bisa digunakan untuk menjalankan satu sistem operasi dan satu aplikasi sedangkan pada *server* virtual memungkinkan menjalankan beberapa sistem operasi dan aplikasi yang banyak didalam satu *server* fisik. Perbedaan utama yaitu adanya virtualisasi pada server virtual yang bertugas untuk membagi sumber daya *server* fisik menjadi beberapa bagian untuk menjalankan sistem operasi. Proses pembagian ini dilakukan menggunakan perangkat lunak, sehingga satu server fisik dapat menampung beberapa VPS yang berjalan secara terpisah. Setiap VPS memiliki akses penuh (*Full Root Access*) untuk mengatur sistem operasi dan konfigurasi sesuai keinginannya, termasuk pengaturan *init script*, pengguna, pemrosesan data, sistem file, serta sumber daya server seperti CPU dan RAM yang beroperasi secara independen.

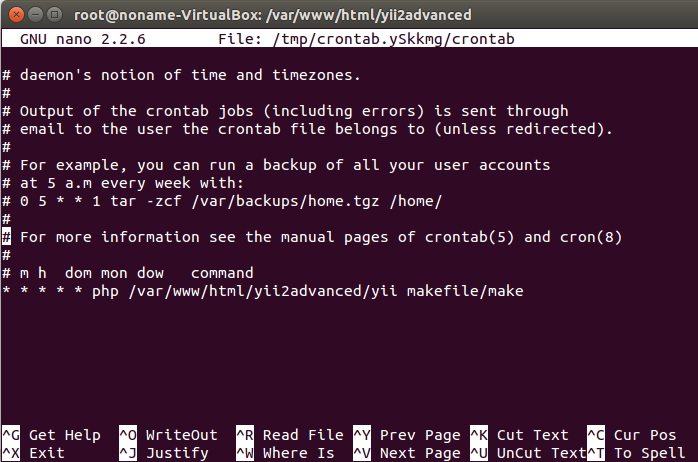
1. Sistem Operasi Ubuntu

Ubuntu adalah salah satu distribusi Linux yang didasarkan pada Debian. Distro ini menyediakan sistem operasi berbasis Debian dengan jadwal rilis yang teratur dan dukungan yang luas, baik untuk pengguna perorangan maupun perusahaan. Selain digunakan sebagai sistem operasi desktop, Ubuntu juga populer digunakan sebagai sistem operasi pada *server*, karena memiliki karakteristik yang ringan dan dapat diandalkan dalam menangani berbagai tugas *server*[12]. Ubuntu menyediakan fitur yang cukup lengkap untuk diaplikasikan pada *server* seperti adanya *firewall* dan penjadwalan menggunakan cronjob.



**Gambar 2.5** Contoh Konfigurasi Firewall Ubuntu

Gambar 2.5 merupakan contoh konfigurasi *firewall* pada sistem operasi ubuntu. Di dalam *firewall* semua komunikasi yang keluar dan masuk akan difilter atau dikontrol, *port*-*port* yang rentan servisnya dapat di tutup atau diblokir, sehingga hanya pihak yang diizinkan saja yang boleh lewat. Cara ini merupakan salah satu pengamanan jaringan yang sering digunakan[13]. Selain *firewall* ubuntu juga menyediakann sistem penjadwalan dengan menggunakan cronjob.

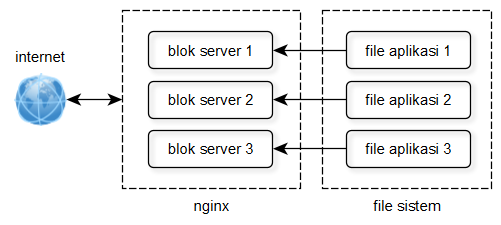


**Gambar 2.6** Contoh Konfigurasi Cronjob

Gambar 2.6 merupakan contoh konfigurasi conjob pada sistem operasi ubuntu, dengan menggunakan cronjob maka dimungkinkan untuk mengeksekusi perintah pada terminal secara berulang setiap rentang waktu tertentu seperti setiap 1 minggu sekali atau 1 jam sekali.

1. Nginx

Nginx merupakan perangkat lunak *open-source* yang memiliki performa tinggi sebagai server HTTP dan *reverse proxy*. Nginx memiliki keunggulan dalam memberikan konten statis secara cepat, sambil menggunakan sumber daya sistem dengan efisien. Selain itu, Nginx dapat mendistribusikan konten HTTP dinamis melalui jaringan menggunakan FastCGI *handler* untuk menjalankan *script*[14]. Nginx dibangun secara modular, dengan demikian nginx mampu mendukung berbagai fitur seperti *Load Balancing* dan *Reverse Proxying*, *Virtual hosts*.

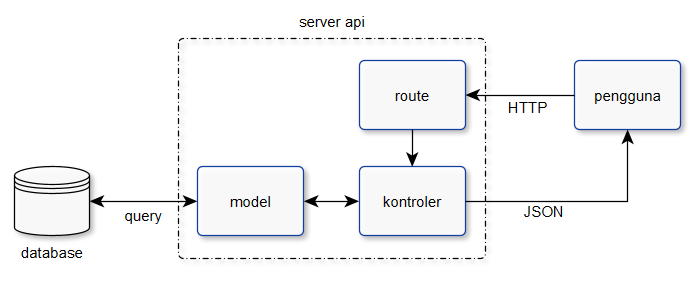


**Gambar 2.7** Virtual Host pada Nginx

Gambar 2.7 menjelaskan penggunaan virtual host pada nginx, dengan menggunakan virtual host nginx dapat menangani beberapa aplikasi yang sudah dikonfigurasi pada masing-masing *port*-nya. Nginx menggunakan pendekatan *asynchronous-event* untuk menangani permintaan sehingga nginx tidak sepenuhnya bergantung pada ‘*thread*’ untuk menangani permintaan (*request*). Arsitektur ini memberikan hasil kinerja nginx pada saat dibebani hanya memerlukan sedikit memori dalam jumlah yang bisa diprediksikan[14].

1. *Application Programming Interface*

*Application Programming Interface* atau API merupakan sebuah protokol yang digunakan untuk menampilkan layanan atau data yang disediakan oleh sebuah aplikasi melalui sumber daya yang telah ditentukan. Dengan menggunakan API, aplikasi lain dapat mengakses layanan atau data tanpa harus mengimplementasikan prosedur atau objek yang mendasarinya[15].



**Gambar 2.8** Cara Kerja API

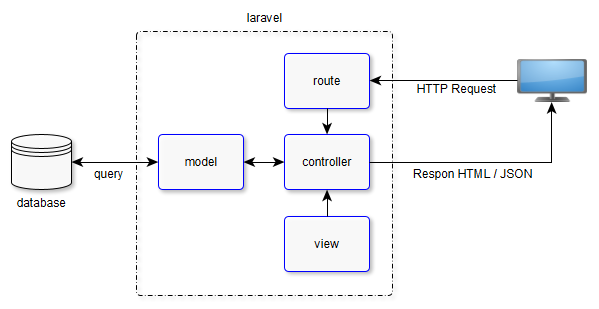
Gambar 2.8 menjelaskan cara kerja dari API, pengguna (*client*) akan mengirimkan permintaan HTTP melalui sebuah *endpoint*, *endpoint* merupakan sebuah URL yang digunakan untuk berinteraksi dan mengakases sumber daya yang disediakan oleh API tersebut. API akan mengirimkan data dengan format JSON sebagai respon dari permintaan *client*. JSON atau Java Script Object Notation merupakan sebuah format data yang yang ditemukan oleh Douglas Crockford pada tahun 2006, JSON menggunakan sintak objek javascript sebagai format data berbasis teks, JSON memiliki ukuran data yang lebih kecil serta waktu proses yang lebih cepat jika dibandingkan dengan format data lainnya[16].

Dengan menggunakan sebuah API maka pengguna tidak akan terhubung secara langsung dengan *database*, sehinggga akan meningkatkan keamanan dari sistem, jika sewaktu waktu terjadi hal-hal yang tidak di ingin kan seperti rusaknya data atau sistem di retas oleh pihak yang tidak bertanggung jawab data asli tetap aman tersimpan pada database tanpa mengalami gangguan sedikitpun[16].

1. *Framework* Laravel

*Framework* adalah struktur kerja yang terdiri dari sekumpulan program, terutama berisi kelas dan fungsi, yang dapat membantu para pengembang (*developer*) dalam menangani berbagai permasalahan yang sering dihadapi dalam pemrograman. Dengan menggunakan framework, para pengembang dapat lebih terfokus dan lebih cepat dalam membangun aplikasi karena berbagai tugas rutin seperti koneksi ke database, pemanggilan variabel, pengelolaan file, dan sebagainya telah diatur dan disediakan dalam framework tersebut[17].

Laravel merupakan kerangka kerja pemrograman web yang menggunakan bahasa pemorgraman PHP yang terbuka dan gratis, laravel diperuntukkan untuk pengembangan aplikasi berbasis *website* maupun API dengan menggunakan pendekatan pola MVC atau Model View Controller[18]. Arsitektut MVC memiliki business logic yang terpisah dari model dan *presentation*, sehingga saat melakukan modifikasi pada program tidak mempengaruhi komponen lain yang tidak diubah, dan proses pengembangan yang lebih cepat, serta dapat menggunakan *reuse of code* dimana fungsi ini berguna dalam pengembangan website tanpa harus melakukan coding dari awal[17].



**Gambar 2.9** Konsep MVC pada Laravel

Gambar 2.9 menjelaskan kosep MVC yang ada pada Laravel, saat pengguna melakukan *request* ke Laravel maka *request* tersebut akan ditangani oleh *route*, *route* disini bertugas sebagai pintu masuk yang menyediakan *url*/*enpoint* mana saja yang dapat diakses oleh *client*. Jika tidak ditemukan masalah maka *route* akan melanjutkan *request* tersebut ke modul MVC dengan tugas sebagai berikut :

1. Model

Model merupakan sebuah modul yang digunakan untuk melakukan proses *query* ke *database*, pada laravel model menyediakan mekanisme *query* *builder* yang menjamin proses *query* berjalan dengan mudah dan aman dengan berbagai filter untuk menghindari kesalahan.

1. *View*

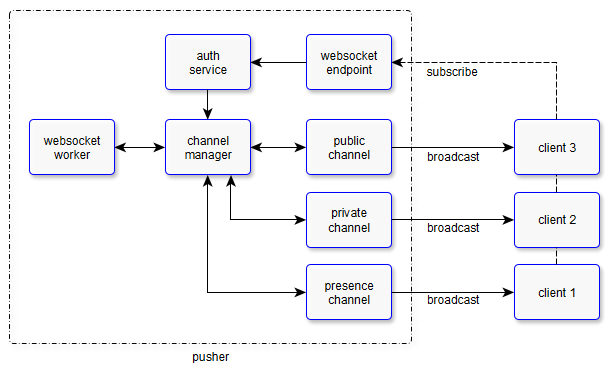
*View* merupakan bagian dari Laravel yang berfungsi untuk menyediakan tampilan antarmuka kepada pengguna, antarmuka akan tampil pada *browser* pengguna. Pada implementasi Laravel sebagai API maka *view* tidak mendapat tugas khusus karena pada API *controller* akan mengembalikan respon dalam format JSON dan tidak mengembalikan tampilan dalam HTML.

1. *Controller*

*Controller* digunakan untuk mengolah data yang diminta oleh pengguna, sebuah *controller* tidak langsung menerima permintaan dari pengguna melainkan *controller* akan dipanggil oleh bagian routing pada saat ada permintaan. *Controller* juga akan mengirimkan respon ke *client* sesuai dengan permintaan yang dikirimkan.

1. *Pusher Web*s*ocket*

*Websocket* merupakan sebuah protokol komunikasi web berbasis *client*-*server*, keberadaan websocket dinilai dapat menggantikan teknilogi AJAX sebagai pendahulu komunikasi *client*-*server*. *Websocket* merupakan teknologi yang mampu memberikan performa terbaik ketika diimplementasikan dalam sistem dengan *rate*-*request* tinggi, dibandingkan dengan teknologi komunikasi lain termasuk AJAX[19]. *Websocket* memungkinkan komunikasi dua arah antara *client* dan *server* dengan menggunakan koneksi yang sudah terjalin, hal ini dikarenakan pada *websocket*, koneksi akan terus terjalin selama tidak terjasi error atau ada permintaan pemutusan koneksi. Salah satu layanan yang menyediakan koneksi *websocket* yaitu Pusher, Pusher menyediakan komunikasi *realtime* antara *server* dan *client* melalui *channel*-*channel* yang telah tersedia.



**Gambar 2.10** Diagram Pusher Channel

Gambar 2.10 diatas menjelaskan bagian-bagian dari pusher channel, pusher hanya mendukung komunikasi satu arah yaitu dari server ke client dengan menggunakan metode broadcasting. Untuk bisa mendapatkan pesan dari broadcasting maka client harus terhubung ke salah satu channel dengan cara melakukan subscribe ke channel tersebut. Didalam pusher terdapat 3 channel yaitu :

* 1. *Public* *Channel*

*Public* *Channel* merupakan *channel* yang tersedia dan dapat di-*subscribe* oleh semua penggguna tanpa adanya proses autentikasi.

* 1. *Private* *Channel*

*Private* *Channel* merupakan *channel* yang hanya tersedia untuk *client* tertentu, sebelum terhubung ke *private* *channel* *client* harus melakukan proses autentikasi untuk mendapatkan kode *signature* channel yang digunakan untuk melakukan *subscribe* ke *channel* tersebut.

* 1. *Presence* *Channel*

*Presence* *Channel* merupakan pengembangan dari *channel* *private*, pada *presence* *channel* perilaku *client* dapat dipantau seperti ada client baru yang melakukan subcribe atau ada client yang keluar.

1. Penelitian Terdahulu

Beberapa penelitian terdahulu yang dijadikan referensi dalam perancangan sistem database dan server pada sistem keamanan kunci pintu gedung menggunakan access control adalah :

**Tabel 2.1** Ringkasan Penelitian Terdahulu

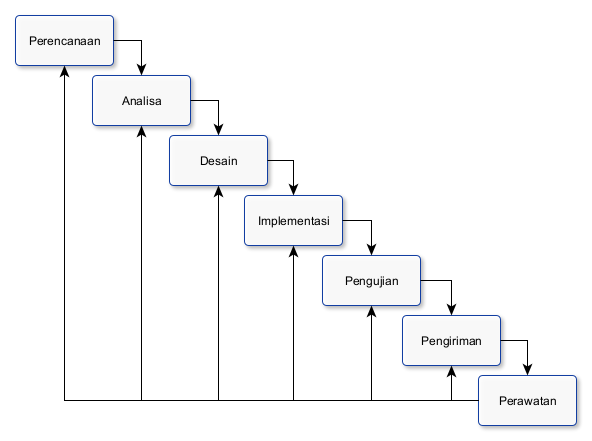
|  |  |  |  |
| --- | --- | --- | --- |
| **Judul** | **Tahun** | **Penulis** | **Pembahasan** |
| *Perancangan Database IoT Berbasi Cloud dengan Restful API* | 2021 | M. Kasyful Anwar | Dalam penelitian ini dibahas mengenai rancangan database berbasis cloud dengan Restful API untuk IoT agar data IoT aman dan memiliki *throughput* yang bagus dengan struktur data yang diatur pada database. |
| *Development of Intelligent Door Lock System for Room Management Using Multi Factor Authentication* | 2023 | I. Hermawan, D. Arnaldy, P. Oktivasari, dan D. A. Fachrudin | Penelitian ini menjelaskan mengenai permasalahan yang muncul pada manajemen kunci secara tradisional seperti jumlah kunci yang banyak, mudah hilang dan mudah diduplikasi. |
| *RFID and GSM Based Attendance Monitoring System using door locking / unlocking system and Its Hardware Implementation* | 2015 | A. Jain, V. L. Kalyani, dan B. Nogiya | Penelitian ini menjelaskan penggunaan RFID untuk melakukan absesnsi siswa sekaligus sebagai kunci untuk membuka pintu ruang kelas. |

BAB III

**PERANCANGAN SISTEM**

1. Metode Perancangan

Dalam perancangan tugas akhir ini menggunakan pendekatan *waterfall* dengan pertimbangan yang matang. Metode *waterfall* memberikan struktur dan ketertiban yang sangat diperlukan dalam pengembangan proyek ini. Setiap tahapan akan dilaksanakan secara berurutan sesuai dengan gambar 3.1.



**Gambar 3.1** Alur Perancangan

Keputusan ini diambil karena proyek tugas akhir ini memiliki batasan waktu yang jelas dan persyaratan yang sudah ditetapkan dengan baik. Dengan menggunakan metode *waterfall*, kita dapat memastikan bahwa setiap tahapan diselesaikan dengan cermat sebelum melanjutkan ke tahap berikutnya, sehingga mengurangi risiko kesalahan dan memastikan kelancaran jalannya proyek. Selain itu, pendekatan ini juga memungkinkan untuk melakukan perencanaan yang lebih terperinci, memahami kebutuhan proyek secara menyeluruh, dan mengidentifikasi risiko potensial dengan lebih baik. Dengan demikian, metode *waterfall* menjadi pilihan yang tepat untuk memastikan keberhasilan dan kualitas dari tugas akhir ini. Pada gambar 3.1 terlihat alur perancangan dimulai dari proses perencanaan samapi dengan proses pemeliharaan dengan penjelasan sebagai berikut :

* + 1. Perencanaan

Pada tahap perencanaan dilakukan pengumpuna informasi mengenai tujuan serta batasan-batasan proyek yang harus terpenuhi. Pada tahap perencanaan juga mendefinisikan beberapa kebutuhan fungsional dan kebutuhan non-fungsional pada rancangan sistem untuk dikembangkan lebih lanjut.

* + 1. Analisa

Pada tahap analisa dilakukan studi lieratur mengenai tujuan dan batasan-batasan proyek serta melakukan pengumpulan informasi mengenai sistem yang akan dibuat dengan melihat dan mempelajari penelitian/perancangan terdahulu. Analisa dilakukan bersama kelompok tugas akhir dengan melakukan penelusuran pustaka serta referensi yang relevan dan disertai dengan diskusi sehingga menghasilkan keluaran berupa gambaran umum mengenai bagian-bagian dari sistem yang akan digunakan pada tahap selanjutnya.

* + 1. Desain

Setelah mendapatkan gambaran mengenai sistem yang akan dibuat, selanjutnya dilakukan pemodelan sistem dengan menggunakan diagram pemodelan. Diagram pemodelan digunakan untuk menggambarkan hubungan dan interaksi antar komponen didalam sistem, dengan menggunakan diagram maka akan menghasilkan gambar sistem yang lebih rinci yang digunakan untuk membangun sistem.

* + 1. Implementasi

Pada tahap implementasi, gambar pemodelan dari sistem kemudian diterjemahkan atau diimplementasikan menggunakan kode program untuk membuat sistem dapat bekerja sesuai dengan dasain awal yang telah ditentukan.

Implementasi dilakukan dengan menggunakan bahasa pemrograman PHP dengan menggunakan *framework* Laravel. Laravel mempunyai karakteristik RAD atau *Rapid* *Application* *Development* sehingga implementasi dapat dilaksanakan dengan cepat dan efisien, Laravel juga menyediakan komponen-komponen yang akan digunakan didalam sistem seperti autentikasi, antrian, notifikasi pesan, *websocket*, dan lain sebagainya sehingga dapat mempermudah proses implementasi.

* + 1. Pengujian

Pada tahap pengujian, sistem yang telah dibangun diuji kinerjanya dengan mengirimkan beberapa permintaan data dan menganalisa hasil keluaran yang diberikan apakah sesuai atau tidak. Proses pengujian bertujuan untuk mengetahu kesesuaian hasil implementasi dengan desain awal dan juga untuk mengetahui performa dari sistem yang telah dibuat.

* + 1. Pengiriman

Pada tahap pengiriman, aplikasi yang sudah selesai dilakukan pengujian dan dinyatakan lolos uji selanjutkan akan dipasang atau diinstall pada server. Pada proses deployment juga akan dilakukan pemasangan aplikasi tambahan seperti *database*, penjadwalan, *firewall*, dan lain sebagainya. Pada tahap ini sistem yang sudah dibuat dapat digunakan oleh pengguna umum.

* + 1. Perawatan

Pada tahap pemeliharaan atau perawatan dilakukan pengecekan secara berkala pada aplikasi yang sudah berjalan pada server. dengan adanya perawatan secara teratur jika ditemukan permasalahan atau komponen yang tidak bekerja maka akan langsung diperbaiki sehingga tidak mengganggu kinerja dari server.

1. Kebutuhan Fungsional dan Non-Fungsional

Kebutuhan fungsional menggambarkan apa saja yang dimiliki dan bisa dilakukan oleh sistem yang akan dibuat sedangkan kebutuhan non-fungsional lebih menekankan keunggulan yang dimiliki oleh sistem yang akan dibuat. Dengan adanya analisa kebutuhan fungsional dan non-fungsional akan memberikan gambaran tentang batasan dan pencapaian yang harus dimiliki oleh sistem agar sistem tersebut dapat dikatakan sesuai dengan rangcangan awal. Bedasarkan analisa pada tahap perancangan maka didapat hasil kebutuhan fungsional sesuai pada tabel xx dan kebutuhan non fungsinal sesuai dengan tabel xx.

**Tabel 3.1** Kebutuhan Fungional

|  |  |  |
| --- | --- | --- |
| **No** | **Kebutuhan** | **Penjelasan** |
| 1 | Autentikasi login dan logout | Sistem memiliki metode autentikasi untuk melakukan login dan logout |
| 2 | Ganti password | Sistem memiliki metode untuk mengganti password |
| 3 | Reset password | Sistem memiliki metode untuk menangani kondisi lupa password |
| 4 | Verifikasi email | Sistem memiliki metode untuk melakukan verifikasi email |
| 5 | Ganti profil | Sistem memiliki metode untuk memperbarui profil pengguna |
| 6 | Lihat daftar akses | Sistem memiliki metode untuk melihat daftar akse yang dimiliki oleh pengguna |
| 7 | Verifikasi akses | Sistem memiliki metode untuk melakukan verifikasi akses pengguna |
| 8 | Lihat riwayat aktifitas | Sistem memiliki metode untuk melihat riwayat aktifitas |
| 9 | Lihat daftar pintu | Sistem memiliki metode untuk melihat daftar pintu yang ada |
| 10 | Membuka pintu jarak jauh | Sistem memiliki metode untuk membuka atau mengunci pintu dari jarak jauh |
| 11 | Register pintu baru | Sistem memiliki metode untuk menambahkan pintu baru ke sistem penguncian |
| 12 | Koneksi websocket untuk pintu | Sistem memiliki jalur komunikasi menggunakan websocket |
| 13 | Mendapatkan signature | Sistem memiliki metode untuk memperoleh signature untuk melakukan subscribe ke channel websocket |
| 14 | Update status pintu | Sistem memiliki metode untuk melakukan update status pintu |
| 15 | Peringatan pintu | Sistem memiliki metode oleh pintu untuk mengirim peringatan |

Tabel 3.1 menjelaskan kebutuhan fungsional yang harus disediakan oleh sistem yang akan dibuat. Sistem *database* dan *server* yang dibuat akan menyediakan layana untuk 2 jenis *client*, pertama yaitu perngkat aplikasi mobile yang dipakai oleh operator dan pengguna, yang kedua yaitu perangkat kunci pintu berbasi IoT. Untuk jenis client ini sistem yang dibuat akan memberikan layanan API yang digunakan untuk melakukan beberapa kegiatan seperti login, logout, ganti password, reset password, verifikasi email, verifikasi akses dan lain sebagainya.

**Tabel 3.2** Kebutuhan Non-Fungsional

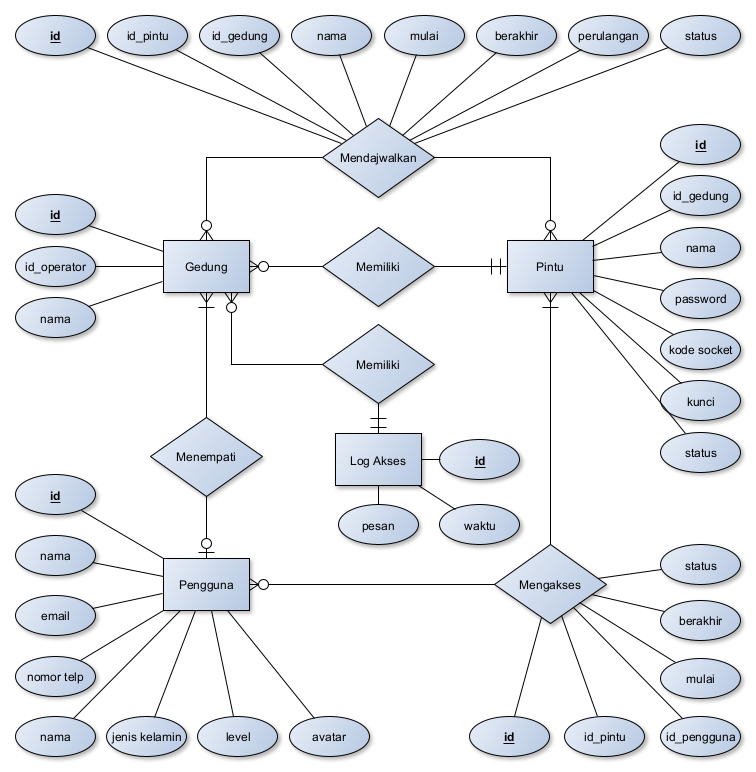
|  |  |  |
| --- | --- | --- |
| **No** | **Kebutuhan** | **Penjelasan** |
| 1 | Tersedia | Sistem dapat beroperasi penuh selama 7 hari dalam satu minggu dan 24 jam dalam satu hari |
| 2 | Cepat | Sistem memiliki waktu respon yang cepat, tidak lebih dari 5 detik |
| 3 | Aman | Sistem dapat beroperasi dengan aman baik dalam penyimpanan data maupun pengiriman data. |

Tabel 3.1 menjelaskan kebutuhan non-fungsional pada sistem, kebutuhan non-fungsional menggambarkan keunggulan dari sistem seperti sistem harus dapat bekerja 24 jam penuh, sistem harus memberikan respon dengan cepat dan juga sistem harus menyimpan dan mengirim data dengan aman.

1. Perancangan *Database*

*Database* digunakan untuk menyimpan data yang digunakan dalam sistem penguncian pintu gedung seperti data pengguna, data pintu, data akses dan lain sebagainya. *Database* yang digunakan harus dapat menunjang kinerja sistem dengan karakteristik respon yang cepat dan pengelolaan data terstruktur.

Berdasarkan penelitian [20] yang membandingkan kinerja dari berbagai tipe dan jenis *database* didapatkan hasil penggunaan MySQL menunjukkan hasil kinerja yang bagus dalam hal waktu eksekusi permintaan, dengan sistem penyimpanan data bersifat relasional dan terstruktur maka MySQL dapat diterapkan pada sistem penguncian pintu gedung ini. Diagram ER (entity relation) dari database pada sistem penguncian gedung ini dapat dilihat pada gambar 3.2.



**Gambar 3.2** ERD Database

Pada gambar 3.2 terlihat relasi antar entitas didalam *database* sistem penguncian pintu gedung. Pada database sistem penguncian pintu gedung terdapat beberapa entitas seperti pengguna, pintu, gedung dan *log* akses, entitas tersebut digunakan untuk meyimpan data yang akan digunakan didalam pengelolaan sistem dengan isi data sesuai dengan atribut dari masing-masing entitas. Dialam sistem database juga terdapat relasi, relasi menunjukkan hubungan antar entitas seperti menempati, memiliki, menjadwalkan dan mengakses, khusus untuk relasi menjadwalkan dan mengakses pada gambar 3.2 terlihat bahwa relasi tersebut memiliki aktribut yang menandakan bahwa relasi tersebut juga bertindak sebagai entitas yang menjadi penghubung antara 2 entitas yang berbeda, pada umumnya kondisi seperti ini berarti relasi tersebut menunjukkan kardinalitas *many-to-many* dimana relasi tersebut bertindak sebagai entitas pivot untuk menghubungkan relasi *many-to-many* dari 2 entitas lainnya.

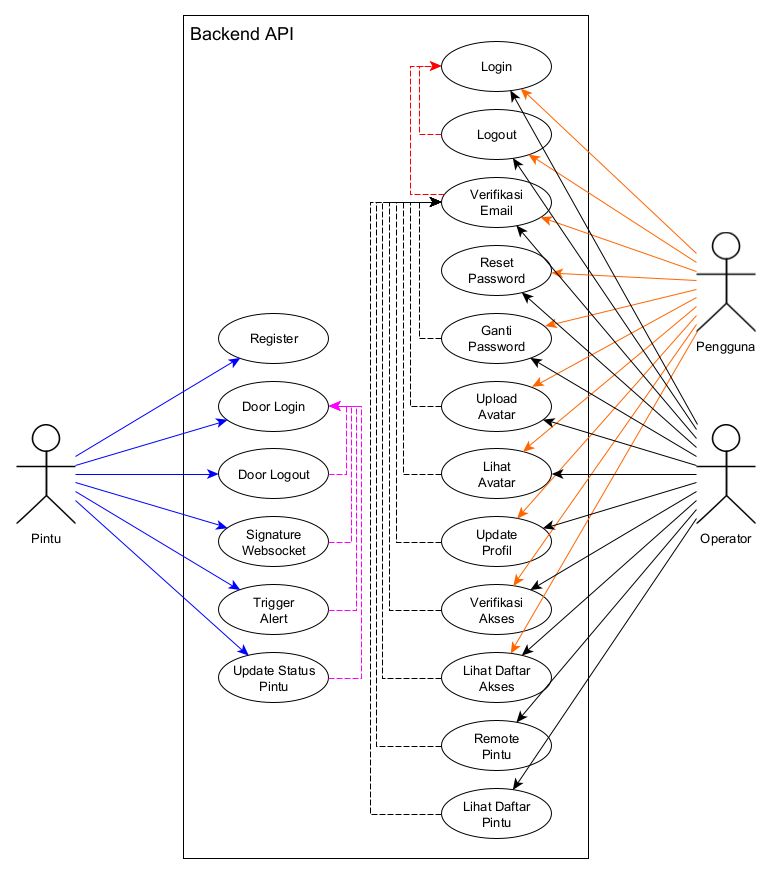
1. Perancangan *Backend* API

*Backend* API digunakan sebagai salah satu jalur komunikasi yang menghubungkan antara perangkat *mobile* dan perangkat kunci pintu berbasis IoT dengan *server*. Selain terdapat *website* yang digunakan sebagai antarmuka utama dalam mengelola sistem ini juga terdapat aplikasi berbasis mobile yang digunakan untuk menunjang kinerja dari sistem terutama pada bagian yang membutuhkan teknologi yang belum disediakan oleh *browser* seperti koneksi *bluetooth* dan pindai kode QR menggunakan kamera.

*Backend* API yang dikembangkan menggunakan bahasa pemrograman PHP dengan menggunakan kerangka kerja dari Laravel. Laravel menyediakan beberapa fitur yang siap digunakan seperti autentikasi *login*, reset *password*, notifikasi email dan lain sebagainnya sehingga mempermudah dan mempercepat proses pengkodean.

1. Diagram *Use Case* API

Tidak semua fitur yang ada pada website diimplementasikan dalam bentuk API, hal ini berkaitan dengan kompleksitas dari sistem yang dikembangkan, fitur yang berkaitan dengan manajemen penguncian seperti menambah pintu baru, membuat jadwal, menambahakan pengguna baru, memberikan akses ke pengguna dan lain sebagainya hanya bisa dilakukan melalui website hal ini dimaksudkan untuk mempertahankan kontrol yang lebih ketat terhadap akses dan perubahan yang dilakukan pada sistem. Dengan membatasi akses langsung melalui API, administrator sistem dapat memastikan bahwa hanya proses yang telah melalui lapisan keamanan dan validasi yang benar yang dapat mempengaruhi manajemen penguncian dan pengaturan lainnya. Diagram *Use* *Case* dari *Backend* API dapat dilihat pada gambar 3.3 dibawah.



**Gambar 3.3** Diagram Use Case API

Dapat dilihat pada gambar 3.3 diatas, terdapat 3 aktor yang berinteraksi dengan sistem melalui API yaitu pintu, pengguna dan operator. Pintu merupakan perangkat IoT yang digunakan untuk melakukan penguncian pada gedung, sedangkan pengguna dan operator merupakan aplikasi mobile yang digunakan sebagai antarmuka sistem.

Pada gambar 3.3 juga terlihat beberapa fungsi seperti *update* profil, lihat avatar, verifikasi akses dan lain sebagainya merupakan *extends* dari fungsi verifikasi email yang menandakan bahawa fungsi-fungsi tersebut hanya bisa diakses jika *client* sudah melakukan proses verifikasi email. Fungsi verifikasi email juga merupakan *extends* dari fungsi login sehingga fungsi verifikasi email hanya bisa diakses jika client sudah melakukan autentikasi login. Hal ini dimaksudkan supaya hanya client yang terautentikasi dan terverifikasi yang dapat mengakses sumber daya yang disediakan.

1. *Endpoint* API

Dalam perancangan *Backend* API ini terdapat dua bagian API yang digunakan, yaitu API untuk aplikasi mobile dan API untuk perangkat kunci pintu IoT dengan penjabaran setiap *endpoint* dan fungsinya dapat dilihat pada tabel 3.1 diabawah.

**Tabel 3.3** *Endpoint* API

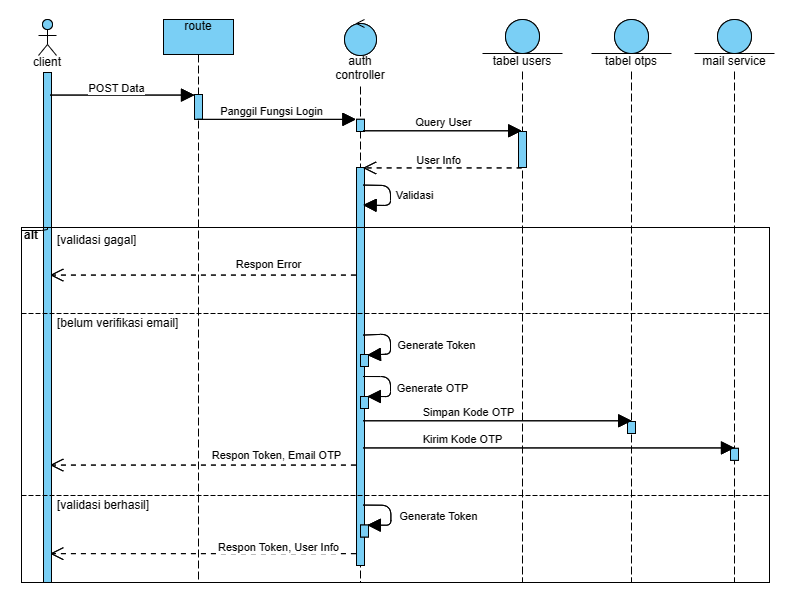
|  |  |  |  |
| --- | --- | --- | --- |
| **Tipe** | ***Endpoint*** | ***Auth*** | **Keterangan** |
| POST | /api/login | - | login pengguna dan operator |
| POST | /api/reset-password | - | reset password menggunakan email |
| POST | /api/verify-email | sanctum | verifikasi email |
| GET | /api/logout | sanctum | logout pengguna dan operator |
| POST | /api/update-profile | sanctum, verified | update profile nama, email, dan lainnya |
| GET | /api/avatar | sanctum, verified | mengambil gambar avatar |
| POST | update-avatar | sanctum, verified | mengubah gambar avatar |
| POST | /api/change-password | sanctum, verified | mengubah password pengguna atau operator |
| GET | /api/my-access | sanctum, verified | mengambil daftar akses |
| GET | /api/get-doors | sanctum, verified | mengambil daftar pintu |
| GET | /api/my-history | Sanctum, verified | mengambil daftar riwayat akses |
| GET | /api/verify-access/{door\_id} | sanctum, verified | verifikasi akses dari kode QR pintu |
| POST | /api/remote-access | sanctum, verified | membuka atau mengunci pintu jarak jauh |
| POST | /door/login | - | login perangkat kunci pintu |
| POST | /door/register | - | menambahkan perangkat kunci pintu baru |
| GET | /door/logout | sanctum | logout perangkat kunci pintu |
| POST | /door/get-signature | sanctum | mengambil kode signature channel pusher |
| POST | /door/update-status | sanctum | update status pintu |
| POST | /door/alert | sanctum | peringatan pada pintu |

Dapat dilihat pada tabel 3.1 diatas, API terbagi menjadi 2 bagian yang ditandai dengan awalan (*prefix*) “/api” dan “/door”. Awalan “/api” merupakan *endpoint* API yang khusus ditujukan untuk penggunaan aplikasi mobile pengguna dan operator sedangkan awalan “/door” merupakan API yang ditujukan untuk perangkat kunci IoT sehingga dengan menggunakan API yang terpisah maka API akan semakin terorganisir dengan baik.

Pada tabel 3.1 juga terlihat bahwa beberapa *endpoint* memerlukan autentikasi sanctum dan *verified*. Autentikasi sanctum merupakan sebuah metode autentikasi berbasis token yang digunakan untuk mengamankan sumber daya API dari *client* dengan hanya mengijinkan pengguna yang sudah terautentikasi yang dapat mengakses API tersebut. Sedangkan *verified* merupakan autentikasi tambahan yang digunakan untuk memastikan bahwa client (pengguna dan operator) sudah melakukan verifikasi email sehingga dapat meningkatkan keamanan API.

1. API *Login*

API *login* digunakan untuk melakukan autentikasi *client* melalui *username* dan *password* yang dikirimkan, API *login* juga memeriksa apakah pengguna dan operator sudah melakukan verifiaksi email, jika belum maka *login* akan tertahan sampai pengguna melalukan verifikasi email. Diagram fungsi login dapat dilihat pada gambar 3.4 dibawah.

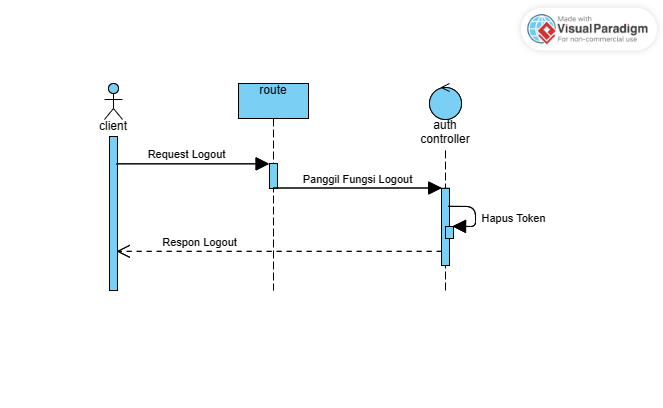


Gambar 3. *Sequence* Diagram API *Login*

API *login* dimulai oleh *client*, *client* akan mengirimkan *username* dan *password* mereka melalui *endpoint* “/api/login” kemudian *route* akan memanggil fungsi *login* didalam kontroler, kontroler akan memeriksa *username* dan *password* dengan melakukan *query* ke tabel *users*, jika cocok maka kontroler akan membuat token menggunakan modul sanctum. Sanctum adalah sebuah paket autentikasi dan autorisasi yang disediakan oleh Laravel yang dirancang untuk memudahkan implementasi otentikasi API yang sederhana namun aman pada aplikasi Laravel. Setelah mendapatkan token kemudian kontroler akan mengembalikan token tersebut disertai dengan data *client* seperti nama, email, nomor hp dan lain sebagainya. Jika terdeteksi *client* belum malakukan verifikasi email maka kontroler akan membuat kode OTP atau *One* *Time* *Password* yang merupakan 6 digit angka acak dan mengirimkan kode terkesebut ke email *client*. Jika autentikasi yang dilakukan gagal, maka kontroler akan mengembalikan respon error ke *client*.

1. API *Logout*

Fungsi *logout* digunakan untuk mengakhiri sesi *login* pengguna dan operator dengan cara menghapus semua token yang dimilikinya. Diagram dari API *logout* dapat dilihat pada gambar 3.5 dibawah.

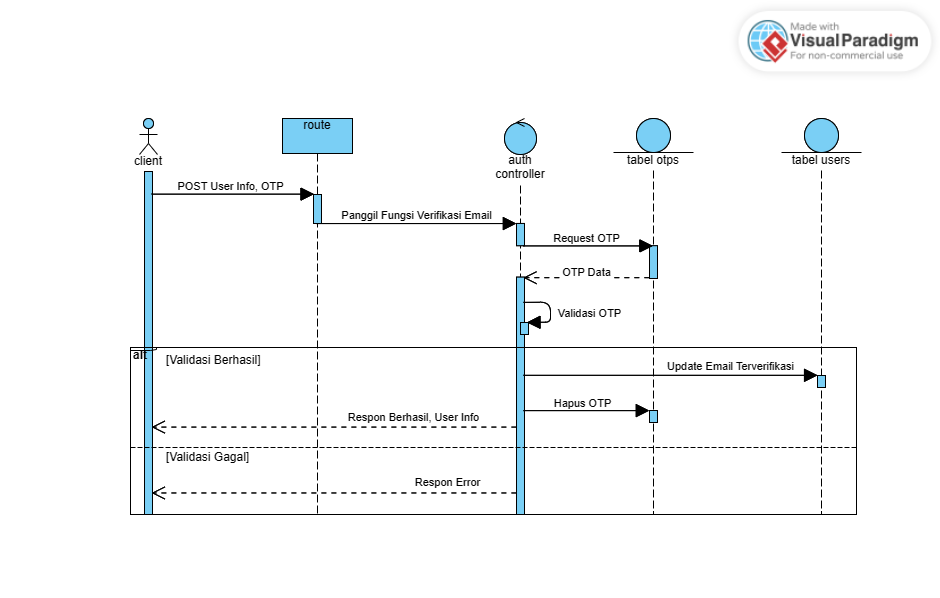


**Gambar 3.5** *Sequence* Diagram API *Logout*

Pada gambar 3.5 diatas, sebuah metode *logout* didalam kontroler akan dipanggil oleh *route* jika ada client yang melakukan *request* ke *endpoint* “/api/logout”. Kemudian kontroler akan menghapus token dari client sesuai dengan token yang dilampirkan didalam *header* pada saat *request* diterima. Proses ini sangat penting untuk menjaga keamanan sistem dan melindungi privasi pengguna. Dengan menghapus token saat klien keluar atau log out, sistem dapat memastikan bahwa akses tidak sah atau akses yang sudah tidak digunakan lagi tidak dapat digunakan kembali oleh pihak yang tidak berwenang.

1. API Verifikasi Email

API verifikasi email digunakan untuk memastikan bahwa pengguna dan operator memiliki email yang valid dan aktif, dengan adanya verifikasi email maka akan meningkatkan keamanan dengan hanya mengijinkan pengguna dan operator yang terpercaya untuk mengakses sumber daya yang ada. Verifikasi email dilakukan dengan cara mengirimkan kode OTP (*One Time Password*) ke email yang telah didaftarkan. Diagram dari API verifikasi email dapat dilihat pada gambar 3.6 dibawah.



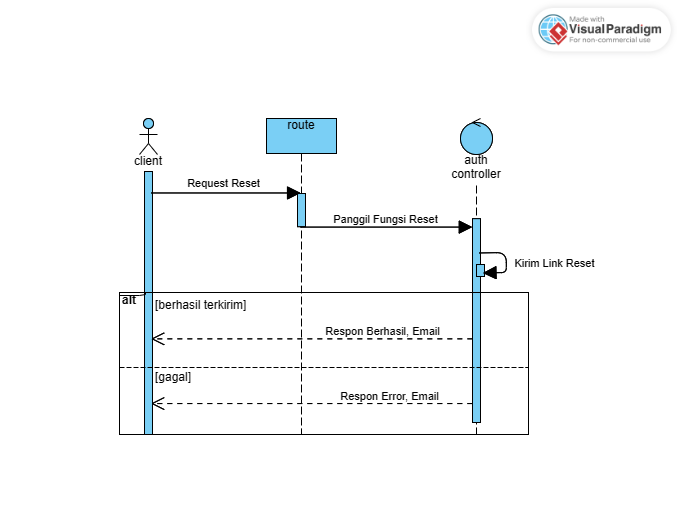
**Gambar 3.6** *Sequence* Diagram API Verifikasi Email

Pada gambar 3.6 diatas dapat dilihat bahwa pengguna melakukan verifikasi email dengan mengirimkan kode OTP yang sudah diterima melalui email disertai dengan detail client seperti id, nama dan email ke endpoint “/api/verify-email”, kemudian pada kontroler akan memeriksa kode yang diterima dengan kode yang tersimpan pada tabel otps, jika cocok dan masih aktif maka kontroler akan memperbarui status client menjadi terverifikasi dan menghapus kode otp yang lama kemudian mengembalikan respon berhasil. Jika kode salah atau sudah kadaluarsa maka kontroller akan mengembalikan respon error.

1. API Reset *Password*

API reset *password* adalah fitur penting dalam sistem keamanan yang memungkinkan pengguna dan operator untuk mengatasi masalah lupa *password* dengan cepat dan mudah. Dengan menggunakan API ini, pengguna dan operator dapat menghindari kehilangan akses akun dan data yang berharga.

API reset *password* digunakan oleh pengguna dan operator untuk memperbarui *password* jika meraka lupa dengan cara memasukkan email yang sudah terdaftar dan menggunakan *link* yang telah dikirimkan ke email untuk memperbarui *password*. Diagram dari API reset *password* dapat dilihat pada gambar 3.7 dibawah.



**Gambar 3.7** *Sequence* Diagram API Reset *Password*

Pada gambar 3.7 di atas, proses reset *password* menggunakan metode yang terstruktur dan efisien. *Client* memulai proses ini dengan melakukan permintaan (*request*) ke *endpoint* "/api/reset-password" pada *server*. Permintaan ini berisi data email yang sudah terdaftar sebagai parameter supaya *server* dapat mengidentifikasi akun yang akan direset *password*-nya.

Setelah permintaan tersebut diterima maka kontroler pada sisi server akan mengeksekusi sebuah fungsi khusus yang telah disediakan oleh Laravel. Fungsi tersebut bertugas untuk mengirimkan *link* reset password ke alamat email yang diberikan oleh *client*. Proses pengiriman email ini dapat penggunaan layanan email eksternal seperti SMTP yang telah dikonfigurasi pada *server*. Jika email berhasil terkirim dengan sukses maka kontroler akan memberikan respon ke client berupa pesan berhasil. Pesan ini memberitahukan bahwa email reset *password* telah berhasil dikirim dan pengguna dapat segera memeriksa kotak masuk email mereka untuk melanjutkan proses selanjutnya. Namun, dalam beberapa situasi, pengiriman email mungkin mengalami kegagalan. Misalnya, alamat email yang diberikan tidak valid, terjadi gangguan jaringan, atau layanan email eksternal mengalami masalah. Jika hal ini terjadi, kontroler akan memberikan respon error ke client. Pesan error ini berisi informasi yang relevan tentang masalah yang terjadi, sehingga pengguna atau operator dapat mengetahui penyebabnya.

1. API Ganti Password

API ganti password digunakan oleh pengguna dan operator untuk mengganti password mereka melalui aplikasi mobile. Berbeda dengan reset *password*, API ganti *password* digunakan untuk mengganti *password* pengguna tanpa melalui link yang dikirimkan ke email. Diagram dari API ganti password dalap dilihat pada gambar 3.8 dibawah.

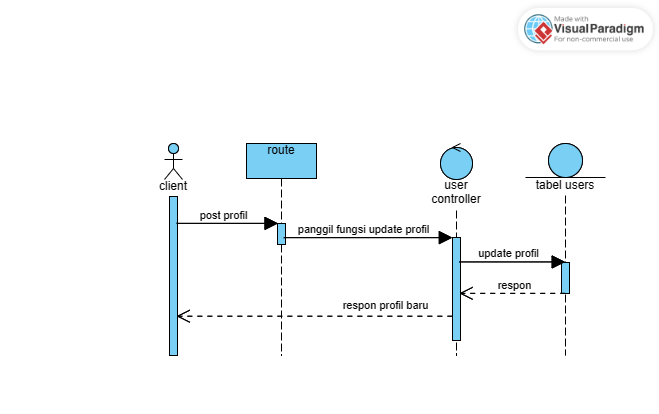


**Gambar 3.8** *Sequence* Diagram API Ganti Password

Dapat dilihat pada gambar 3.8 diatas, untuk mengganti *password* pertama client mengirimkan permintaan ganti password ke enpoint “/api/change-password” dengan mengirimkan *password* lama, *password* baru dan konfirmasi *password* baru, kemudian didalam kontroler *password* lama yang dikirimkan akan dicocokkan dengan *password* client sekarang dengan menggunakan fungsi Hash. Fungsi Hash merupakan sebuah fungsi yang disediakan oleh laravel yang digunakan untuk pengolahan data yang berkaitan dengan enkripsi. Jika kedua *password* cocok maka kontroler akan memperbarui *password* pada tabel users dan mengmbalikan respon berhasil, jika *password* tidak sesuai maka kontroler akan mengembalikan respon error.

1. API Ganti Profil

API ganti profil digunakan untuk mengganti data pengguna dan operator seperti nama, email, nomor hp dan lain sebagainya melalui aplikasi *mobile*. Diagram dari API ganti profil dapat dilihat pada gamabr 3.9 dibawah.

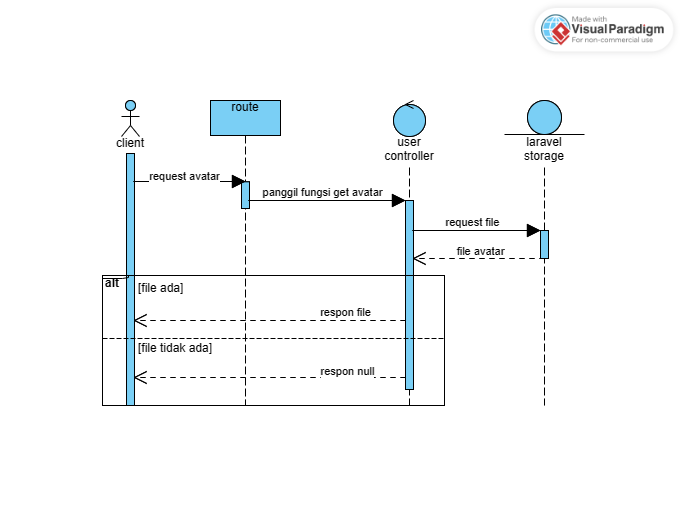


**Gambar 3.9** *Sequence* Diagram API Ganti Profil

Dapat dilihat pada gambar 3.9 diatas, unttuk mengganti profil pertama penguna atau operator mengirimkan data profil mereka ke enpoint “/api/update-profile” kemudian didalam kontroler data yang telah terima akan dimasukkan kedalam tabel *users* untuk memperbarui profil dan terakhir kontroler mengembalikan respon bahwa profil berhasil dirubah.

1. API Lihat Avatar

API lihat avatar digunakan untuk mendapatkan gambar avatar (foto profil) dari pengguna dan operator untuk ditampilkan didalam aplikasi mobile. Diagram dari API lihat avatar dapat dilihat pada gambar 3.10 dibawah.

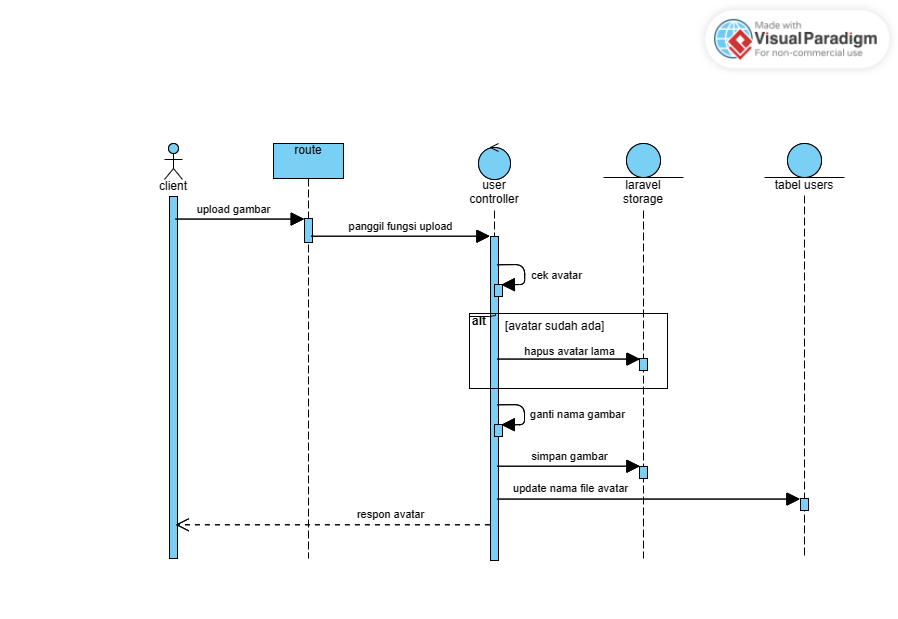


**Gambar 3.10** *Sequence* Diagram API Lihat Avatar

Dapat dilihat pada gambar 3.10 diatas, untuk mendapatkan gambar avatar pertama pengguna atau operator melakukan permintaan data avatar melalui endpoint “/api/avatar” kemudian kontroler akan mengambil gambar avatar didalam penyimpanan Laravel (*storage*). *Storage* merupakan salah satu fitur yang disediakan laravel untuk melakukan peyimpana file seperti gambar, log dan berkas lainnya, dengan menggunakan *storage* kita bisa mengatur siapa saja yang boleh mengakses penyimpanan tersebut sehingga lebih aman. Jika file telah ditemukan maka kontroler akan mengembalikan respon file dan jika tidak ditemukan maka kontroler akan mengembalikan nilai null atau kosong.

1. API Ganti Avatar

API ganti avatar digunakan untuk mengganti gambar avatar pengguna atau operator, pengguna dapat mengganti gambar avatar melalui aplikasi mobile. Diagarm dari API ganti avatar dapat dilihat pada gambar 3.11 dibawah.

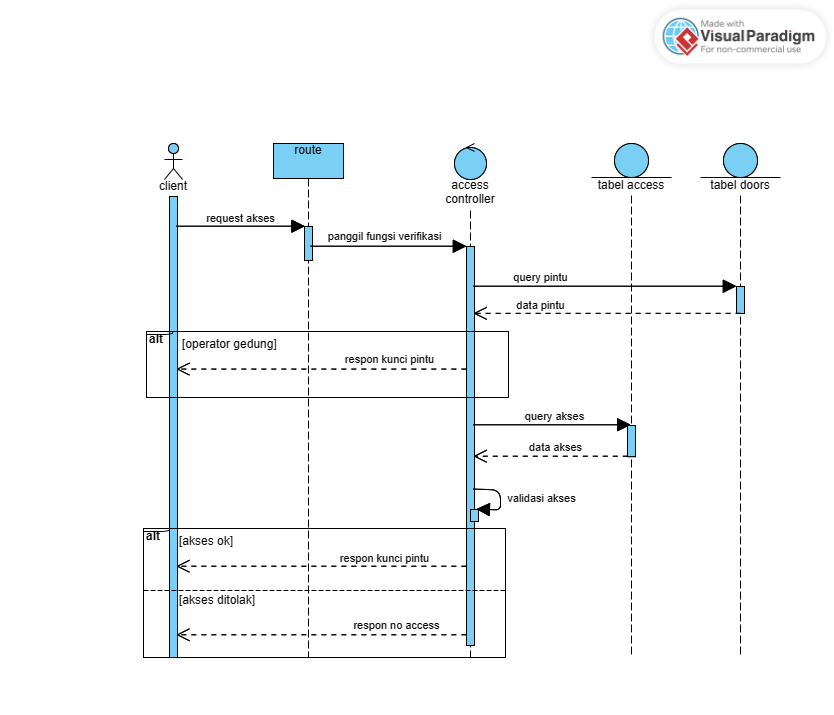


**Gambar 3.11** *Sequence* Diagram API Ganti Avatar

Dapat dilihat pada gambar 3.11 diatas, untuk mengganti avatar pertama pengguna atau operator akan mengirimakan file avatar yang baru melalui *endpoint* “/api/update-avatar”, kemudian kontroler akan melakukan pemeriksaan apakah sebelumnya pengguna atau operator sudah memiliki gambar avatar, jika sudah maka file avatar sebelumnya akan dihapus dari *storage*, kemudian file avatar yang baru akan disesuaikan namanya sesuai dengan format *server*, setelah namanya diganti kemudian file akan disimpan didalam *storage* dan kontroler juga akan memperbarui nama avatar didalam tabel *users*, terakhir kontorler akan mengembalikan respon yang menginformasikan avatar telah diganti.

1. API Verifikasi Akses

Verifikasi akses digunakan oleh pengguna dan operator untuk memverifikasi diri mereka dan untuk mendapatkan akses terhadap suatu pintu dengan cara memindai kode QR dengan perangkat mobile. Diagram dari API verifikasi akses dapat dilihat pada gambar 3.12 dibawah.

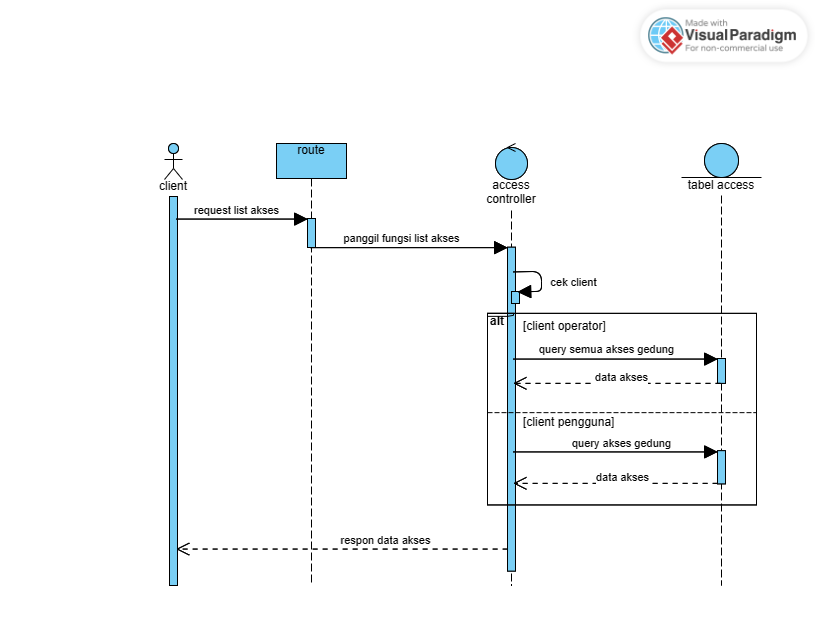


**Gambar 3.12** *Sequence* Diagram API Verifikasi Akses

Untuk mendapatkan akases ke pintu, setelah memindai kode QR pada pintu maka pengguna atau operator akan mendapatkan data informasi terkait pintu tersebut, kemudian data tersebut dikirimkan ke server melalui endpoint “/api/verify-access/{door-id}”, kemudian kontroler akan mengambil data pintu pada tabel *doors*, jika permintaan berasal dari operator gedung dimana pintu tersebut berada maka kontroler akan mengijinkan dengan mengambalikan respon berupa kode kunci pintu. Jika pengguna permintaan akses dilakukan oleh pengguna biasa maka kontroler akan memeriksa daftar akses didalam tabel *access*, jika pengguna memiliki akses dan masil berlaku maka kontroler akan mengembalikan respon akses diijinkan dan mengirimkan kode kunci untuk membuka pintu.

1. API Daftar Akses

API daftar akses digunakan untuk mendapatkan daftar dari pintu mana saja yang dapat diakses oleh pengguna, sehingga pengguna dapat mnegetahui pintu mana saja yang bisa diakses oleh dirinya. Diagram dari API dafar akses dapat dilihat pada gambar 3.13 dibawah.

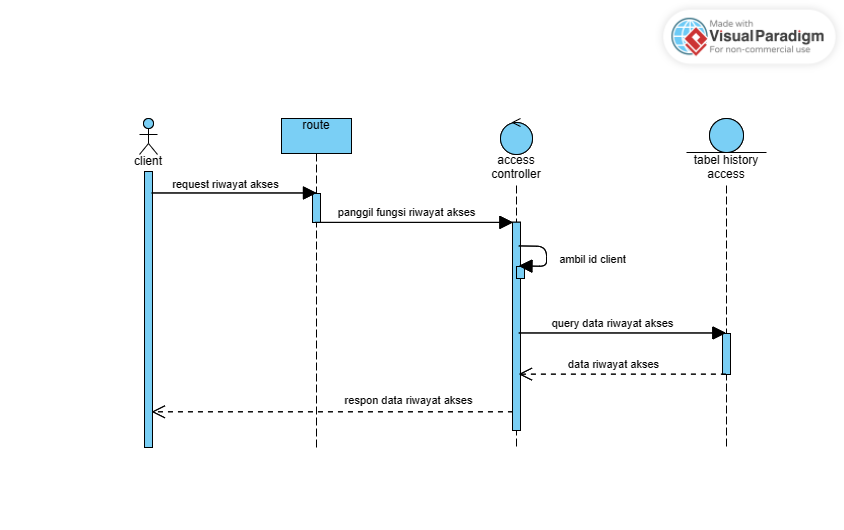


**Gambar 3.13** *Sequence* Diagram API Daftar Akses

Dapat dilihat pada gambar 3.13 diatas, untuk mendapatkan daftar akses yang dimiliki pengguna mengirimkan request endpoint “/api/my-access”, kemudian kontroler akan melakukan pemeriksaan apakah permintaan datang dari seorang operator atau pengguna biasa. Jika permintaan berasal dari seorang operator maka kontroler akan mengembalikan semua akses pintu atau semua daftar pintu yang ada, namun jika permintaan datang dari pengguna biasa maka kontroler akan mengembalikan data akses sesuai dengan akses yang dimiliki pengguna tersebut.

1. API Riwayat Akses

API riwayat akses digunakan untuk memberikan informasi pencatatan akses pengguna, riwayat akses pengguna dapat dilihat pada masing-masing aplikasi *mobile* pengguna. Diagram dari API riwayat akses dapat dilihat pada gambar 3.14 dibawah.

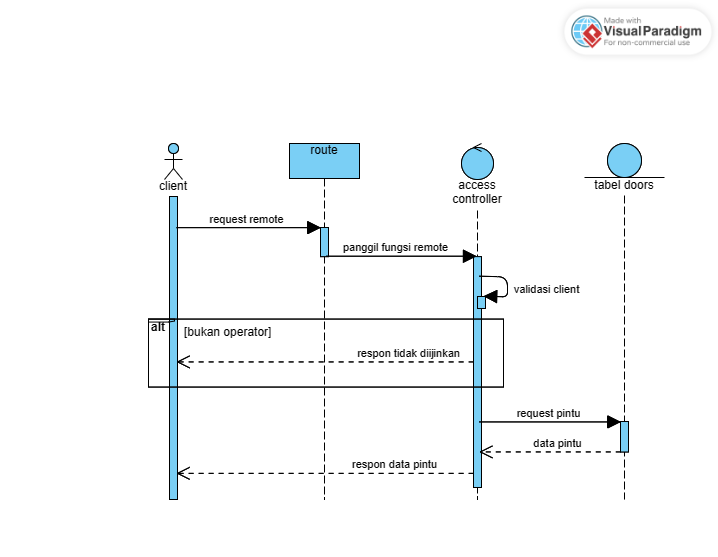


**Gambar 3.14** *Sequence* Diagram API Riwayat Akses

Dapat dilihat pada gambar 3.14 untuk mendapatkan data riwayat akses maka pengguna akan melakukan permintaan ke *enpoint* “/api/my-history”. Kemudian kontroller akan mengambil data riwayat akses pengguna dengan melakukan *query* ke databse dengan menyertakan identitas pengguna sebagai parameter. Selanjutnya data yang sudah diperoleh seperti nama pintu, mana gedung, waktu aktifitas dan jenis aktifitas akan dikirimkan kembali ke pengguna sebagai respon untuk diolah pada tampilan aplikasi *mobile*.

1. API Daftar Pintu

API daftar pintu digunakan oleh operator untuk menampilkan daftar pintu yang ada pada suatu gedung, sehingga operator dapat mengetahui jumlah dan status pada setiap pintu. Diagram dari API daftar pintu dapat dilihat pada gambar 3.15 dibawah.

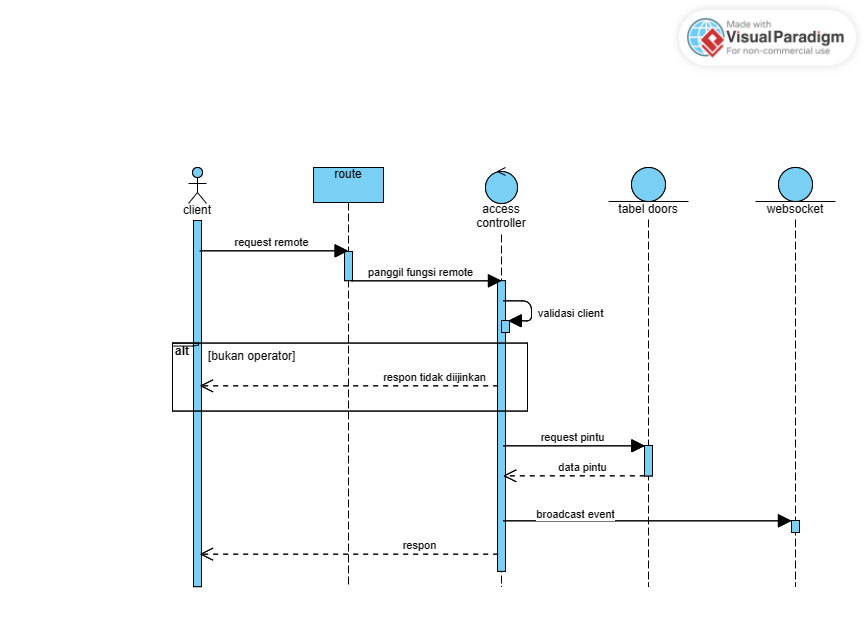


**Gambar 3.15** *Sequence* Diagram API Daftar Pintu

Dapat dilihat pada gambar 3.15 diatas, untuk mendapatkan daftar pintu maka operator akan melalakukan request ke endpoint “/api/get-door” kemudian didalam kontroler dilakukan pengecekan untuk memastikan permintaan hanya berasal dari operator, jika permintaan berasal dari pengguna biasa maka kontroler akan mengembalikan respon tidak diijinkan. Selanjutnya kontroler akan mengambil data pintu pada tabel *doors* dan mengirimkan rebagai respon ke operator.

1. API *Remote* Pintu

API *remote* pintu digunakan oleh operator untuk membuka atau mengunci pintu secara jarak jauh melalui aplikasi *mobile*. Dengan adanya fitur ini operator dapat mengendalikan pintu melaui aplikasi *mobile* dimana saja dan kapan saja tanpa harus berapa di ruangan operasional dengan menggunaka komputer. Diagram dari API *remote* pintu dapat dilihat pada gambar 3.16 dibawah.

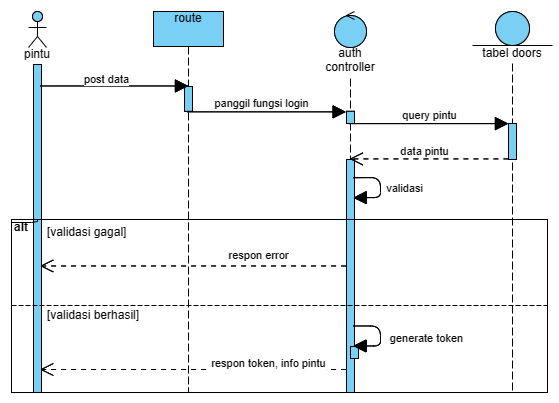


**Gambar 3.16** *Sequence* Diagram API Remote Pintu

Dapat dilihat pada gamabar 3.16 diatas, untuk melakukan *remote* pintu pertama operator akan melakukan *request* ke *endpoint* “/api/remote-access” dengan mengirimkan identitas dari pintu yang akan dikendalikan, kemudian kontroler akan memeriksa *client* untuk memastikan permintaan berasal dari operator, jika barasal dari pengguna biasa maka kontroler akan mengeembalikan respon tidak diijinkan. Selanjutnya kontroler akan mengambil data pintu untuk melengkapi informasi yang dibutuhkan seperti kode kunci, kode gedung, token dan lain sebagainya, kemudian data akan disiarkan ke perangkat kunci pintu melalui koneksi websocket yang sudah terhubung, terakhir kontroler akan mengembalikan respon remote pintu telah dilaksanakan.

1. API *Door* *Login*

API *door* *login* digunakan untuk proses autentikasi perangkat kunci pintu, berbeda dengan *login* pada pengguna dan operator pada *login* pintu hanya terdapat pengecekan *username* dan *password* saja tanpa ada pengeceka verifikasi email dan lainnya hal ini dikarenakan pada perangkat kunci pintu tidak menggunakan email sebagai identitasnya. Diagram dari API door login dapat dilihat pada gambar 3.17 dibawah.

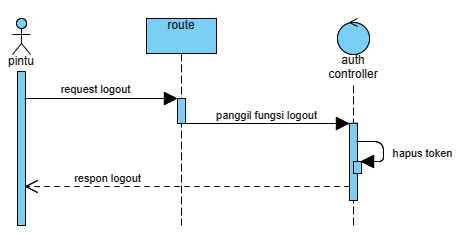


**Gambar 3.17** *Sequence* Diagram API *Door* *Login*

Dapat dilihat pada gambar 3.17 diatas, perangkat kunci pintu akan mengirimkan data *username* dan *password* melalui endpoint “/door/login”, kemudian kontroler akan memeriksa dan membandingkan dengan data pada tabel *doors*, jika data sesuai maka kontroler akan membuat token baru menggunakan sanctum dan mengembalikan respon token yang menandakan *login* berhasil, jika data tidak sesuai maka kontroler akan mengembalikan respon error.

1. API *Door Logout*

API *door logout* digunakan untuk mengakhiri sesi *login* perangkat kunci pintu dengan cara menghapus semua token yang dimilikinya. Diagram dari API door logout dapat dilihat pada gambar 3.18 dibawah.

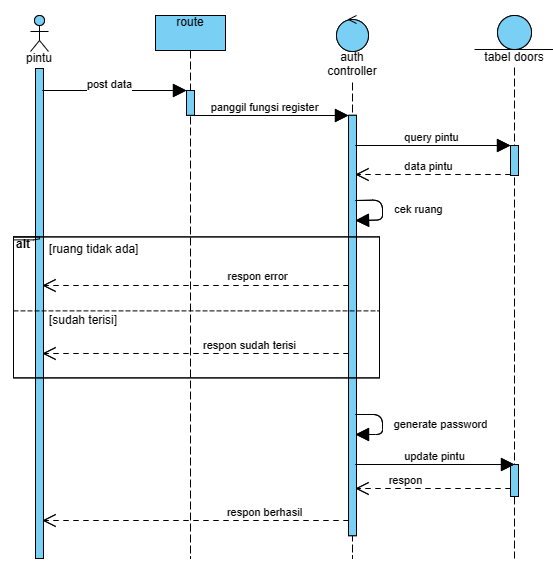


**Gambar 3.18** *Sequence* Diagram *API* *Logout*

Pada gambar 3.18 diatas, sebuah metode logout didalam kontroler akan dipanggil oleh route jika ada perangkat kunci pintu yang melakukan request ke endpoint “/door/logout”. Kemudian kontroler akan menghapus token dari perangkat kunci pintu sesuai dengan token yang dilampirkan didalam header pada saat request diterima. Terakhir kontroler akan mengembalikan respon logout.

1. API *Door* *Register*

API *door* *register* digunakan untuk menambahkan perangkat penguncian yang baru kedalam pintu. Pada saat operator menambahakan pintu baru melalui *dashboard* *website* operator maka pintu tersebut belum terpasang perangkat kunci pintu sehingga harus ditambahkan secara manual melalui prosedur pendaftaran. Diagram dari API door register dapat dilihat pada gambar 3.19 dibawah.

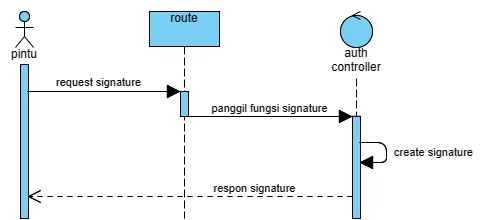


**Gambar 3.19** *Sequence* Diagram API *Door* *Register*

Dapat dilihat pada gambar 3.19 diatas, untuk menambahkan perangkat penguncian baru maka perangkat kunci pintu akan mengirimkan data nama perangkat yang akan didaftarkan dan kode pintu yang akan ditempati melalui endpoint “/door/register” kemudian kontroler akan mengambil data pada tabel *doors* untuk memastika bahwa pintu masih kosong, jika pintu sudah ada perangkat pengunciannya maka kontroler akan mengembalikan respon sudah terisi dan kontroler akan mengembalikan respon error jika pintu yang dituju tidak ditemukan. Selanjutnya kontroler akan membuat *password* acak yang akan digunakan untuk proses autentikasi, dengan menggunakan metode ini maka akan meningkatkan keamanana karena *password* bersiftar acak dan hanya diketahui oleh perangkat kunci pintu dan server. Terakhir kontroler akan mengembalikan respon berhasil direstai dengan password tadi untuk disimpan pada perangkat kunci pintu.

1. API *Door* *Signature*

API *door* *signature* digunakan untuk mendapatkan kode unik yang digunakan untuk melakukan *subcribe* ke *channel* pusher. Pusher merupakan sebuah protokol komunikasi notifikasi yang dibangun menggunakan websocket, didalam pusher terdapat *channel*-*channel* yang dapat di-*subcribe* yang nantinya client dapat menunggun *event* yang disiarkan melalui *broadcast* pada setiap *channel*. Diagram dari API door signature dapat dilihat pada gambar 3.20 dibawah.



**Gambar 3.20** *Sequence* Diagram API *Door* *Signature*

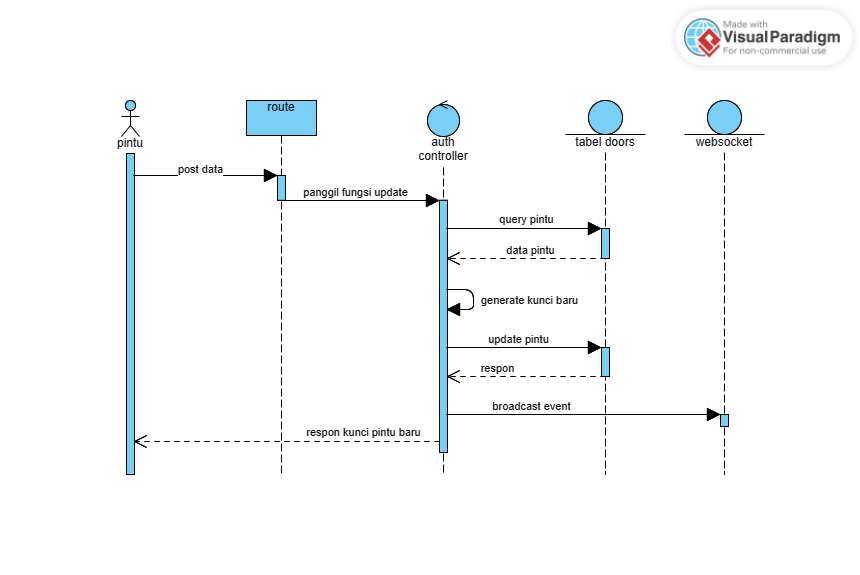
Dapat dilihat pada gamabr 3.20 diatas, untuk mendapatkan kode *signature* pusher pertama perangkat kunci pintu melakukan permintaan ke endpoint “/door/get-signature” dengan mengirimkan data-data seperti *socket*-id, *office*-id dan *channel*-data, dari data tersebut kemudian kontroler akan membuat kode *signature* menggunakan metode yang ada pada protokol pusher yaitu :

sign = hash\_hmac('sha256','socket:channel:data','secret\_key')

dimana ‘*secret*\_*key*’ merupakan kunci pusher yang terdapat pada server. Setelah mendapatkan nilai signature kemudian kontroler akan megembalikan respon kode *signature* ke perangkat kunci pintu.

1. API *Door* *Update* *Status*

API *door* *update* *status* digunakan oleh perangkat kunci pintu untuk memperbarui status pintu seperti pintu terbuka, pintu terkunci atau pintu terkoneksi. Pada setiap proses updata ini kunci pintu juga akan diperbarui sehingga meningkatkan keamanan karena kode kunci selalu berubah secara dinamis. Diagram dari API door update status dapat dilihat pada gambar 3.21 dibawah.

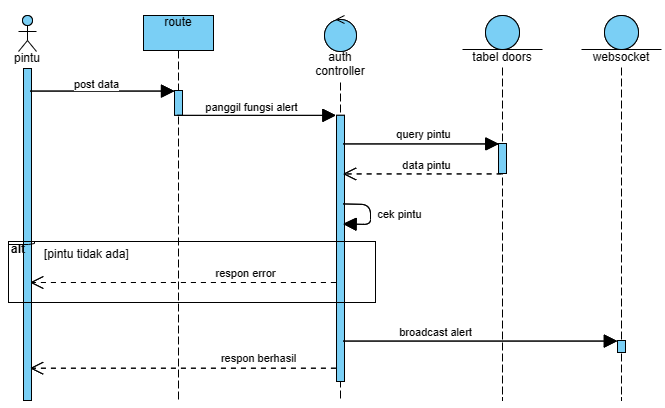


**Gambar 3.21** *Sequence* Diagram API *Door* *Update* *Status*

Terlihat pada gambar 3.21 diatas, untuk melakukan update status pertama perangkat kunci pintu akan mengirimkan data-data seperti status penguncian dan id *socket* melalui *endpoint* “/door/update-status”, kemudian kontroler akan mengambil data pintu yang terkait untuk diperbarui menggunakan data status dan kode kunci yang baru, kemudian kontroler juga akan melakukan *broadcasting* untuk menyiarkan bahwa status pintu berubah sehingga setiap informasi perubahan dapat tersampaikan secara langsung, terakhir kontroler akan mengembalikan respon update status telah dilaksanakan.

1. API *Door Alert*

API *door* *alert* digunakan oleh perangkat kunci pintu untuk memberikan peringatan kepada operator bahwa pintu dalam kondisi yang tidak aman seperti terbuka tanpa autentikasi yang sah. Diagram dari API *door* *alert* dapat dilihat pada gambar 3.22 dibawah.



**Gambar 3.22** *Sequence* Diagram API *Door* *Alert*

Dapat dilihat pada gambar 3.22 diatas, untuk memberikan peringatan pertama perangkat kunci pintu akan mengirimakan data-data seperti id-pintu, id-*office* dan status peringatan melalui *endpoint* “/door/alert” kemudian kontroler akan memeriksa pada tabel pintu untuk memastika bahwa pintu valid, jika pintu tudak ditemuakan maka kontroler akan mengembalikan respon error. Selanjutnya kontroler akan menyiarkan peringatan melalui websocket, terakhir kontroler akan mengembalikan respon peringatan sudah dilaksanakan.

1. Perancangan Penjadwalan

Penjadwalan digunakan untuk memeriksa data jadwal yang ada pada tabel schedules. Penjadwalan bekerja dengan menggunakan *kernel* yang disediakan oleh laravel, kernel ini nantinya akan dijalankan oleh laravel dengan menggunakan crontab yang diatur pada sistem operasi server. Crontab akan diatur untuk menjalankan perintah “*schedule*:*run*” setiap 1 menit sekali yang akan menjalankan fungsi pengecekan jadwal yang ada pada kernel.

1. Pengecekan Jadwal

Pengecekan jadwal dilakukan dengan cara memeriksa data jadwal pada tabel *schedules* setiap satu menit sekali. *Kernel* akan menjalankan *job* setiap satu menit sekali untuk memeriksa jadwal, jika ada jadwal yang harus dilaksanakan seperti membuka pintu atau mengunci pintu maka perintah akan dikirimkan ke perangkat kunci pintu melalui websocket. Diagram dari pengecekan jadwal dapat dilihat pada gambar 3.23 dibawah.

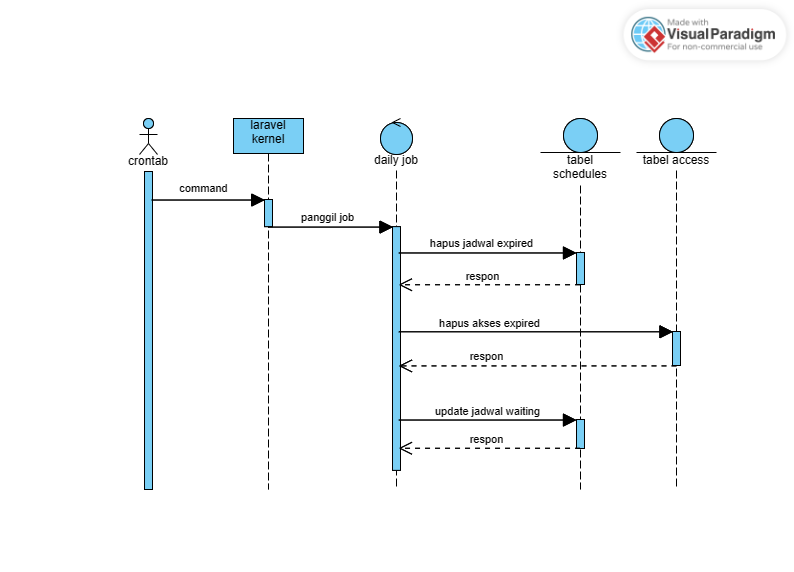


**Gambar 3.23** *Sequence* Diagram Pengecekan Jadwal

Terlihat pada gambar 3.23 diatas, *crontab* akan mengirimkan perintah ke *kernel* setiap 1 menit sekali dan *kernel* akan menjalankan sebuah *job* yang digunakan untuk memeriksa jadwal. Pertama *job* tersebut akan mengambil data jadwal hari ini yang akan dilaksanakan tentunya dengan menggunakan *query* untuk mengambil data yang sesuai dengan memperhatikan tanggal, waktu dan status jadwal, setelah data didapatkan kemudian job akan mengirimkan event ke setiap perangkat kunci pintu yang ada pada data jadwal tersebut, setelah semua event dikirimkan kemudian *job* akan memperbarui status jadwal menjadi “*running*” atau sedang berjalan. Terakhir job akan memperbarui semua jadwal yang sudah dijalankan dan sudah melewati waktu pelaksanaan menjadi “*done*” atau sudah selesai dilaksanakan.

1. Atur Ulang Jadwal

Setiap jadwal yang sudah dilaksanakan harus dihapus dari *database* dan jadwal yang berulang harus diatur statusnya seperti awal mula sehingga memerlukan proses atur ulang jadwal untuk membersihkan dan mengatur ulang jadwal. Jadwal akan dibersihkan dan diatur ulang setiap 24 jam sekali dengan menggunakan sebuah *job* pada *kernel* Laravel. Diagram dari *job* atur ulang jadwal dapat dilihat pada gambar 3.24 dibawah.



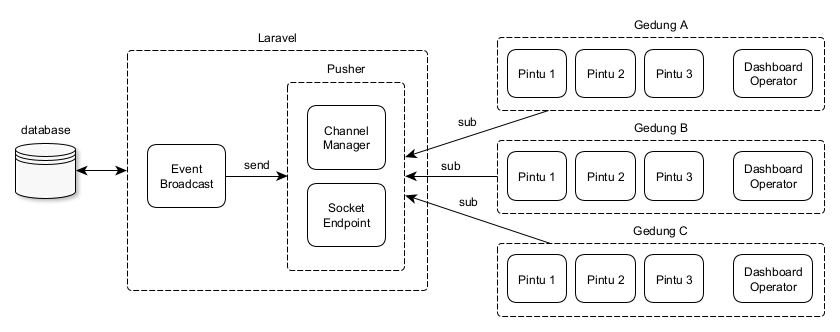
**Gambar 3.24** *Sequence* Diagram Atur Ulang Jadwal

Dapat dilihat pada gambar 3.24 diatas, *crontab* akan mengirimkan perintah ke *kernel* untuk menjalankan job atur ulang jadwal. Didalam *job* tersebut pertama *job* akan menghapus data jadwal yang sudah kadaluarsa pada tabel *schedules* yaitu jadwal yang sudah dilaksanakan dan tidak berulang, kemudian pada job ini juga akan menghapus kartu akses pengguna yang sudah kadaluarsa, kemudian job akan memperbarui setiap jadwal yang sudah pernah dilaksanakan menjadi “*waiting*” atau menunggu untuk dilaksanakan kembali.

1. Perancangan Komunikasi Websocket

Websocket digunakan sebagai jalur komunikasi yang menghubungkan antara *server* dengan perangkan kunci pintu, dengan adanya komunikasi websocket maka perangkat kunci pintu dan *server* akan selalu terhubung sehingga dapat berkomunikasi secara langsung.

Pada perancangan *backend* *server* untuk mendukung kinerja perangkat kunci pintu ini menggunakan pusher sebagai protokol komunikasi websocket yang menghubungkan antara perangkat kunci pintu dengan *server* dengan konfigurasi seperti yang terlihat pada gambar 3.25 dibawah.



**Gambar 3.25** Konfigurasi Websocket

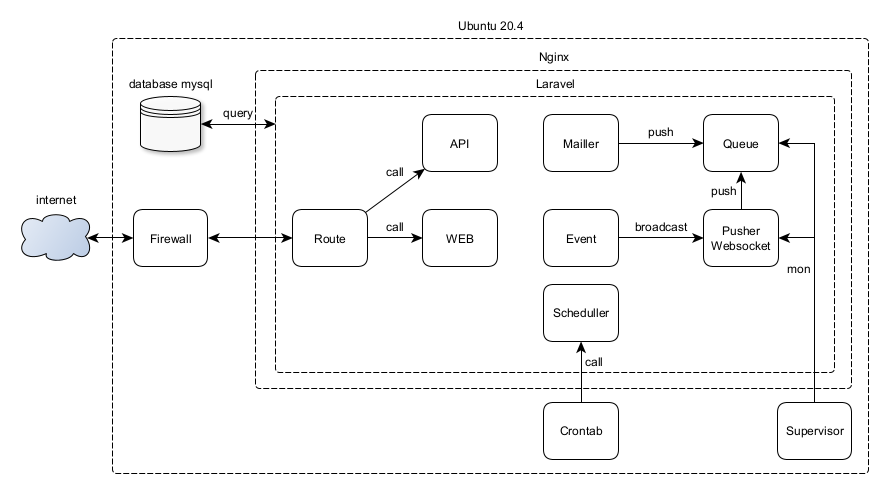
Dapat dilihat pada gambar 3.25 diatas, setiap pintu akan dikelompokkan berdasarkan dengan gedung dengan satu orang operator, setiap gedung akan memiliki satu *channel* pusher yang dapat di-*subscribe* oleh perangkat penguncian didalam gedung tersebut, untuk mengawasi semua aktifitas yang terhubung ke *channel* dengan mudah maka konfigurasi channel menggunakan *presence* *channel*, dengan menggunakan *presence* *channel* maka aktifitas semua perangkat kunci pintu seperti perangkat kunci melakukan *subscribe* dan koneksi terputus dapat diketahui secara langsung.

Laravel mengirimkan data atau perintah ke perangkat kunci pintu melalui pusher menggunakan sebuah *event*, setiap *event* seperti kunci atau buka pintu dipanggil baik melalui website maupun API maka *event* tersebut akan disiarkan ke *channel* sesuai dengan kode perintah dan *channel* yang dituju.

Didalam pusher sendiri terdapat 2 komponen utama yaitu *channel* *manager* dan *socket* *endpoint*, *channel* *manager* digunakan untuk mengelola semua channel yang ada serta mengelola semua *subsciber* dan melakukan *broadcast* *event*, sedangkan *socket* *endpoint* digunakan sebagai pintu masuk berupa URL yang digunakan untuk memulai koneksi dengan websocket.

1. Perancangan Server

Dengan menggunakan Laravel sebagai backend yang mengatur kinerja dari perangkat kunci pintu tentunya diperlukan sebuah *server*. *Server* ini akan bertindak sebagai pusat pengolahan data dan berfungsi untuk menerima permintaan dari perangkat kunci pintu, mengatur akses, memproses logika bisnis, dan berkomunikasi dengan *database*. Diagram dari *backend* *server* dapat dilihat pada gambar 3.26 dibawah.



**Gambar 3.26** Konfigurasi Server

Dapat dilihat pada gambar 3.26 diatas, *server* dibangun menggunakan sistem operasi ubuntu 20.04, ubuntu merupakan bagian dari sistem operasi linux yang biasa digunakan baik untuk perangkat dekstop maupun *server* karena *open* *source* dan ringan. Dengan menggunakan ubuntu 20.04 sebagai sistem operasi server, kita dapat memanfaatkan kestabilan, keamanan, dan dukungan jangka panjang untuk menjalankan aplikasi laravel dengan aman dan efisien.

Didalam sistem operasi ubuntu 20.04 dipasang nginx yang digunakan sebagai web *server* untuk menjalankan aplikasi laravel. Dengan menggunakan nginx maka aplikasi laravel dapat dijalankan dengan efisien dikarenakan nginx memiliki karakteristik ringan dan cepat. Dengan menggunakan nginx kita juga bisa membagi server menjadi beberapa blok yang dapat digunakan untuk menjalankan API dan websocket secara bersamaan. Pada nginx juga dipasang sertifikat SSL yang digunakan untuk mengenkripsi semua komunikasi dengan menggunakan HTTPS sehingga kemanan data akan terjamin dengan adanya jalur komunikasi yang terenkripsi.

Didalam ubuntu juga dipasang MySQL sebagai pusat penyimpanan data yang terhubung ke laravel, dengan menggunakan database yang berjalan pada server yang sama maka kecepatan transfer data akan sangat cepat dengan menghilangkan latensi jaringan hal ini sesuai dengan karakteristik sistem yang dibangun yaitu sistem cepat dan efisien.

Dengan adanya fitur penjadwalan otomatis didaiam sistem kunci pintu maka kita juga harus memasang crontab untuk menjalankan penjadwalan yang ada pada laravel, penjadwalan akan dipanggil setiap 1 menit sekali untuk memeriksa apakah ada jadwal yang harus dilaksanakan atau tidak.

Pada laravel juga mengimplementasikan fitur antrian yang bertujuan untuk meningkatkan kinerja dasi sistem sehingga fungsi antrian harus dijaga agar selalu berada dalam kondisi berjalan, oleh karena itu maka pada server suja dipasang supervisor yang digunakan untuk memantau kinerja dari antrian. Supervisor merupakan sebuah sistem pengawas atau *process* *control* *system* yang digunakan untuk mengontrol dan mengelola proses-proses yang berjalan didalam sistem operasi. Supervisor akan memastikan bahwa proses antrian pada laravel tetap berjalan secara terus-menerus dan akan memulai ulang proses jika terjadi kegagalan serta mengelola jumlah pekerja antrian atau *queue* *workers* yang berjalan secara paralel untuk meningkatkan efisiensi dan kinerja.

Supervisor juga digunakan untuk menjaga websocket berjalan secara terus menerus, websocket memegang peranan penting didalam sistem ini karena websocket menjadi jalur komunikasi yang menghubungkan perangkat kunci pintu dengan *server* pusat, sehingga jika terjadi gangguan pada kinerja websocket maka seluruh kinerja dari perangkat kunci pintu akan terganggu, oleh karena itu diperlukan pengawasan menggunakan supervisor untuk menjaga kinerja dari websocket.

BAB IV

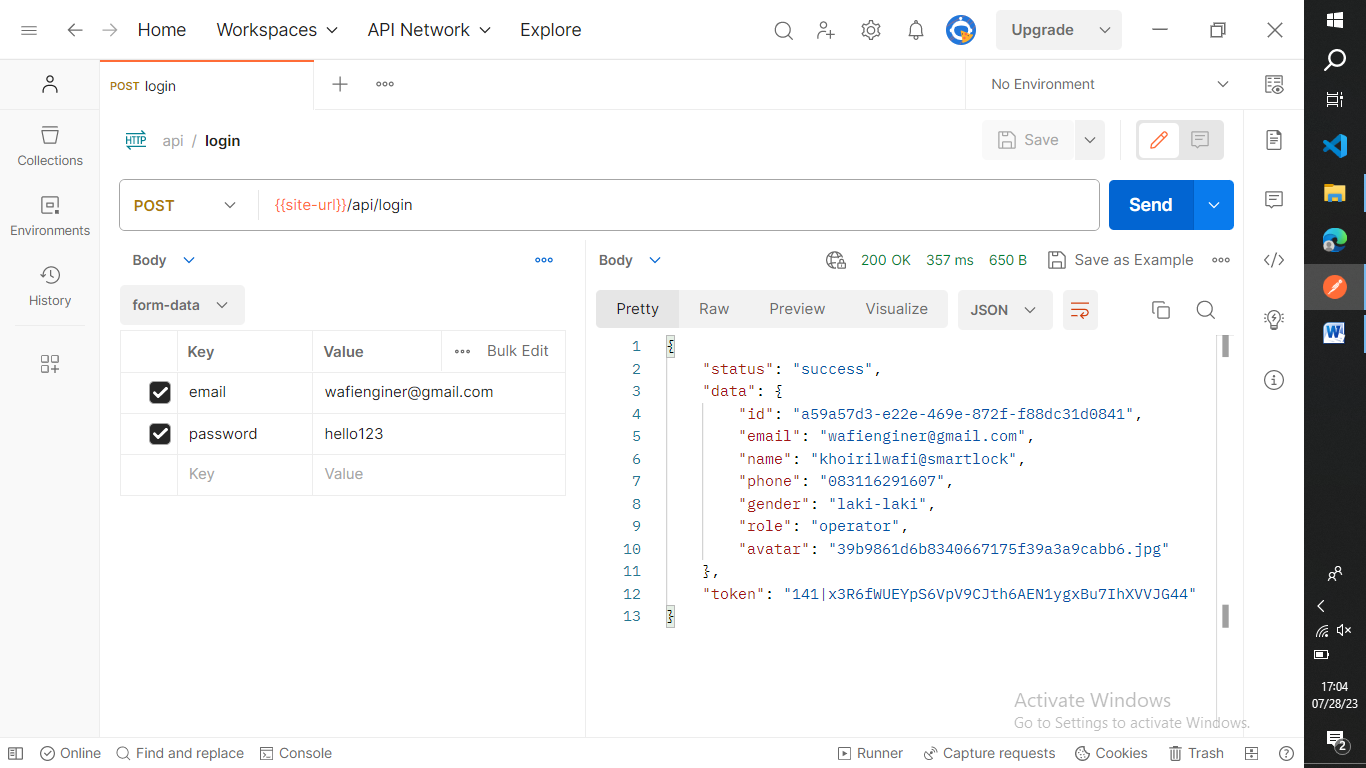
**HASIL DAN PEMBAHASAN**

1. Kajian Hasil Perancangan

Untuk melihat hasil dari perancangan API aplikasi mobile dan perangkat kunci pintu maka dilakukan uji coba menggunakan postman. Postman merupakan sebuah software yang digunakan untuk melakukan request ke API, dengan menggunakan postman maka akan terlihat hasil respon yang dikirimkan oleh server sesuai dengan permintaan *client* melalui *endpoint* tertentu.

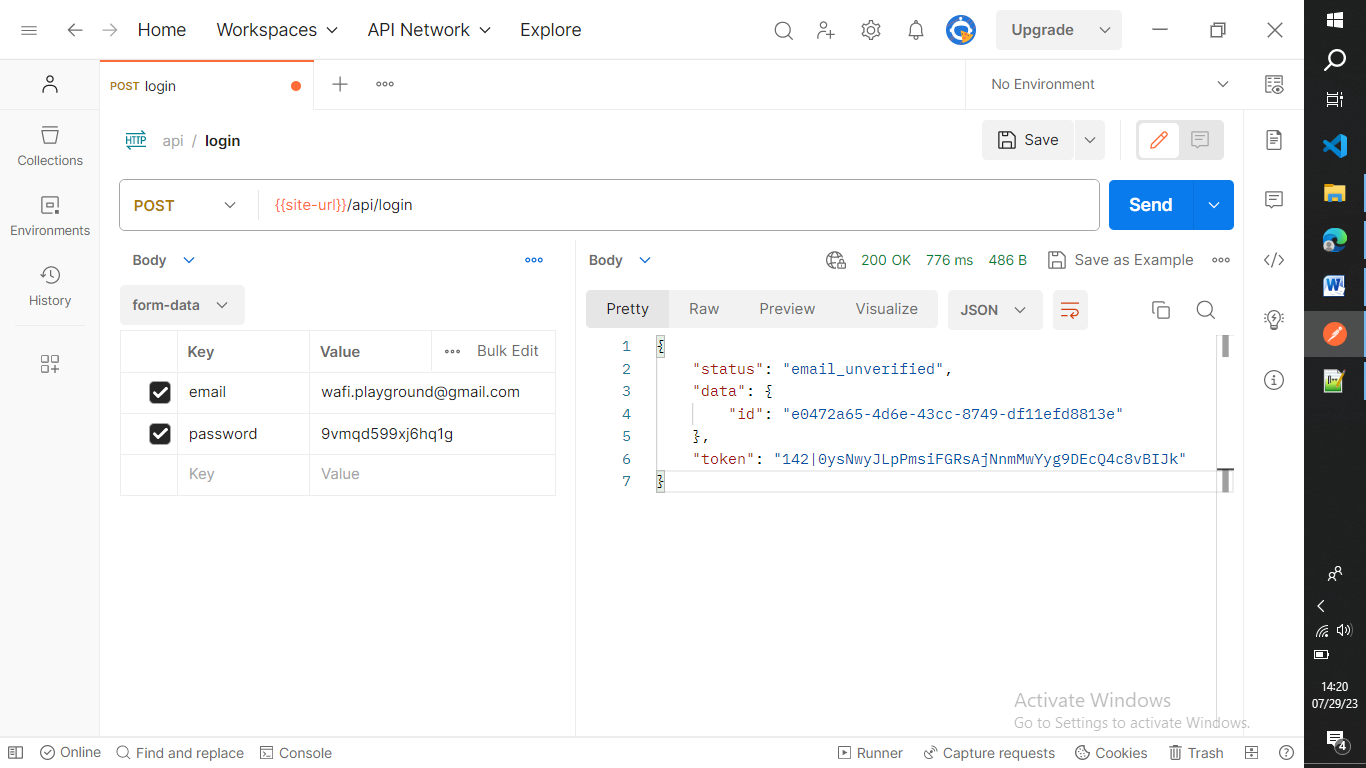
1. API *Login*

API *login* akan menerima alamat email dan *password* pengguna untuk melakukan autentikasi. Hasil dari perancangan dari API *login* dapat dilihat pada gambar 4.1 dibawah.



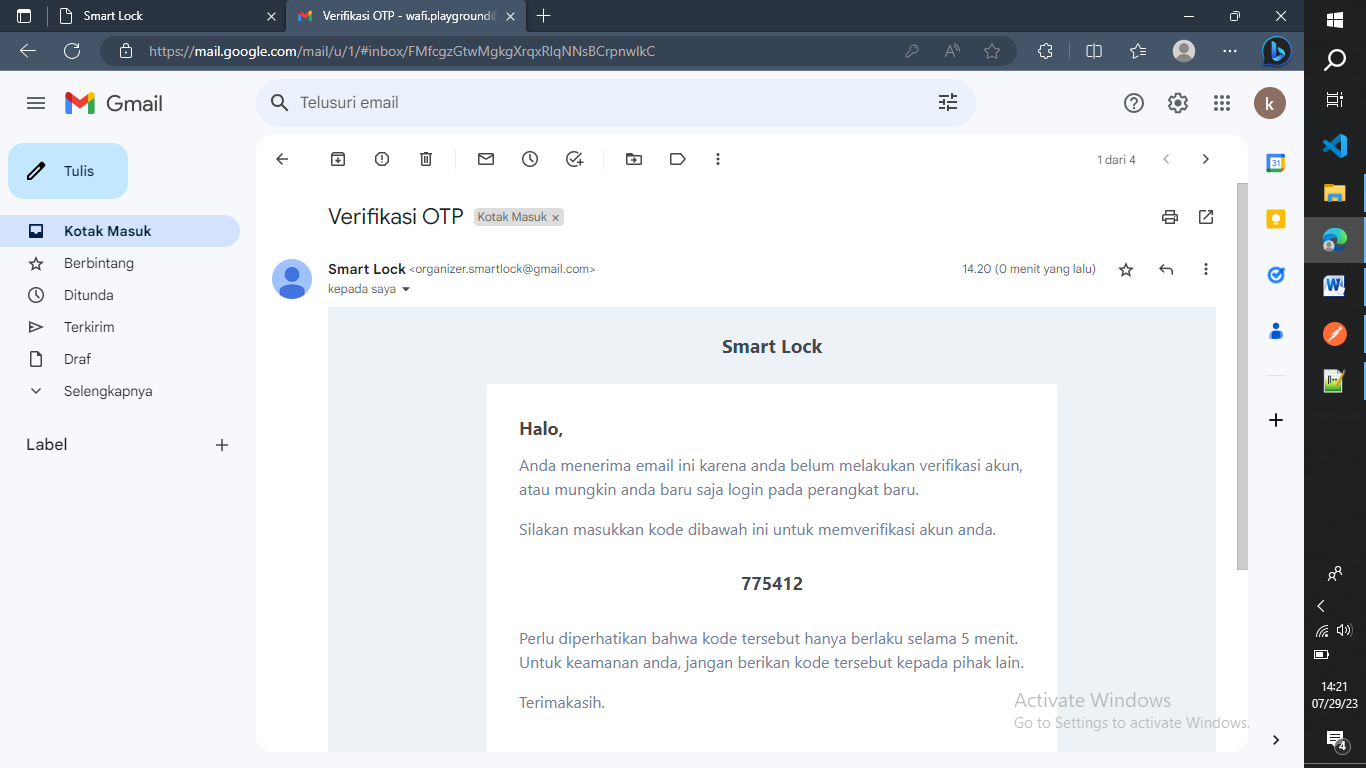
**Gambar 4.1** Hasil API *Login*

Gambar 4.1 menampilkan hasil *request* *login* yang dikirimkan melalui *enpoint* “/api/login”. API *login* akan memberikan respon status *login*, data pengguna yang melakukan *login* serta token yang akan digunakan untuk mengakses API lainnya yang membutuhkan autentikasi token.



**Gambar 4.2** Hasil API *Login* Email Belum Terverifikasi

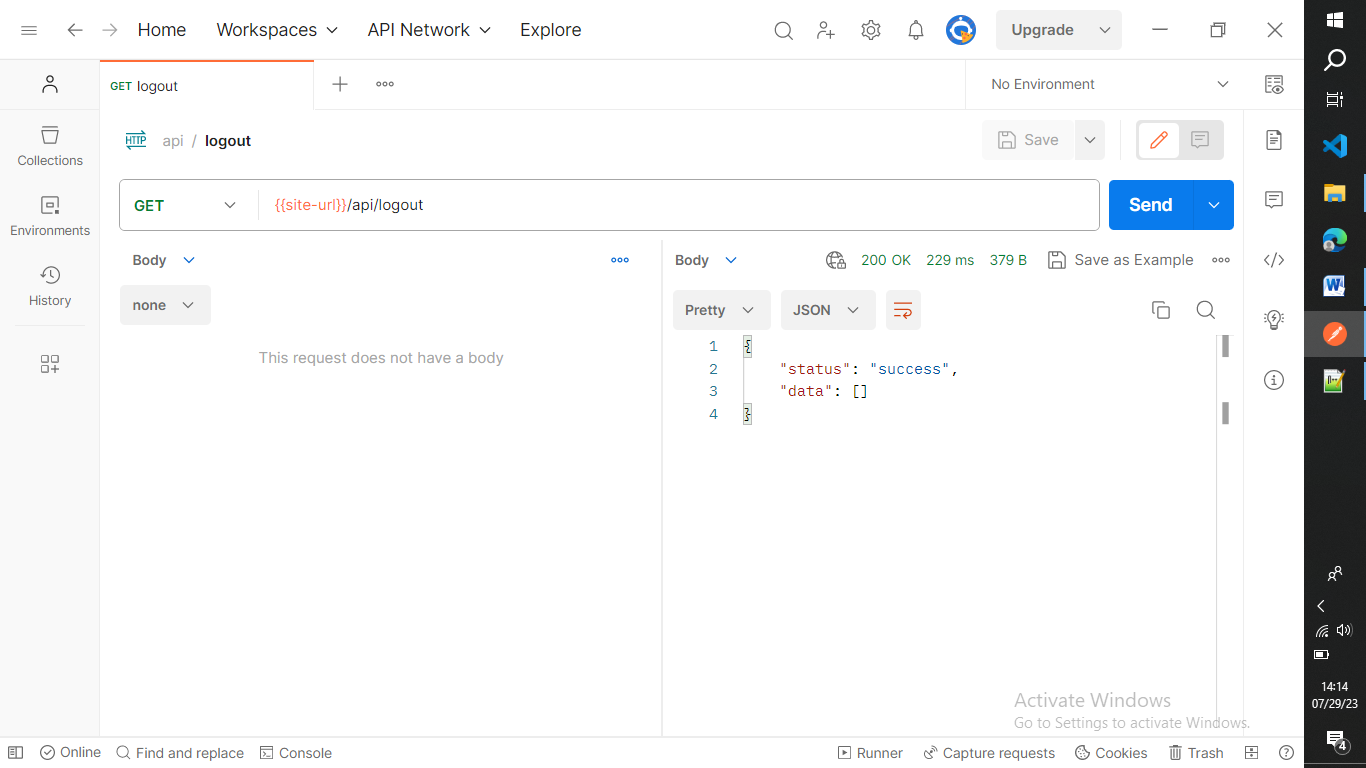
Pada API *login* juga dilakukan pengecekan email pengguna, seperti yang terlihat pada gambar 4.2, pengguna yang belum melakukan verifikasi email akan menerima respon status email belum terverifikasi dan pengguna akan menerima kode OTP yang dikirimkan ke email pengguna seperti yang terlihat pada gambar 4.3. Pada kondisi ini API juga memberikan token yang akan digunakan oleh pengguna untuk melakukan verifikasi email karena API untuk melakukan verifikasi email membutuhkan akses *login*.



**Gambar 4.3** Email Kode OTP

1. API *Logout*

API *logout* digunakan untuk mengakhiri sesi *login* pengguna dengan menghapus semua token yang dimiliki oleh pengguna. API *logout* akan menerima permintaan *logout* pengguna dengan melampirkan token yang dimiliki. Hasil dari API *logout* dapat dilihat pada gambar x.x dibawah.

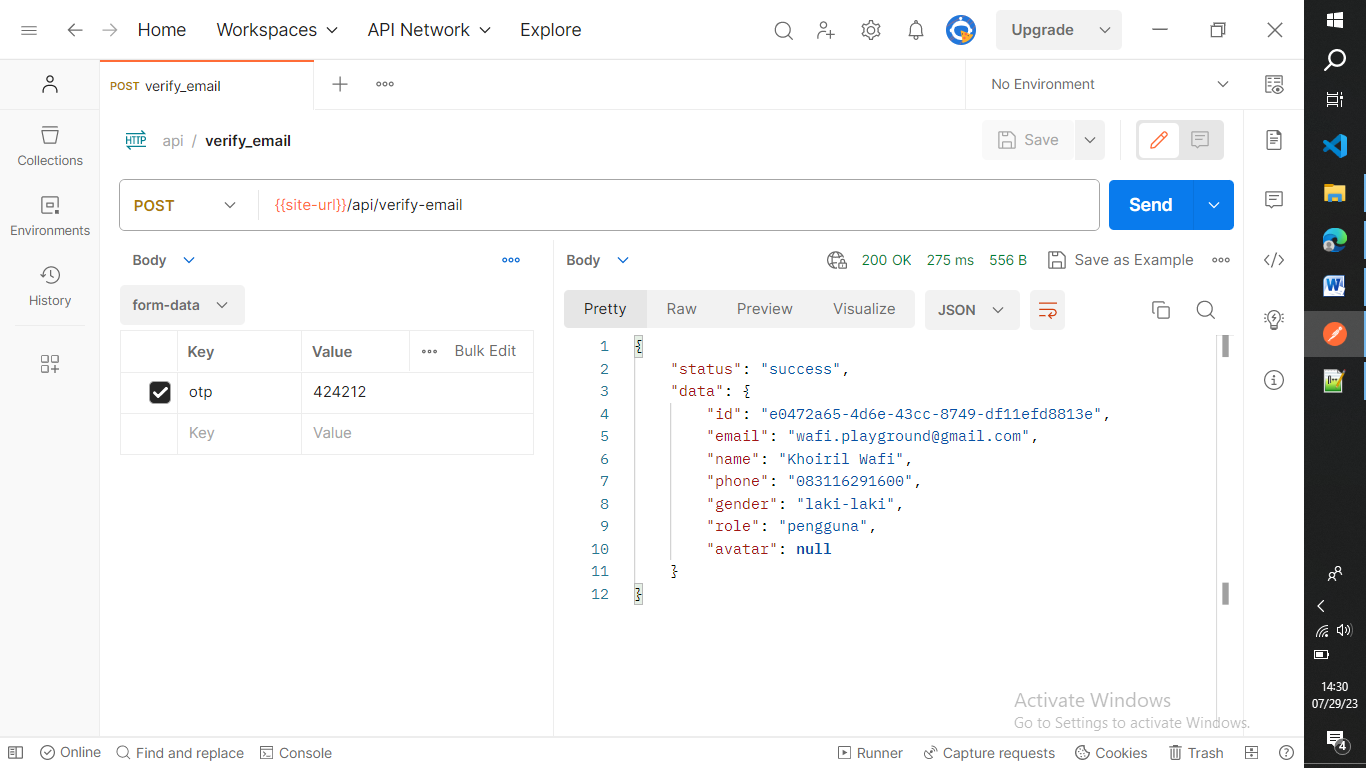


**Gambar 4.4** Hasil API *Logout*

Gambar 4.4 menampilkan hasil *request* dari API *logout* yang dikirimkan melalui *endpoint* “/api/logout”. API logout akan memberikan respon status *logout* yang dilakukan oleh pengguna dengan melampirkan *token* yang dimiliki oleh pengguna yang bersangkutan pada bagian *header* *request*.

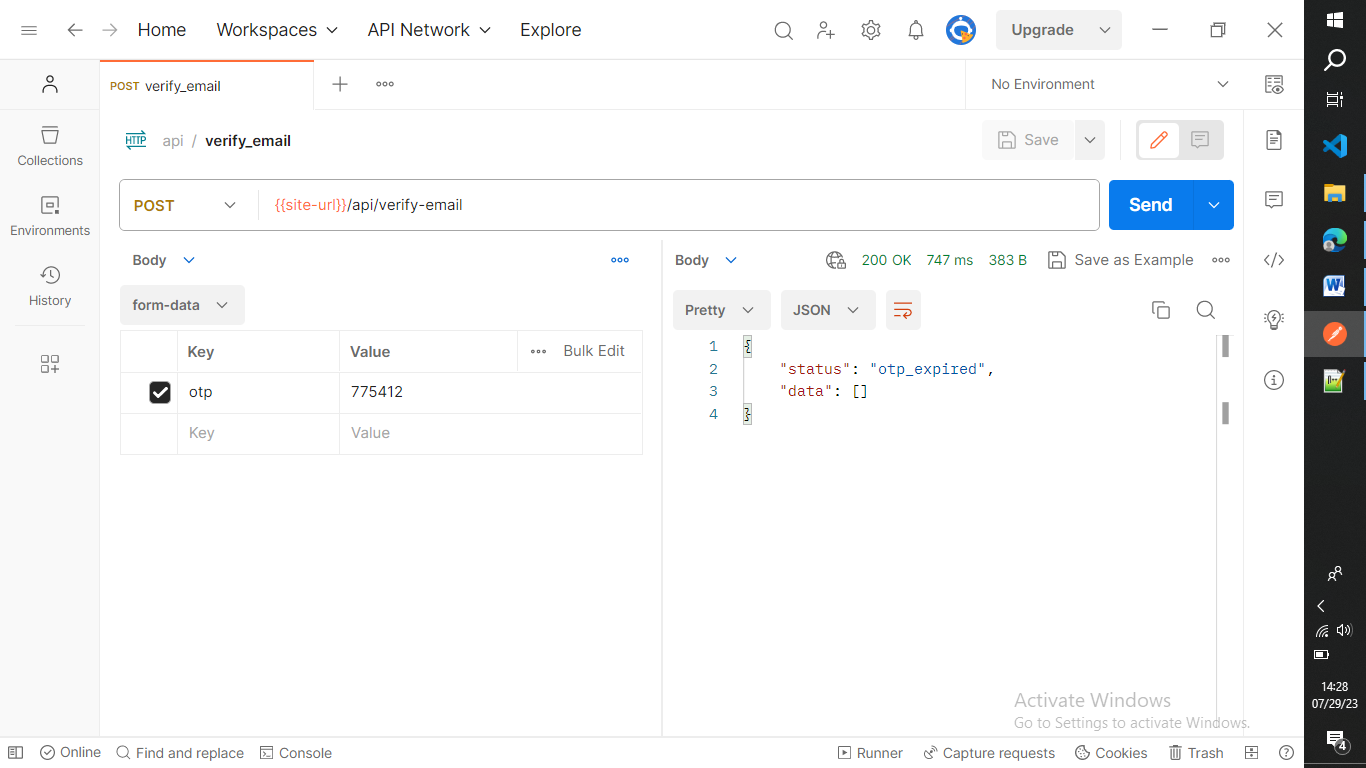
1. API Verifikasi Email

Pada saat pengguna melakukan login dan mendapatkan respon email belum terverifikasi maka pengguna akan mendapatkan sebuah kode OTP yang dikirimkan ke email pengguna. API verifikasi email akan menerima kode OTP yang telah diterima oleh pengguna untuk melakukan verifikasi email. Dengan menggunakan langkah verifikasi ini maka hanya pengguna atau operator dengan email yang valid dan aktif yang dapat mengakses API sehingga akan meningkatkan keamanan dari API tersebut. Hasil dari API verifikasi email dapat dilihat pada gambar 4.5 dibawah.



**Gambar 4.5** Hasil API Verifikasi Email

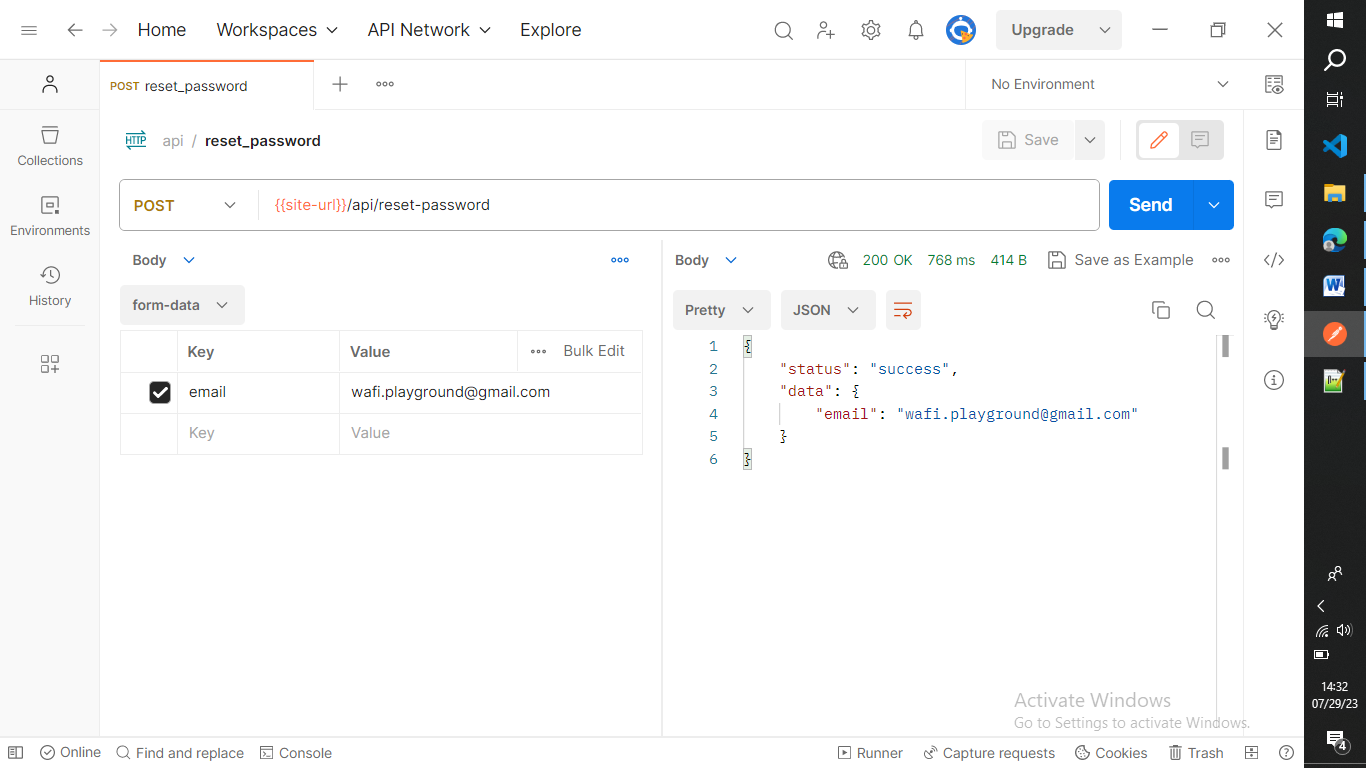
Gambar 4.5 menampilkan hasil dari API verifikasi email yang dikirimkan melalui *endpoint* “api/verify-email”. Pengguna akan mengirimkan kode OTP melalui *body* serta melampirkan token *login* pada *header* *request*, jika verifikasi berhasil maka API akan memberikan respon status berhasil dan mengirimkan data pengguna seperti pada saat *login*. API verifikasi email juga akan memeriksa waktu kadaluarsa dari kode OTP, jika kode OTP sudah kadaluarsa maka verifikasi akan gagal seperti yang terlihat pada gambar x.x dibawah.



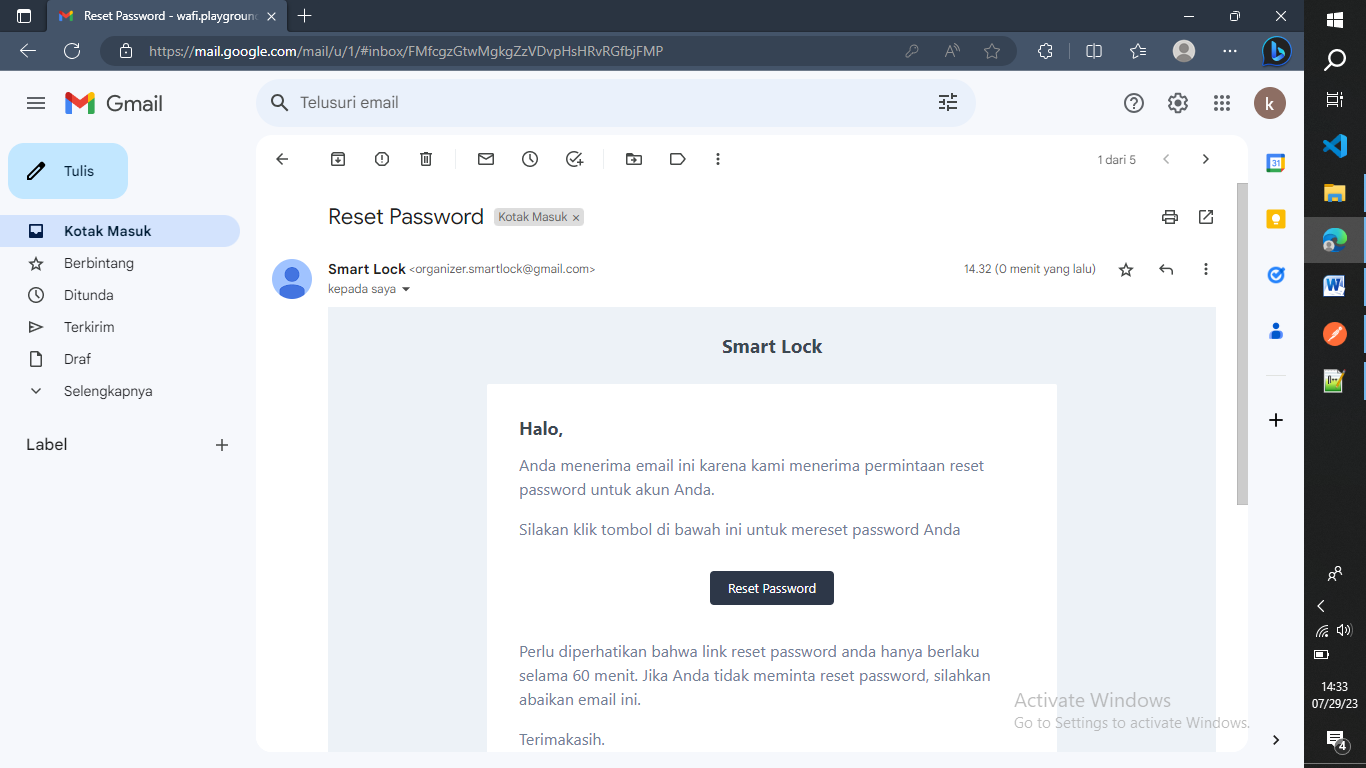
**Gambar 4.6** Kode OTP Kadaluarsa

1. API *Reset* *Password*

API *reset* *password* akan menerima email dari pengguna dan mengirimkan url yang dapat digunakan untuk membuat *password* baru ke email pengguna. Hasil dari API *reset* *password* dapat dilihat pada gambar 4.7 dibawah.



**Gambar 4.7** Hasil API *Reset* *Password*

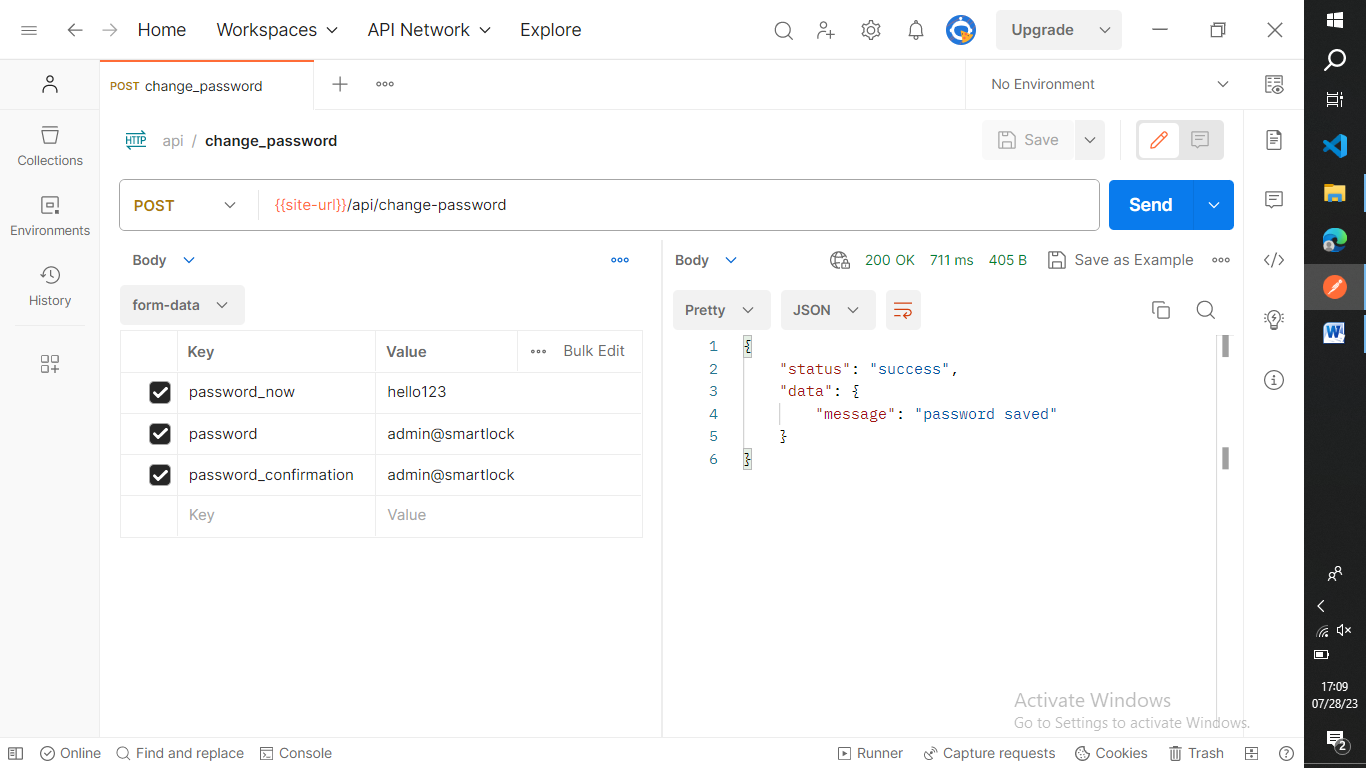


**Gambar 4.8** Email Reset Password

Gambar 4.7 merupakan hasil API *reset* *password* yang dikirimkan melalui *endpoint* “/api/reset-password”. Pengguna akan mengirimkan alamat email pada *body* *request* dan API akan memberikan respon status beserta data alamat email. Pada proses ini API juga akan mengirimkan url *reset* *password* ke email pengguna seperti yang terlihat pada gambar 4.8, pengguna dapat menggunakan url tersebut untuk membuat *password* baru melalui halaman *website*.

1. API Ganti *Password*

API ganti password akan menerima *password* lama dan *password* baru pengguna untuk memperbarui *password* pengguna. API ganti *password* juga akan memeriksa token *login* yang dikirimkan untuk memastikan bahwa *pengguna* sudah melakukan proses autentikasi *login*. Hasil dari API ganti *password* dapat dilihat pada gambar x.x dibawah.

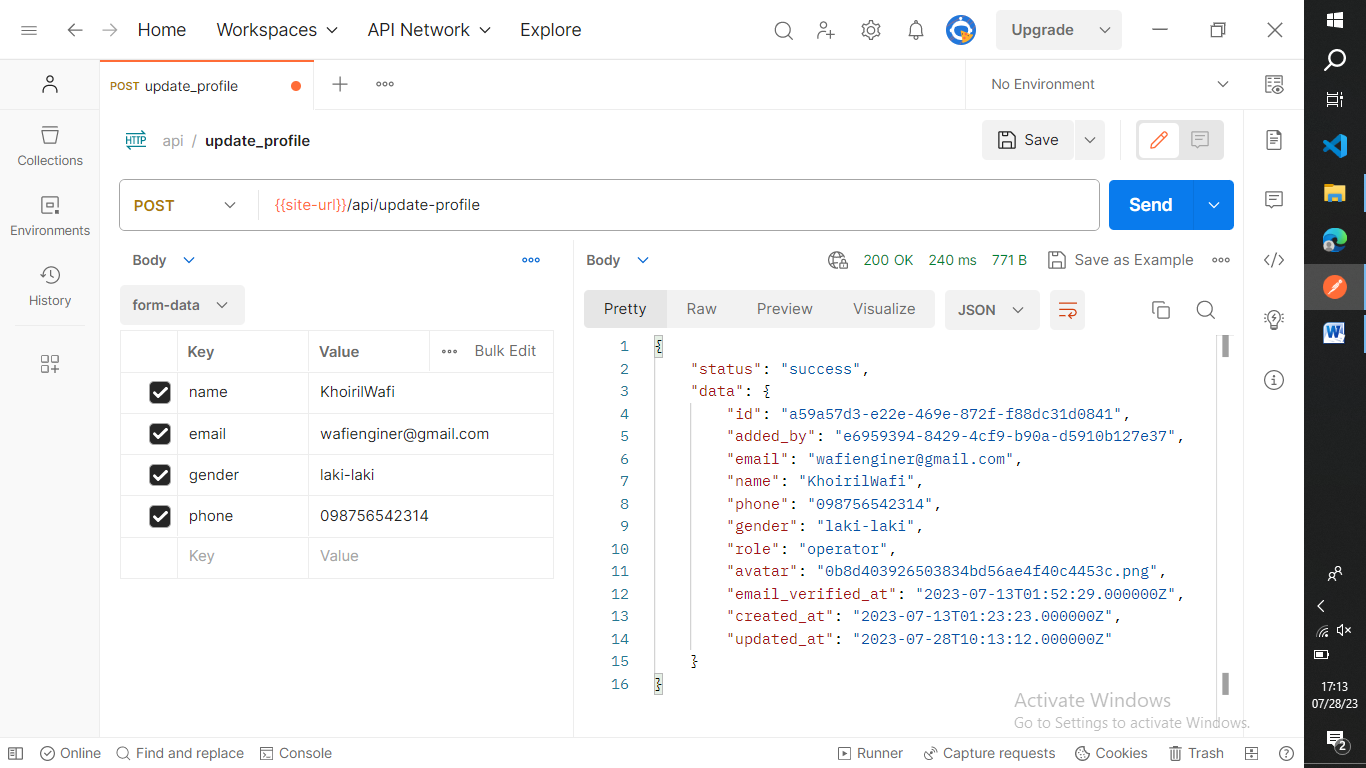


**Gambar 4.9** Hasil API Ganti *Password*

Gambar 4.9 merupakan hasil dari API ganti *password* yang dikirimkan melalui *enpoint* “/api/change-password”. Pengguna akan mengirimkan *password* lama dan *password* baru serta konfirmasi *password* baru pada *body* *request*, penggunakan juga memerlukan token *login* untuk mengakses *enpoint* ini. API ganti *password* akan memberikan respon status dan pesan apakah *password* baru berhasil tersimpan atau tidak. Dengan adanya API ganti password maka pengguna dan operator dapat mengganti password secara berkala sesuai dengan kehendak masing-masing untuk meningkatkan kemanan akun mereka.

1. API Ganti Profil

API ganti profil akan menerima data pengguna yang akan diganti seperti nama, email, jenis kelamin dan nomor hp, API ganti profil juga memerlukan token *login* untuk memastikan pengguna sudah melakukan autentikasi. Hasil dari API ganti profil dapat dilihat pada gambar 4.10 dibawah.

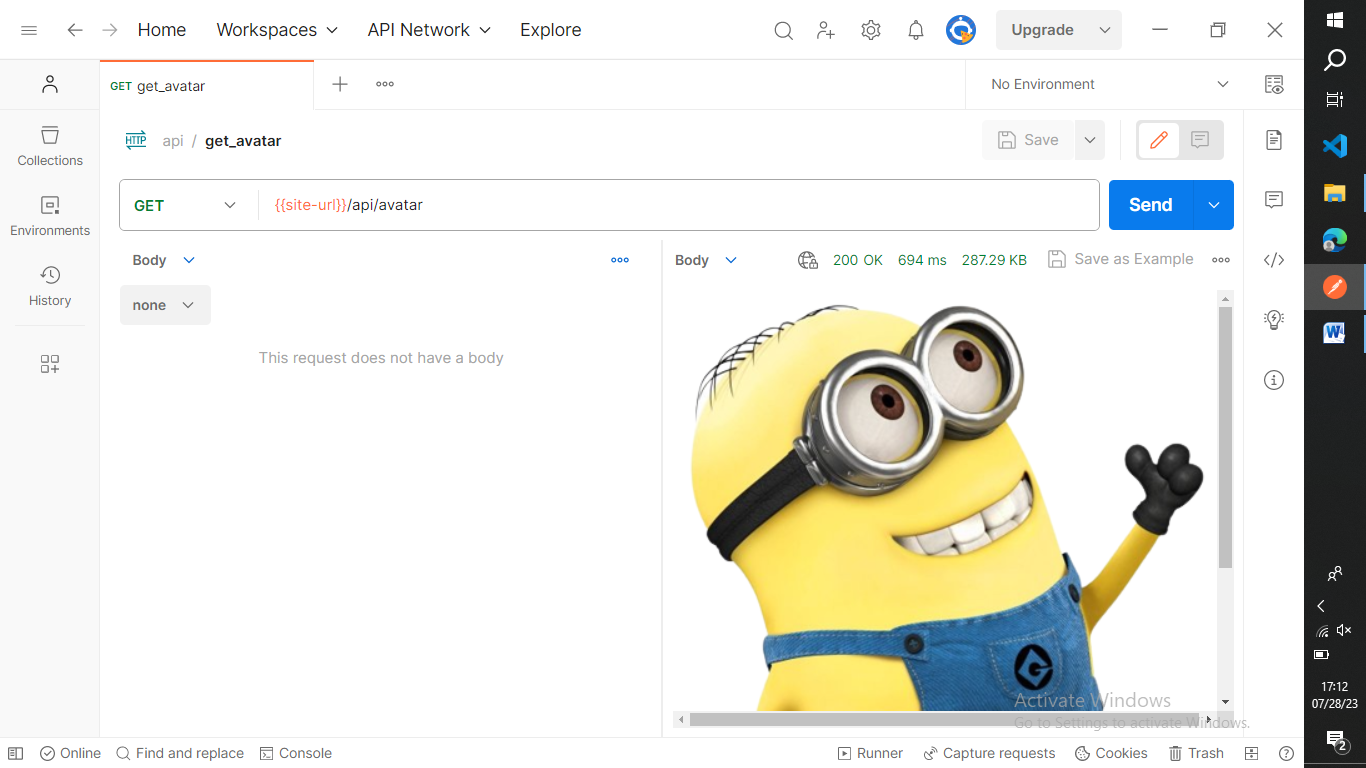


**Gambar 4.10** API Ganti Profil

Gambar 4.10 merupakan hasil dari API ganti profil yang dikirimkan melalui *endpoint* “/api/update-profile”. Pengguna akan mengirimkan data yang akan diganti seperti nama, email, jenis kelamin dan nomor hp pada *body* *request*, pengguna juga menyertakan token *login* pada *header* *request*. API akan memberikan respon berupa status dan data pengguna yang sudah dirubah seperti yang terlihat pada gambar 4.10 diatas.

1. API Lihat Avatar

Setiap gambar avatar pengguna tidak tersedia untuk publik, pengguna akan melakukan *request* untuk mendapatkan gambar avatar dengan melampirkan token *login* ke API lihat avatar. API lihat avatar aakan memberikan respon berupa gambar avatar dari setiap pengguna atau operator untuk ditampilkan pada aplikasi masing-masing mobile. Hasil dari API lihat avatar dapat dilihat pada gambar 4.11 dibawah.

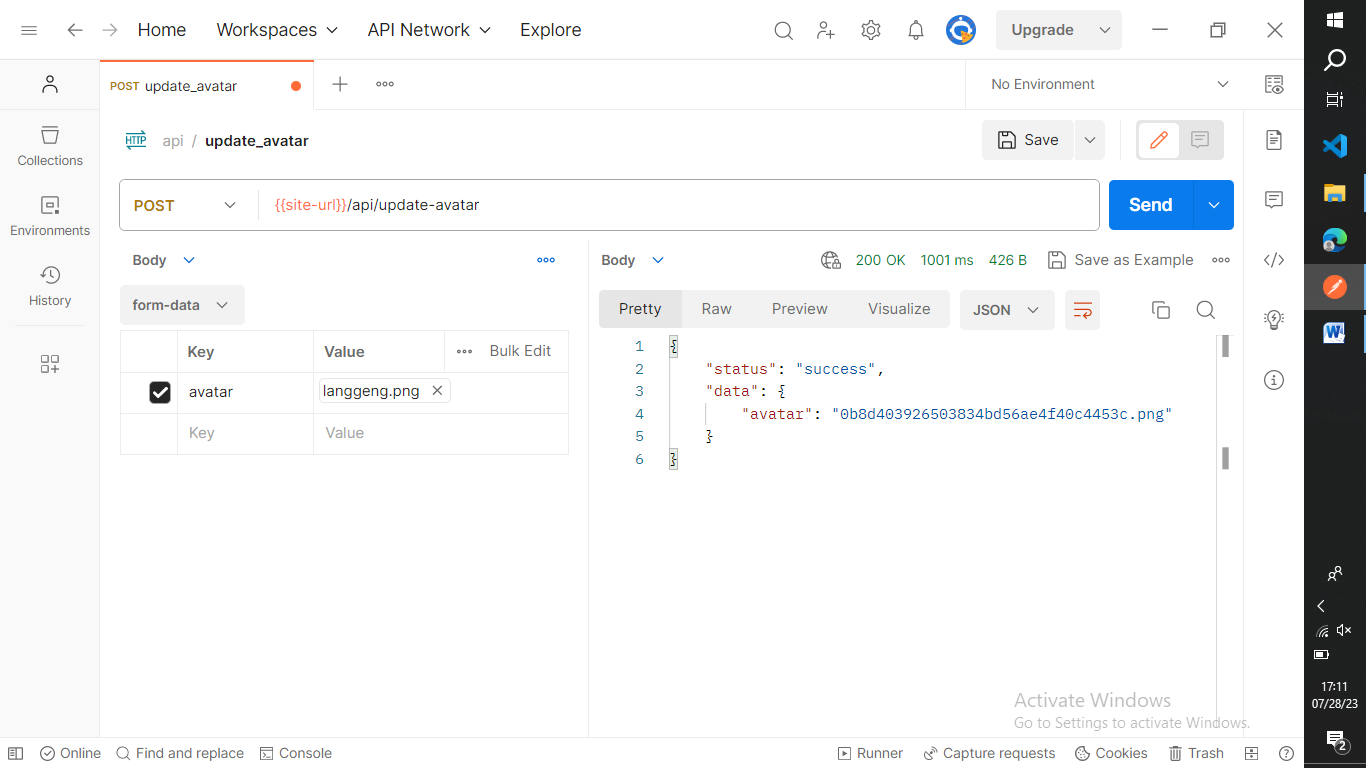


**Gambar 4.11** Hasil API Lihat Avatar

Gambar 4.11 merupakan hasil dari API livat avatar yang dikirimkan melalui *endpoint* “/api/avatar”, pengguna akan mengirimkan permintaan disertai dengan token *login* dan API akan memberikan respon berupa gambar avatar masing-masing pengguna dalam format file gambar. Jika pengguna belum memiliki avatar maka API akan mengembalikan respon berupa gambar *default* untuk avatar yang terlas disediakan oleh server.

1. API Ganti Avatar

Pengguna dan operator dapat memperbarui gambar avatar melalui aplikasi mobile. Untuk mengganti avatar melalui aplikasi mobile pengguna akan mengirimkan gambar avatar melalui API ganti avatar, pengguna juga memerlukan akses *login* untuk mengakses API ganti avatar. Pada API Ganti avatar pengguna hanya diijinkan untuk mengunggah file gambar dengan dengan ukuran maksimal 1 MB hal ini bertujuan untuk mencegah pengguna mengunggah file lain seperti dokumen atau pdf dan juga untuk membatasi ukuran file proses pengiriman gambar dapat berjalan dengan lancar. Hasil dari API ganti avatar dapat dilihat pada gambar 4.12 dibawah ini.

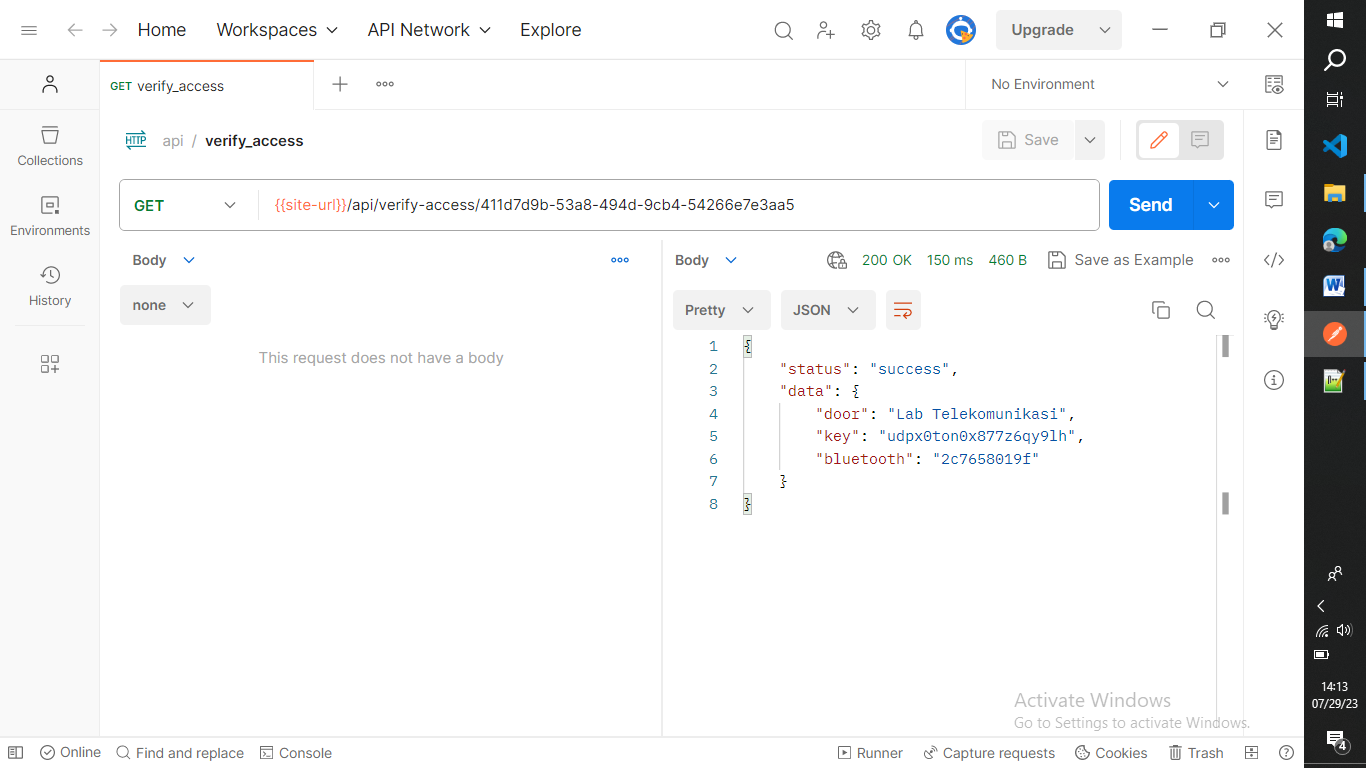


**Gambar 4.12** Hasil API Ganti Avatar

Gambar 4.12 merupakan hasil dari API ganti avatar yang dikirimkan melalui *endpoint* “/api/update-avatar”. Pengguna akan mengirimkan file gambar avatar ke API disertai dengan token *login* yang dimiliki, API akan memberikan respon status dan nama file avatar yang berhasil tersimpan didalam server.

1. API Verifikasi Akses

Pada setiap pintu akan terpasang kode QR yang menjadi identitas dari pintu tersebut. Pengguna dan operator dapat menggunakan kode QR tersebut untuk membuka pintu. Ketika pengguna atau operator melakukan pemindaian kode QR untuk membuka pintu maka perangkat mobile akan mengirimkan request ke API verifikasi akses untuk mendapatkan kunci dari pintu tersebut. Kode QR yang terpasang nantinya akan menyimpan informasi identitas dari pintu yang digunakan untuk membedakan pintu satu dengan yang lainnya. Dengan menggunakan metode ini maka akses pengguna dapat diatur secara langsung karena setiap ingin membuka kunci pintu maka pengguna harus meminta kunci pintu melalui API verifikasi akses sehingga akan meningkatkan keamanan sistem penguncian pintu gedung ini. Hasil dari API verifikasi akses dapat dilihat pada gambar 4.13 dibawah.

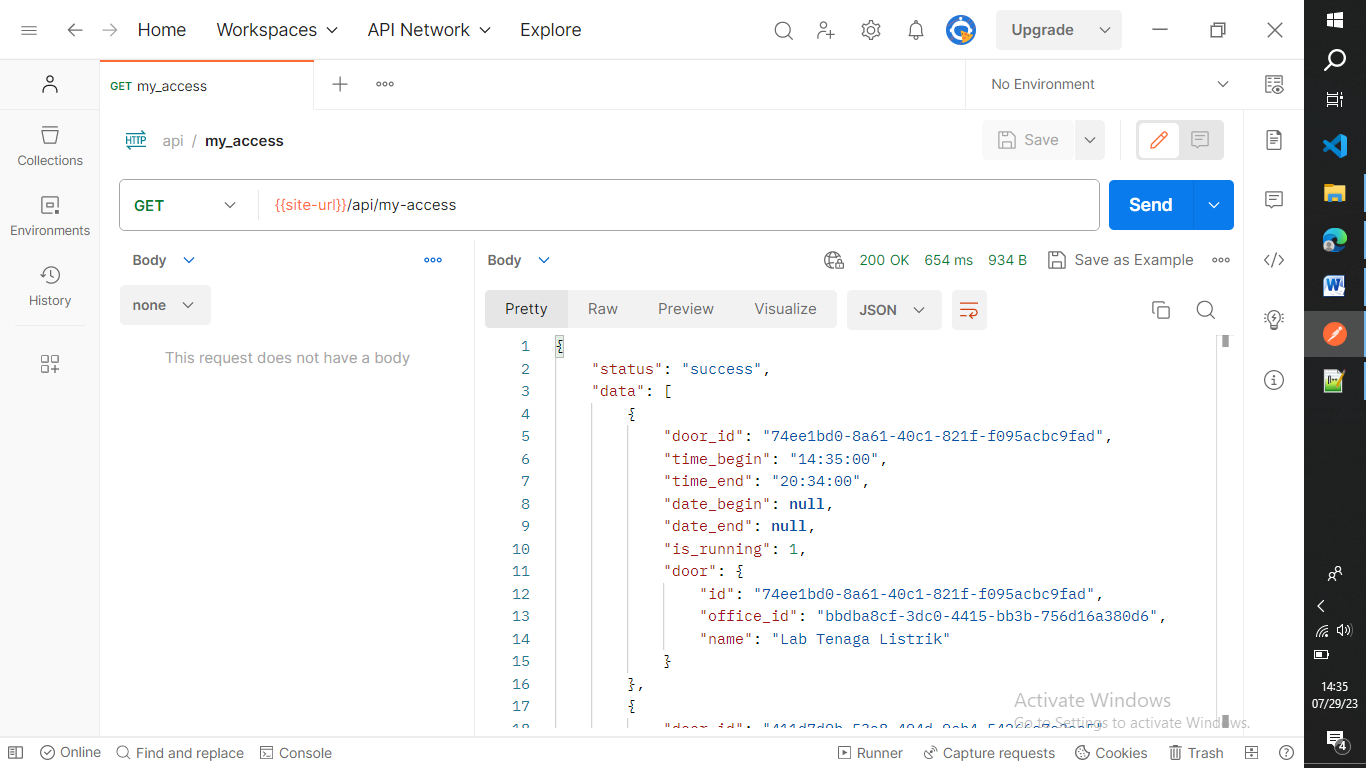


**Gambar 4.13** Hasil API Verifikasi Akses

Gambar 4.13 merupakan hasil dari API verifikasi akses yang dikirimkan melalui *endpoint* “/api/verify-access/{id}” dimana id merupakan identitas dari pintu. Pengguna akan mengirimkan token *login* yang dimiliki dan API akan memberikan respon terkait akses yang dimiliki pengguna tersebut, jika diizinkan maka API akan memberikan respon status diijinkan dan API akan memberikan respon data yang digunakan untuk membuka pintu seperti nama pintu, kunci pintu dan nama bluetooth untuk berkomunikasi dengan perangkat kunci pintu. Namun jika akses pengguna tidak ditemukan atau akses pengguna sedang diblokir maka Api verifikasi akses akan memberikan respon gagal atau tidak diijinkan dan tidak akan meberikan data untuk membuka pintu.

1. API Daftar Akses

API daftar akses akan memberikan informasi pintu mana saja yang dapat diakses oleh pengguna. Setiap penggguna dengan pengguna lain mungkin memiliki akses yang berbeda, pemberian akses pintu kepada pengguna tertentu akan diatur oleh seorang operator. Seorang operator dapat memberikan akses atau menghapus akses pengguna pada pintu tertentu, operator juga dapat menghentikan sementara akses pengguna untuk keperluan tertentu. Hasil dari API daftar akses dapat dilihat pada gambar 4.14 dibawah.

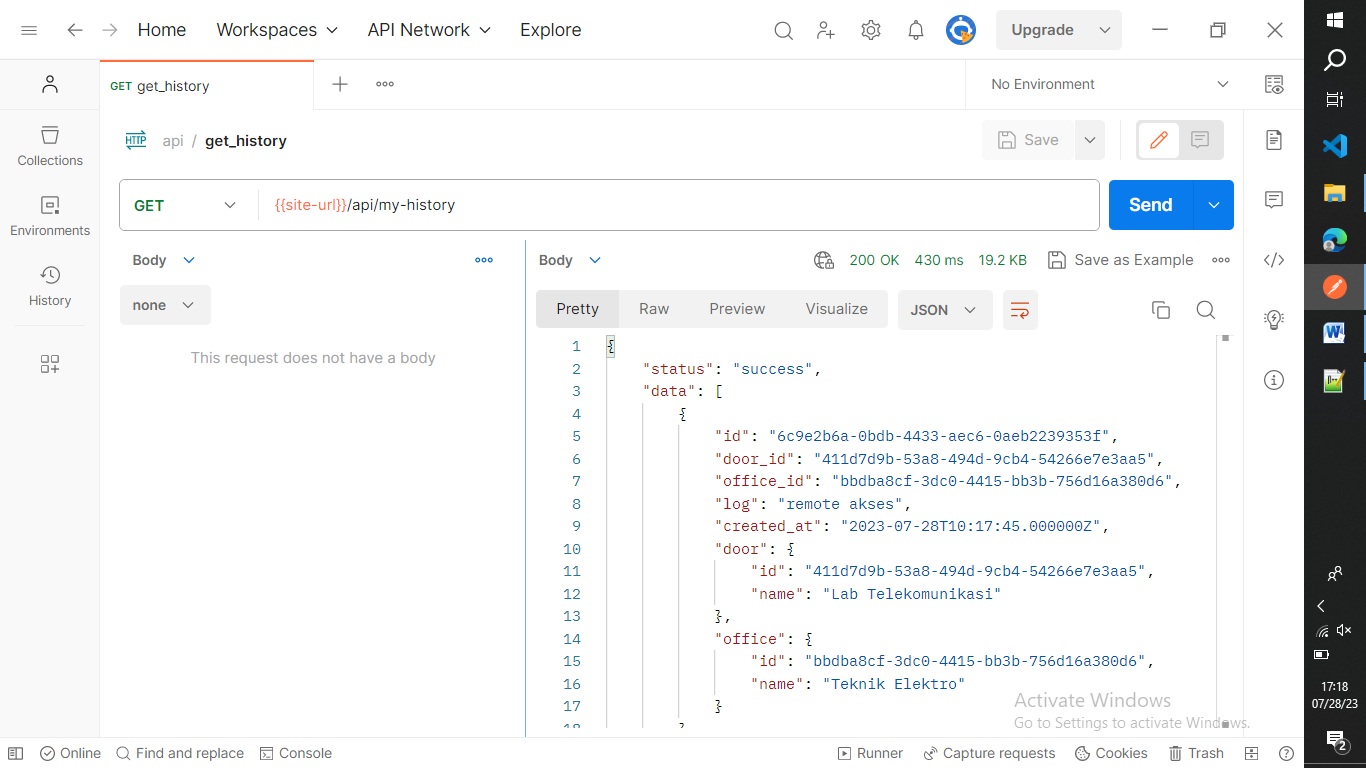


**Gambar 4.14** Hasil API Daftar Akses

Gambar 4.14 merupakan hasil dari API daftar akses yang dikirimkan oleh pengguna melalui *enpoint* “/api/my-access”. API daftar akses akan memberikan respon berupa daftar informasi pintu yang dapat diakses oleh penggua tersebut yaitu berupa nama pintu, nama gedung, rentang waktu akses, rentang tanggal akses dan status akses. Dengan menggunakan data akses yang dimiliki pengguna dapat mengetahui pintu mana saja yang dapat diakses oleh pengguna tersebut dan juga mengetahui jika akses pengguna masih aktif atau sedang ditangguhkan. Dengan adanya informasi ini pengguna dapat mengelola akses merekan dengan efisien sesuai dengan kebutuhan masing-masing pengguna.

1. API Riwayat Akses

API riwayat akses akan memberikan informasi kepada pengguna terkait dengan catatan aktifitas pengguna didalam sistem penguncian gedung ini dengan menampilkan data riwayat aktifitas pada aplikasi pengguna. API ini dapat diakses oleh pengguna maupun operator untuk melihat aktifitas mereka didalam sistem penguncian pintu gedung ini. Pencatatan aktifitas dilakukan berdasarkan kegiatan pengguna dan operator seperti memindai kode QR, membuka kunci pintu melakui akses *remote*, membuat jadwal pintu otomatis dan lain sebagainya. Hasil dari API dapat dilihat pada gambar 4.15 dibawah.

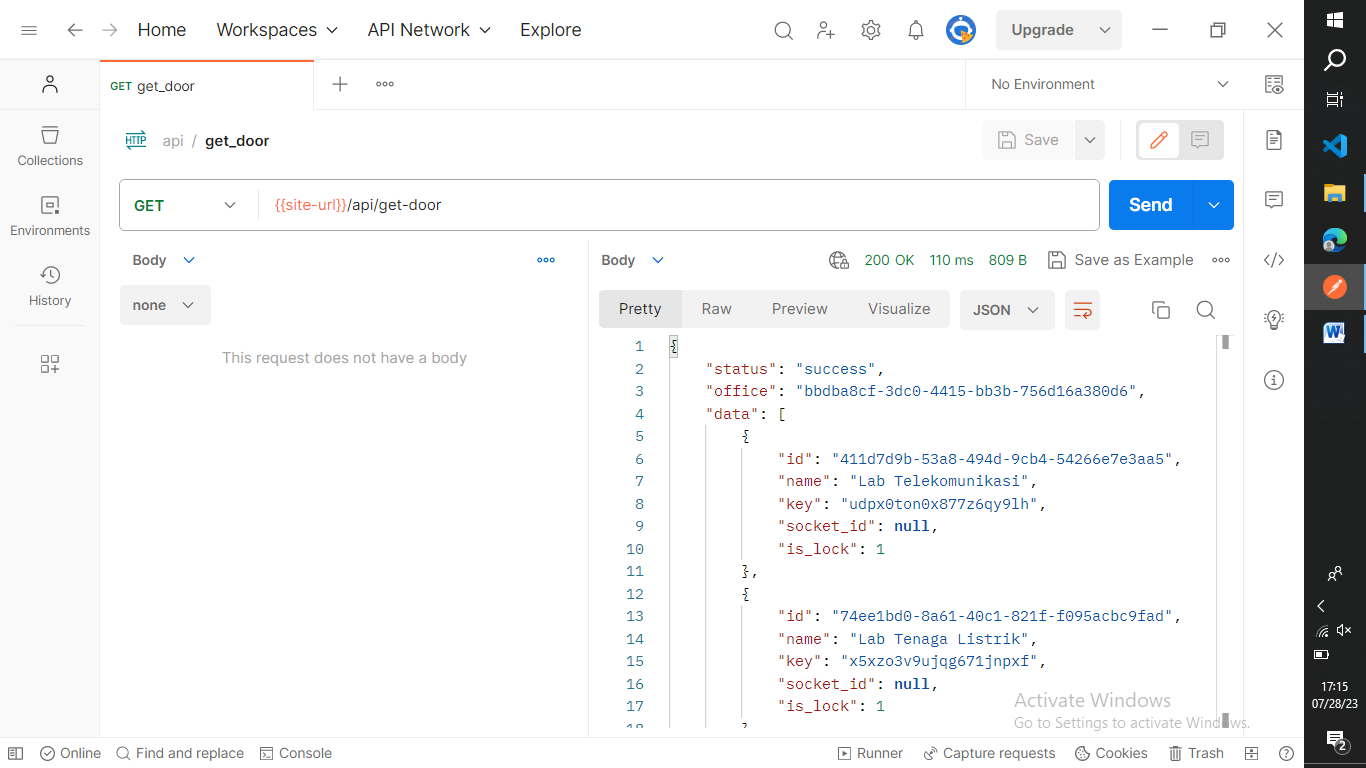


**Gambar 4.15** API Riwayat Akses

Gambar 4.15 merupakan hasil dari API riwayat akses. API riwayat akses akan memberikan data pencatatan aktifitas pengguna seperti melakukan pindai kode QR dan membuka pintu disertai dengan waktu dan tanggal aktifitas tersebut. Setiap aktifitas penguncian akan tersimpan didalam *database* sebagai riwayat aktifitas, seorang operator dapat melihat semua aktifitas yang terjadi didalam sistem penguncian tetapi pengguna hanya dapat melihat aktifitas dirinya sendiri. Pada gambar 4.15 terlihat bahwa data riwayat aktifitas berupa nama pintu, waktu, dan jenis aktifitas yang dilakukan. Data riwayat akses hanya tersedia dalam kurun waktu 1 bulan dan setiap awal bulan maka data riwayat akses akan di pindahkan kedalam penyimpanan server (*backup*) dan tabel riwayat akses dalam *database* akan dibersihkan.

1. API Daftar Pintu

API daftar pintu digunakan untuk mendapatkan daftar pintu yang ada pada sebuah gedung, data tersebut kemudian ditampilkan pada aplikasi mobile yang akan memberikan informasi terkait kondisi pintu apakah terkunci, *offline* atau kondisi lainnya. API ini hanya dapat diakses oleh operator gedung karena hak pengelolaan pintu gedung berada pada tangan operator. Hasil dari API daftar pintu dapat dilihat pada gambar 4.16 dibawah.

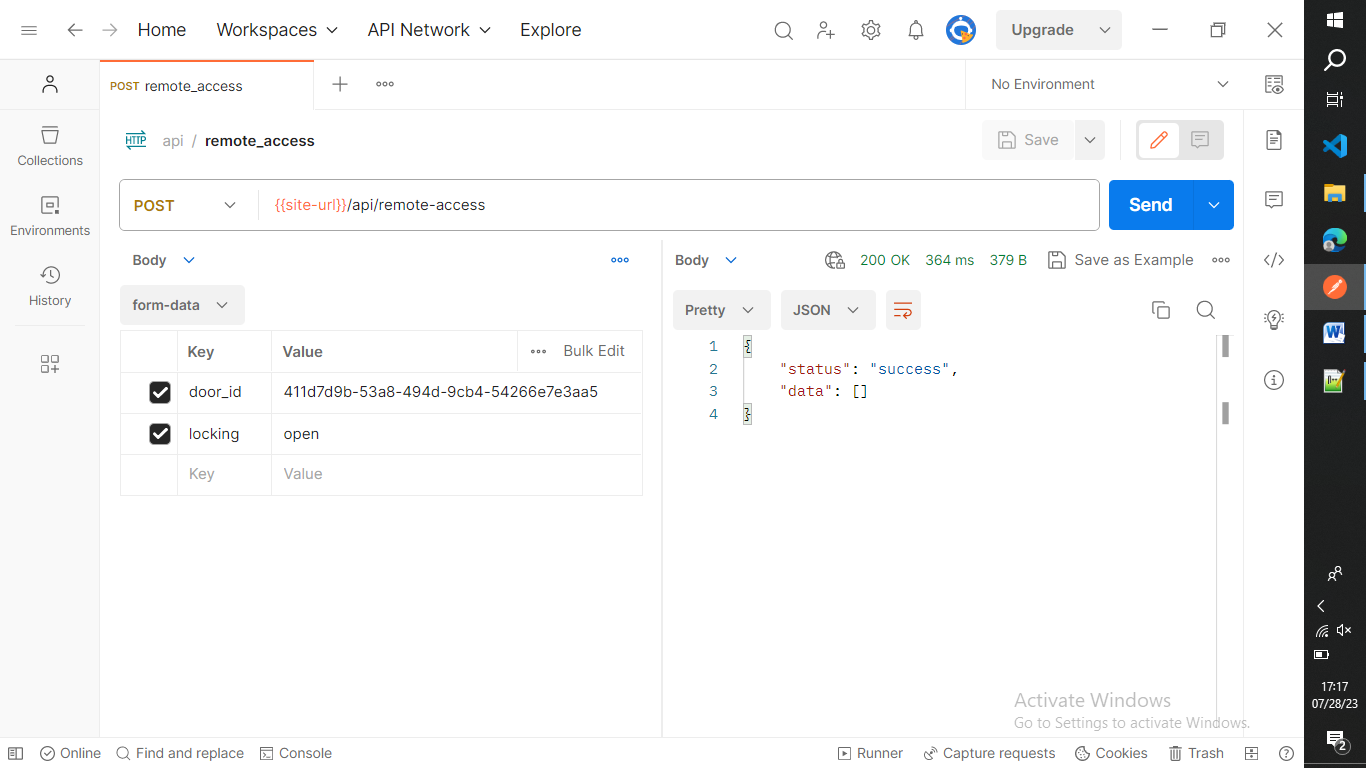


**Gambar 4.16** Hasil API Daftar Pintu

Gambar 4.16 merupakan hasil dari API daftar gedung yang dikirimkan melalui *endpoint* “/api/get-door”. Pengguna akan mengirimkan token login yang dimiliki, jika pengguna merupakan seorang operator maka API akan memberikan respon daftar pintu yang ada pada gedung tersebut beserta dengan informasi tambahan seperti nama pintu, kondisi penguncian, koneksi, dan kunci pintu. Namun jika pengguna bukan seorang operator maka API akan memberikan respon bahwa pengguna tersebut tidah diijinkan untuk mengakses API ini.

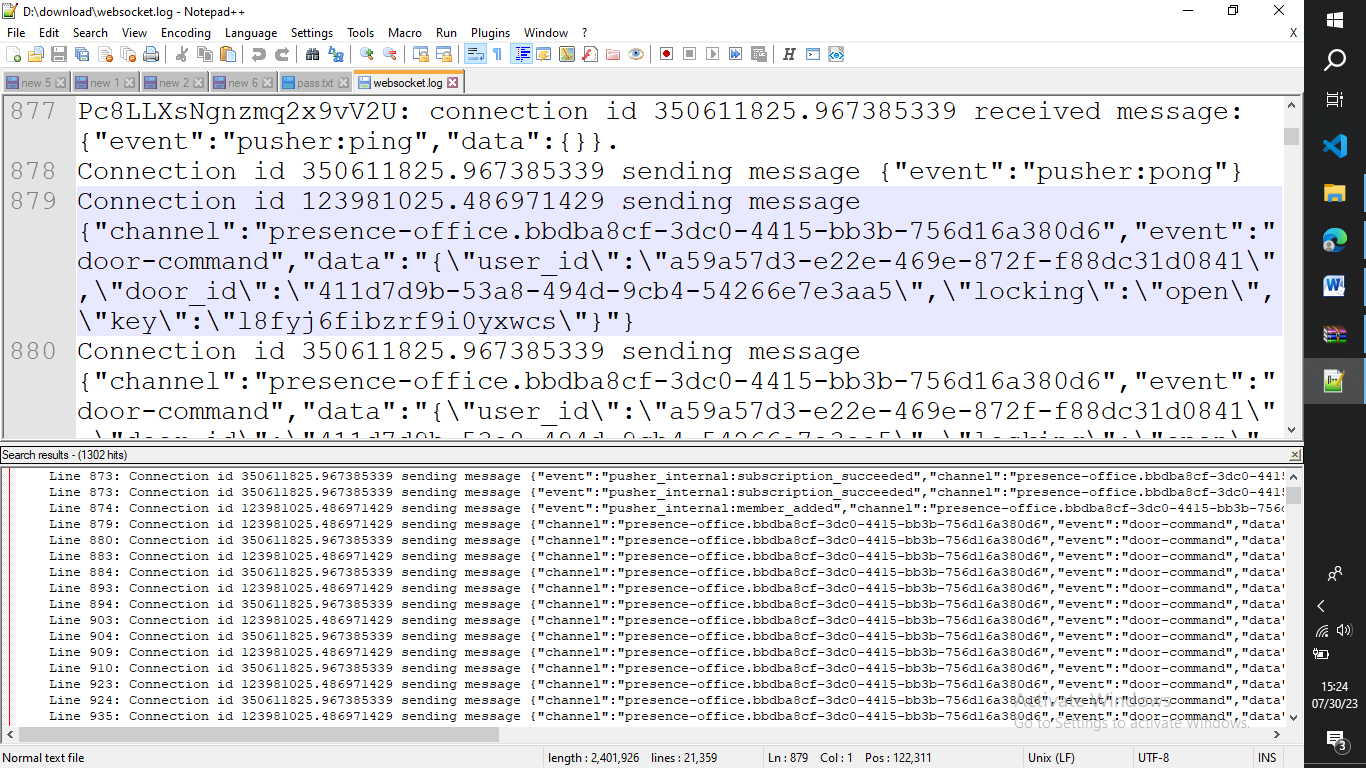
1. API *Remote* Pintu

Pada aplikasi yang digunakan oleh operator terdapat tombol yang dapat digunakan untuk membuka dan mengunci pintu sehingga memungkinkan operator dapat melakukan pengelolaan pintu secara jarak jauh melalui koneksi internet. Untuk membuka atau mengunci pintu sacara jarak jauh melalui koneksi internet, maka seorang operator akan melakukan *request* dengan mengirimkan identitas pintu beserta token *login* yang dimiliki, jika sesuai maka server akan mengirimkan perintah ke perangkat kunci pintu melalui koneksi websocket yang sudah terhubung. Namun jika terjadi permasalahan maka API akan memberikan respon error. Hasil dari API remote pintu dapat dilihat pada gambar 4.17 dibawah.



**Gambar 4.17** Hasil API *Remote* Pintu

Gambar 4.17 merupakan hasil dari API *remote* pintu yang dikirimkan melalui *endpoint* “/api/remote-access”. Untuk membuka atau mengunci pintu sacara jarak jauh melalui koneksi internet operator akan mengirimkan identitas pintu yang dituju serta aktifitas yang dilakukan seperti *lock* atau *open* yang menjelaskan aktifitas membuka dan mengunci pintu, kemudian server akan memeriksa permintaan dan mengirimkan perintah ke perangkat kunci pintu melalui koneksi websocket seperti yang terlihat pada gambar 4.18 dibawah.

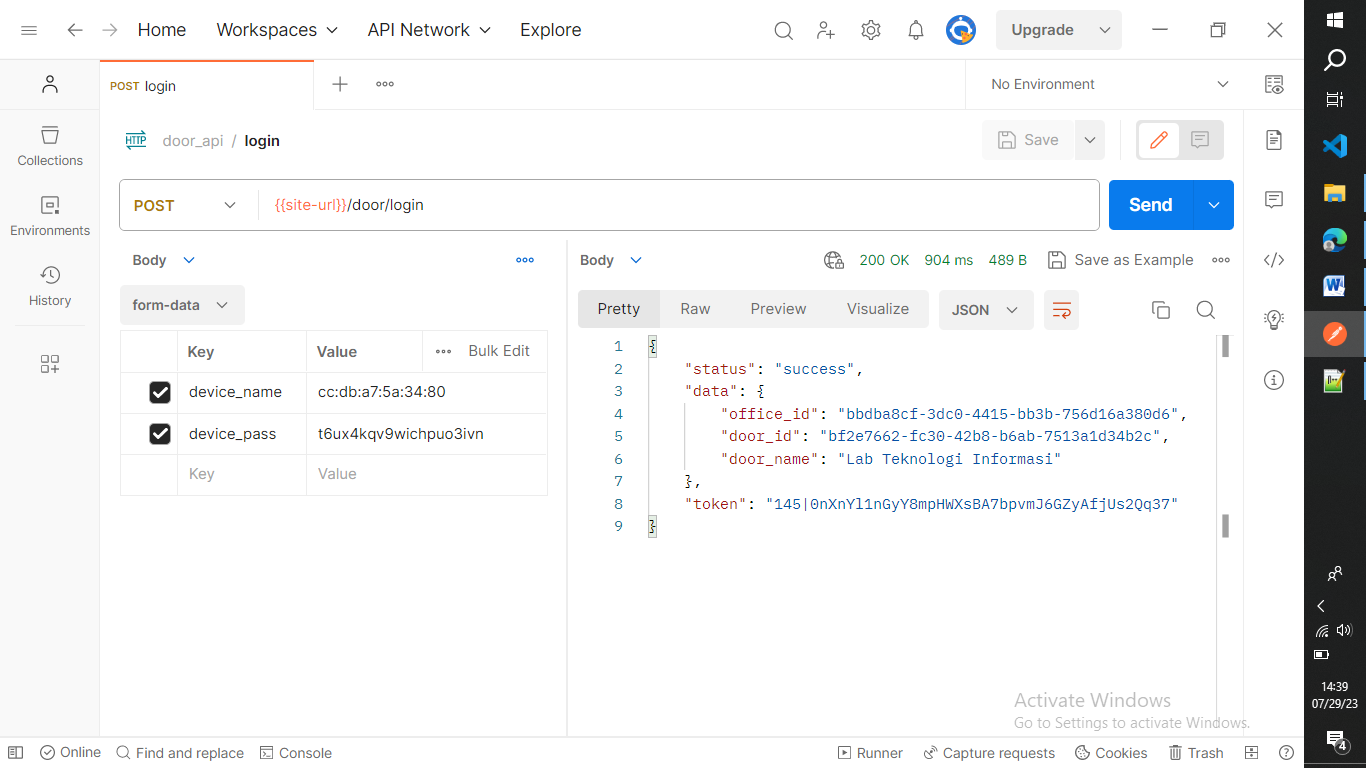


**Gambar 4.18** Aktifitas *Door* *Remote*

Gambar 4.18 diatas memperlihatkan aktifitas websoket pada *server*, pada saat API remote pintu dipanggil maka *server* akan mengirimkan perintah ke perangkat kunci pintu pada *channel* yang sudah ditentukan. Websocket akan mengirimkan data identitas dari pintu, aktifitas yang harus dilakukan serta kode kunci dari pintu tersebut. Sebagai tabahan untuk pencatatan riwayat aktifitas maka ditambahkan identitas aktor yang melakukan request ke API tersebut.

1. API *Door Login*

API *door* *login* digunakan oleh setiap perangkat kunci pintu untuk mendapatkan token akses. Setiap pintu akan mengirimkan *username* dan *password* yang dimiliki untuk mendapatkan token akses. Hasil dari API *door* *login* dapat dilihat pada gambar 4.19 dibawah.

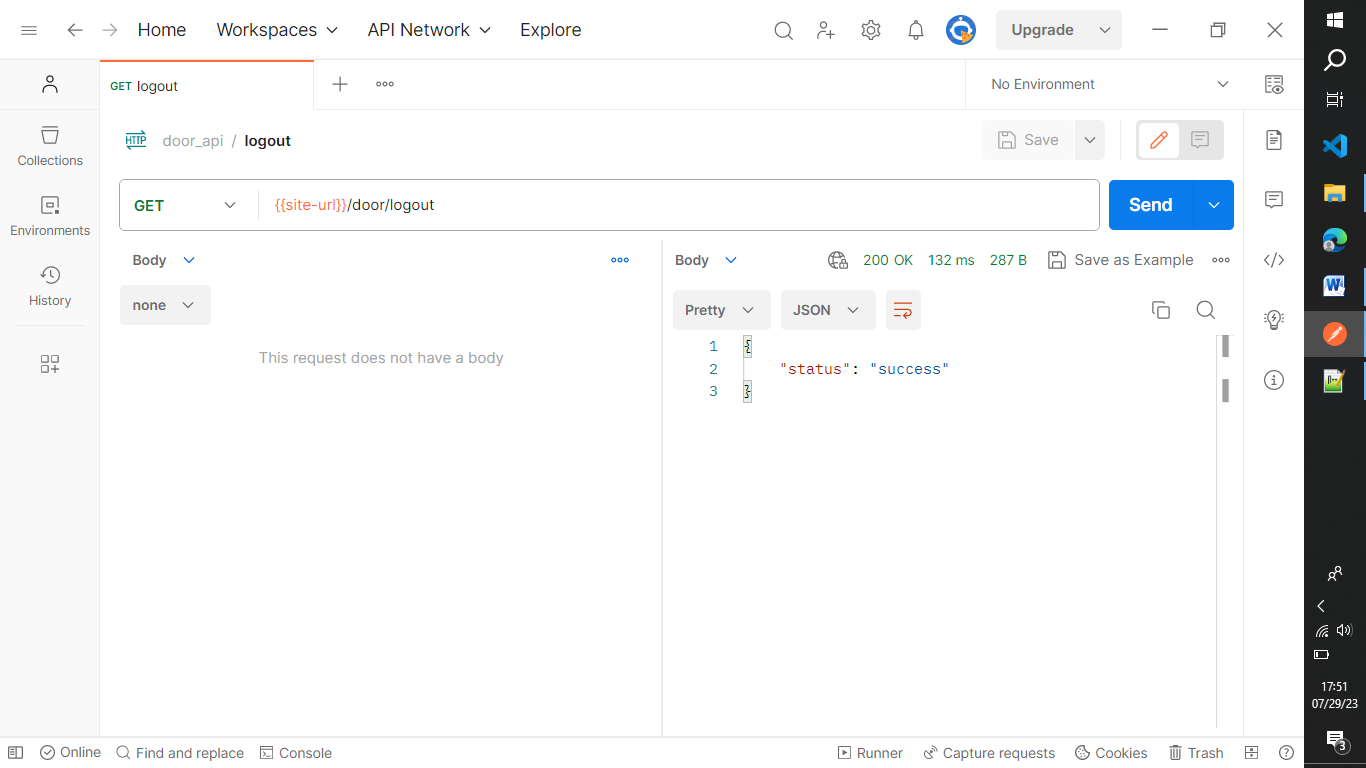


**Gambar 4.19** Hasil API *Door* *Login*

Gambar 4.19 diatas merupakan hasil dari API *door* *login* yang dikirimkan melalui *endpoint* “/door/login”. Setiap pintu akan mengirimkan *device*\_*name* sebagai *username* dan *device*\_*pass* sebagai *password* untuk mendapatkan token akses. Jika *login* berhasil maka API akan memberikan respon berupa token akses disertai dengan data pendukung seperti identitas gedung dan identitas pintu. Namun jika gagal maka API akan memberikan respon *login* gagal.

1. API *Door* *Logout*

API *door* *logout* digunakan untuk menghapus semua token yang dimiliki oleh pintu tersebut. Sesekali pintu akan melakukan permintaan *logout* untuk menghapus semua token yang disimpan didalam *database*. Hasil dari API *door* *logout* dapat dilihat pada gambar 4.20 dibawah.

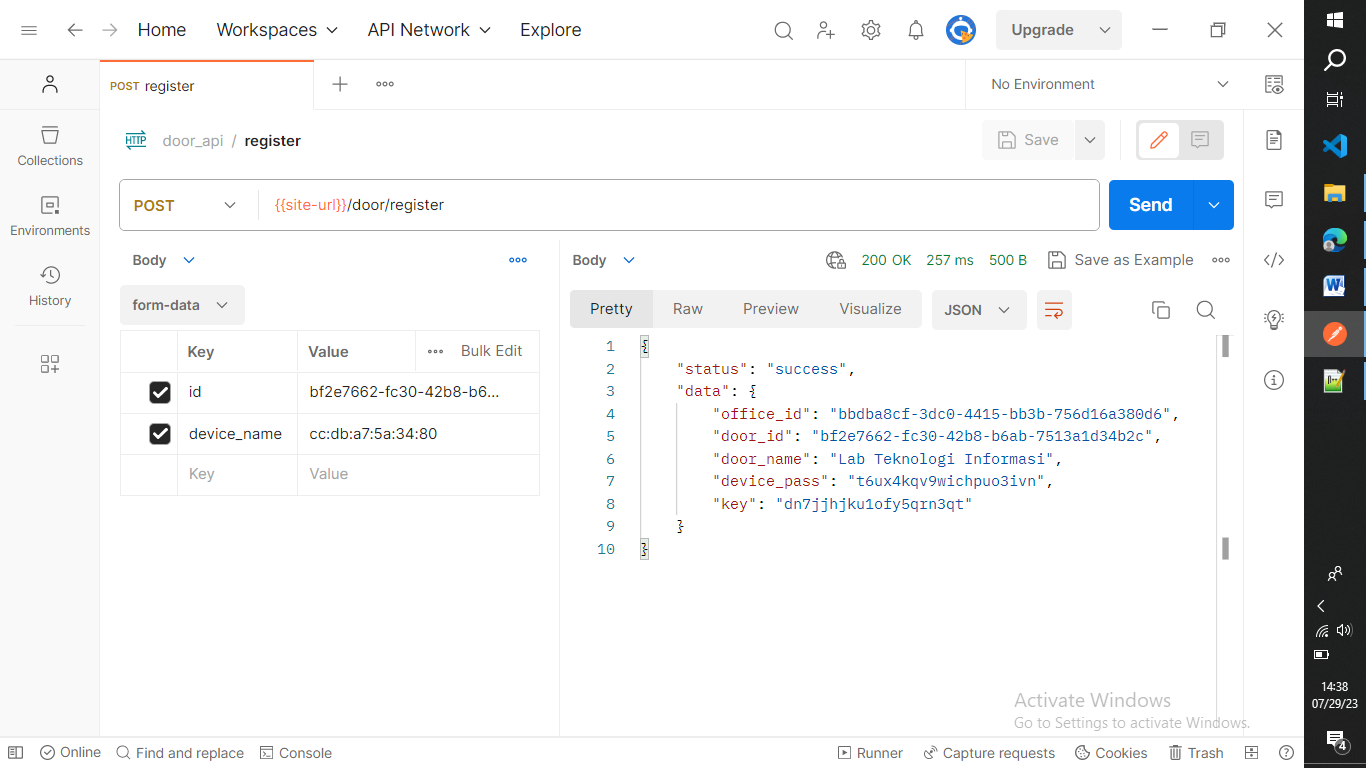


**Gambar 4.20** Hasil API *Door* *Logout*

Gamabr 4.20 diatas merupakan hasil dari API *door* *logout* yang dikirimkan melalui *endpoint* “/door/logout”. Pintu akan mengirimkan permintaan dengan melampirkan token akses yang digunakan untuk autentikasi, jika autentikasi berhasil maka API akan memberikan respon berhasil atau sukses.

1. API *Door* *Register*

API *door* *register* digunakan oleh perangkat kunci pintu untuk mendaftarkan perangkat penguncian ke *server* sehingga perangkat tersebut dapat dikenali dan dapat diatur kinerjanya oleh *server*. Pada proses pendaftaran *server* akan memperbarui data pintu pada *database*, pada proses pendaftaran *server* juga akan memberikan *password* acak yang digunakan oleh perangkat penguncian untuk melakukan proses autentikasi login. Hasil dari API *door* *register* dapat dilihat pada gambar 4.21 dibawah.

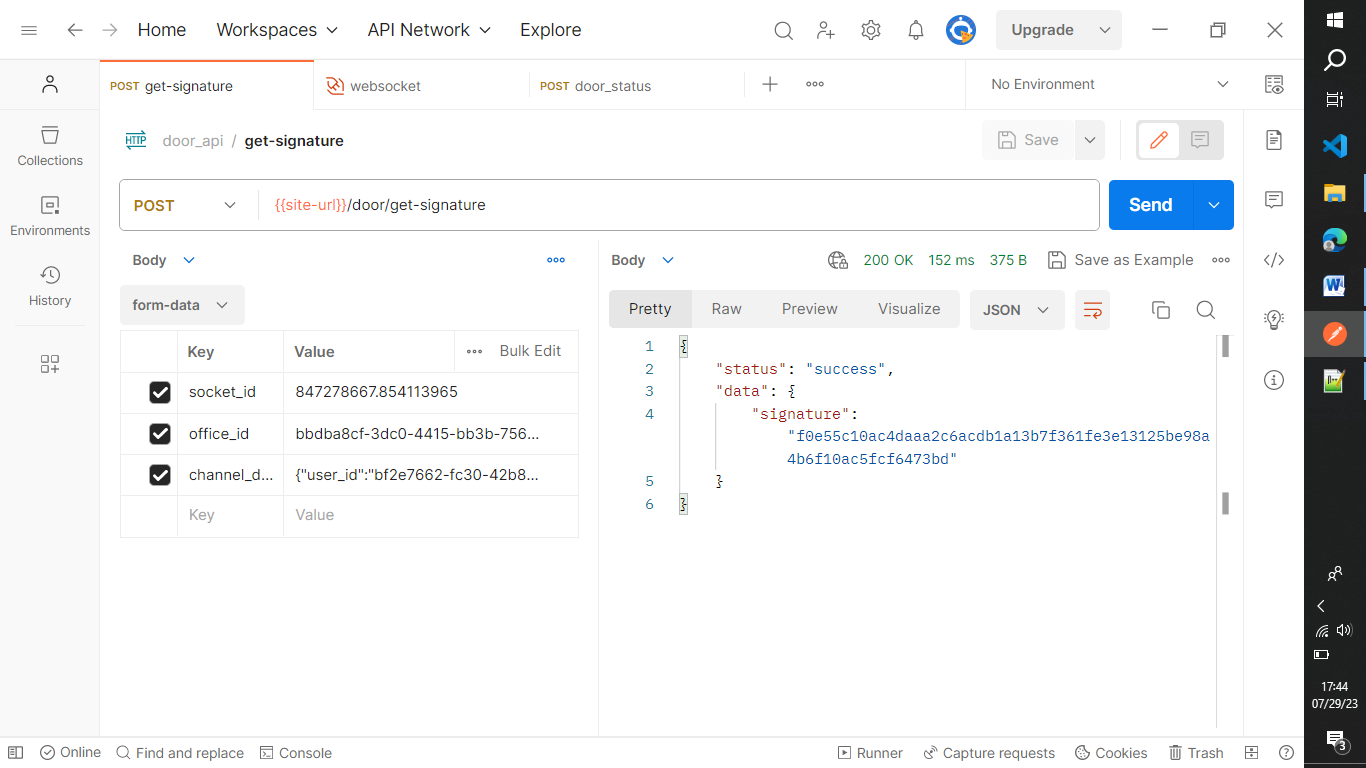


**Gambar 4.21** Hasil API *Door* *Register*

Gambar 4.21 diatas merupakan hasil dari API *door* *register* yang dikirimkan melalui *endpoint* “/door/register”. Perangkat kunci pintu akan mengirimkan identitas unik yang nantinya akan digunakan sebagai nama perangkat yang dikenali oleh server. Jika proses pendaftaran berhasil maka API akan memberikan respon sukses dan mengirimkan data seperti identitas pintu, identitas gedung dan juga *password* acak untuk melakukan proses login. Dengan menggunakan *password* acak maka *password* tersebut hanya diketahui oleh *server* dan perangkat penguncian sehingga akan meningkatkan keamanan sistem.

1. API *Door* *Signature*

API *door* *signature* digunakan oleh perangkat kunci pintu untuk mendapatkan kunci websocket. Komunikasi antara *server* ke perangkat kunci pintu dilakukan menggunakan websocket, dengan menggunakan websocket maka *server* dapat langsung memberikan perintah ke perangkat kunci pintu tanpa harus menunggu *polling* yang dilakukan oleh perangkat kuni pintu. Untuk mengamankan jalur komunikasi websocket ini menggunakan kunci *signature* yang harus dikirimkan oleh perangkat kunci untuk bisa terhubung dan menerima perintah melalui koneksi websokcet. Hasil dari API door signature dapat dilihat pada gambar 4.22 dibawah.

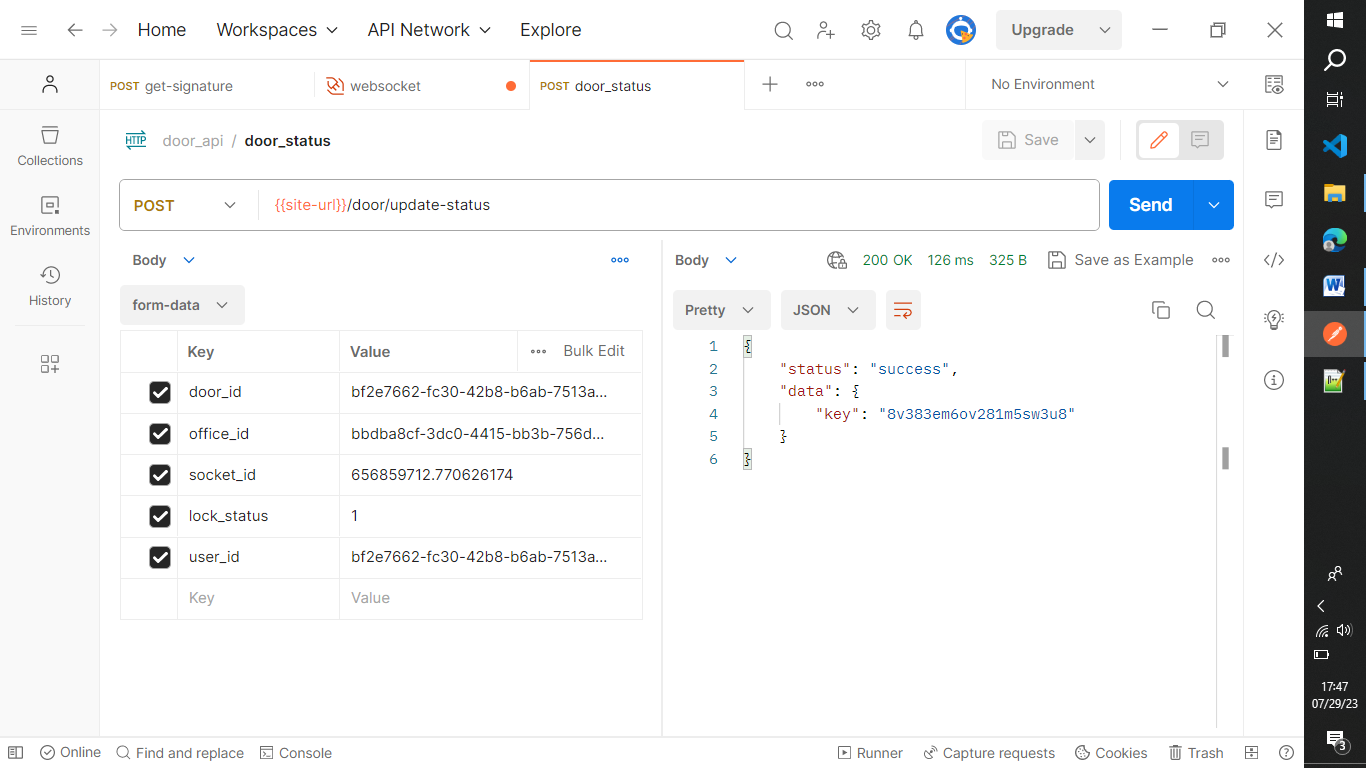


**Gambar 4.22** Hasil API *Door* *Signature*

Gambar 4.22 diatas merupakan hasil dari API *door* *signature* yang dikirimkan melalui *endpoint* “/api/signature”. Parangkat kunci pintu akan mengirimkan identitas *socket* yang sudah terjalin disertai dengan data pendukung seperti identitas gedung, identitas pintu dan tentunya token akses, kemudian *server* akan membuat kunci signature yang dapat digunakan untuk terhubung ke *channel* websocket. Dengan adanya proses autentikasi dengan menggunakan *signature* maka jalur komunikasi websocket akan lebih aman dikarenakan hanya pengguna yang memiliki kode *signature* yang sesuai yang dapat men-*subscribe* ke *channel* untuk mendapatkan data pada jalur websocket.

1. API *Door* *Update* Status

API *door* *update* status digunakan oleh perangkat kunci pintu untuk memperbarui status penguncian pintu, jika ada perubahan status pintu seperti terkunci atau terbuka maka perangkat kunci pintu akan mengirimkan status tersebut ke *server*. Proses *update* status ini akan memastikan adanya perubahan kondisi pada pintu sehingga jika ada perintah yang dikirimkan ke perangkat kunci pintu maka perangkat kunci pintu hanya akan memberikan respon jika perubahan benar benar terjadi dan memastikan bahwa perintah sudah benar-benar dikerjakan. Hasil dari API update status dapat dilihat pada gambar 4.23 dibawah.

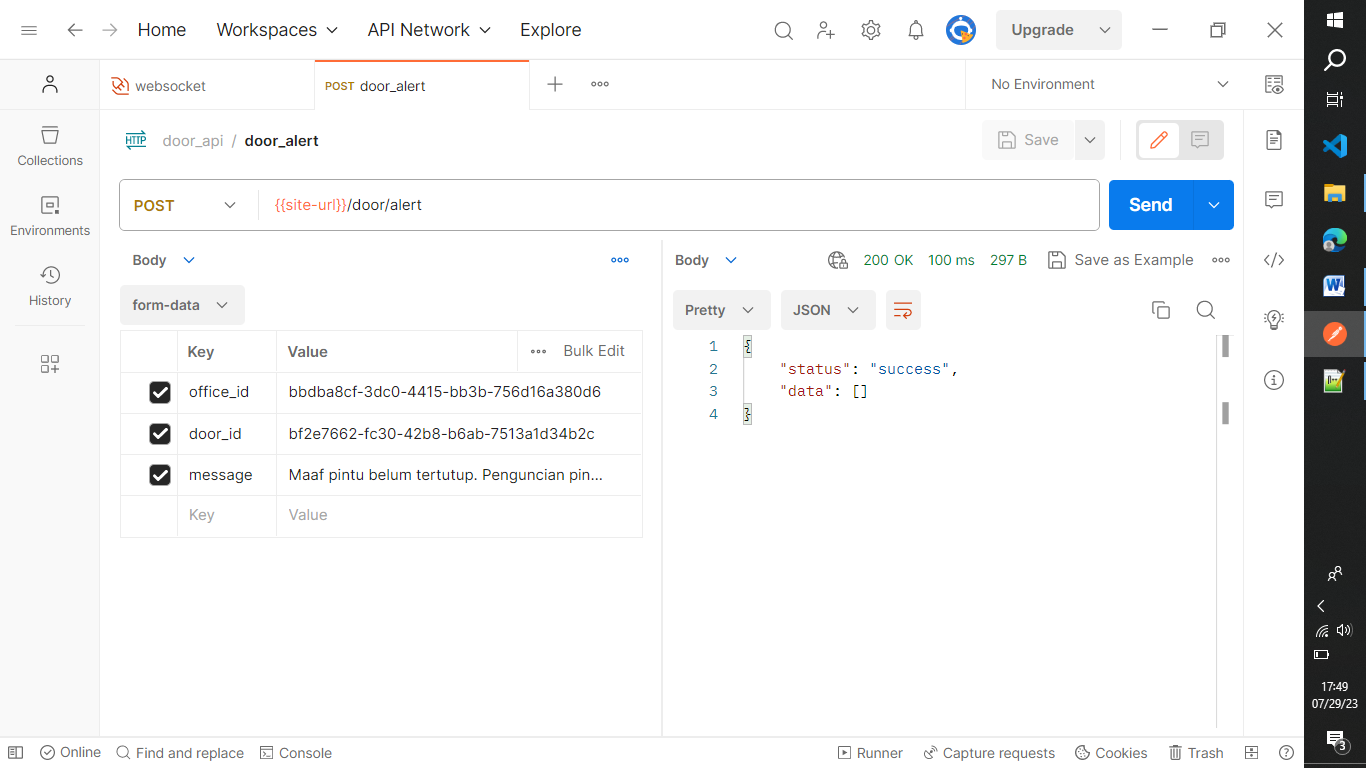


**Gambar 4.23** Hasil API *Door* *Update* Status

Gambar 4.23 diatas merupak hasil dari API *update* status yang dikirimkan melalui *endpoint* “/door/update-status”. Perangkat kunci pintu akan mengirimkan data pendukung seperti identitas pintu, identitas gedung, identitas socket, status penguncian dan identitas aktor yang melakukan kegiatan tersebut. Jika data yang dikirimkan sesuai maka server akan mengembalikan respon status sukses yang mengirimkan kunci pintu baru. Setiap melakukan update status maka *server* akan memberikan kunci pintu baru sehingga kunci pada setiap pintu bersifat dinamis, hal ini dimaksudkan untuk memperkuat keamana sistem penguncian dengan menggunakan kode kunci pintu yang berubah ubah untuk melakukan autentikasi.

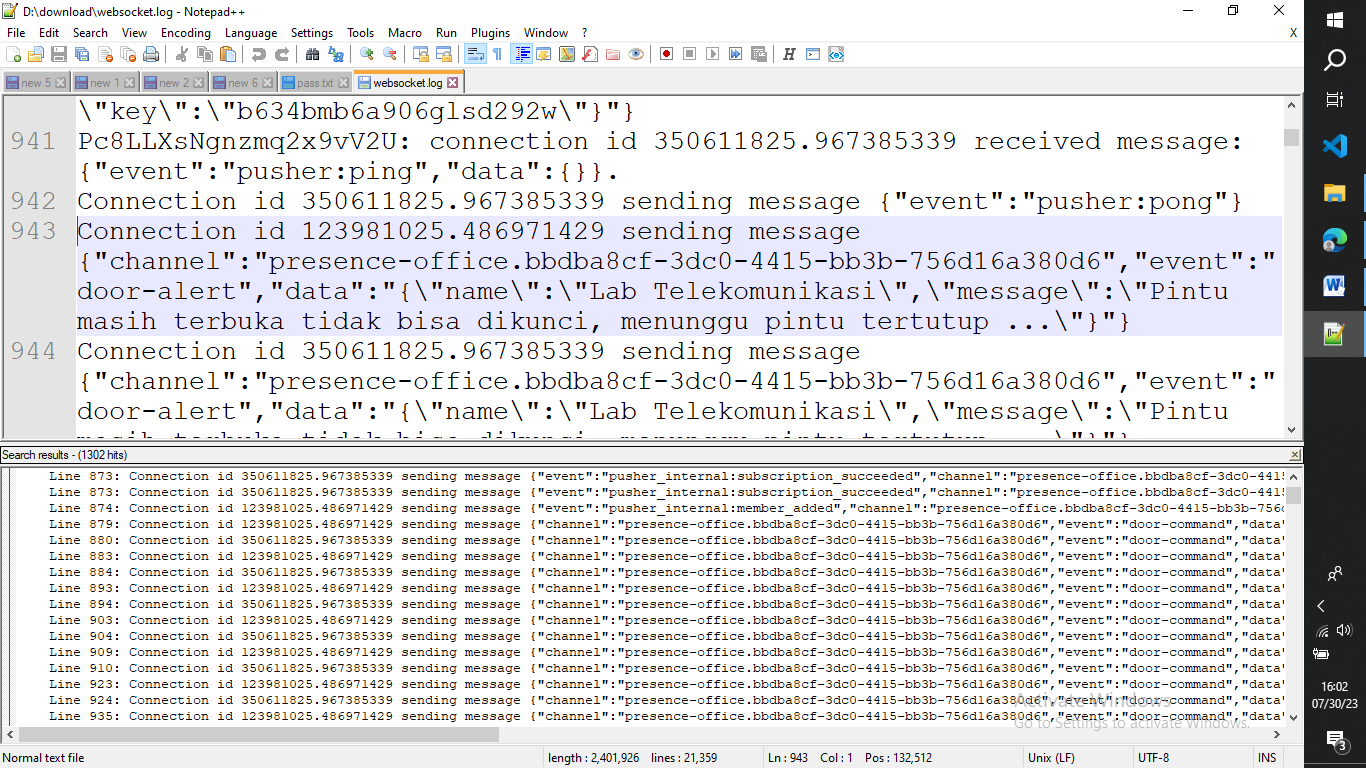
1. API *Door* *Alert*

API *door* *alert* digunakan oleh perangkat kunci pintu untuk mengirimkan status peringatan ke *server*, peringatan yang dikirmkan dapat berupa pintu yang terbuka tanpa autentikasi yang sah atau juga pintu dalam kondisi terbuka sehingga menghalangi perintah penguncian untuk dijalankan. Hasil dari API door alert dapat dilihat pada gambar 4.24 dibawah. Dengan adanya sistem peringatan ini maka sistem dapat dengan cepat kondisi yang tidak normal yang terjadi pada pintu-pintu yang sudah terpasang sehingga informasi tersebut adapat tersampaikan dengan cepat untuk sedera ditindaklanjuti lebih dalam.



**Gambar 4.24** Hasil API *Door* *Alert*

Gambar 4.24 merupakan hasil dari API *door* *alert* yang dikirimkan melalui *endpoint* “/door/alert”. Perangkat kunci pintu akan mengirimkan identitas pintu, identitas gedung dan pesan peringatan. Jika data yang dikirimkan sesuai maka server akan memberikan respon status sukses dan mengirimkan peringatan ke operator melalui koneksi websocket seperti yang terlihat pada gambar 4.25 dibawah.

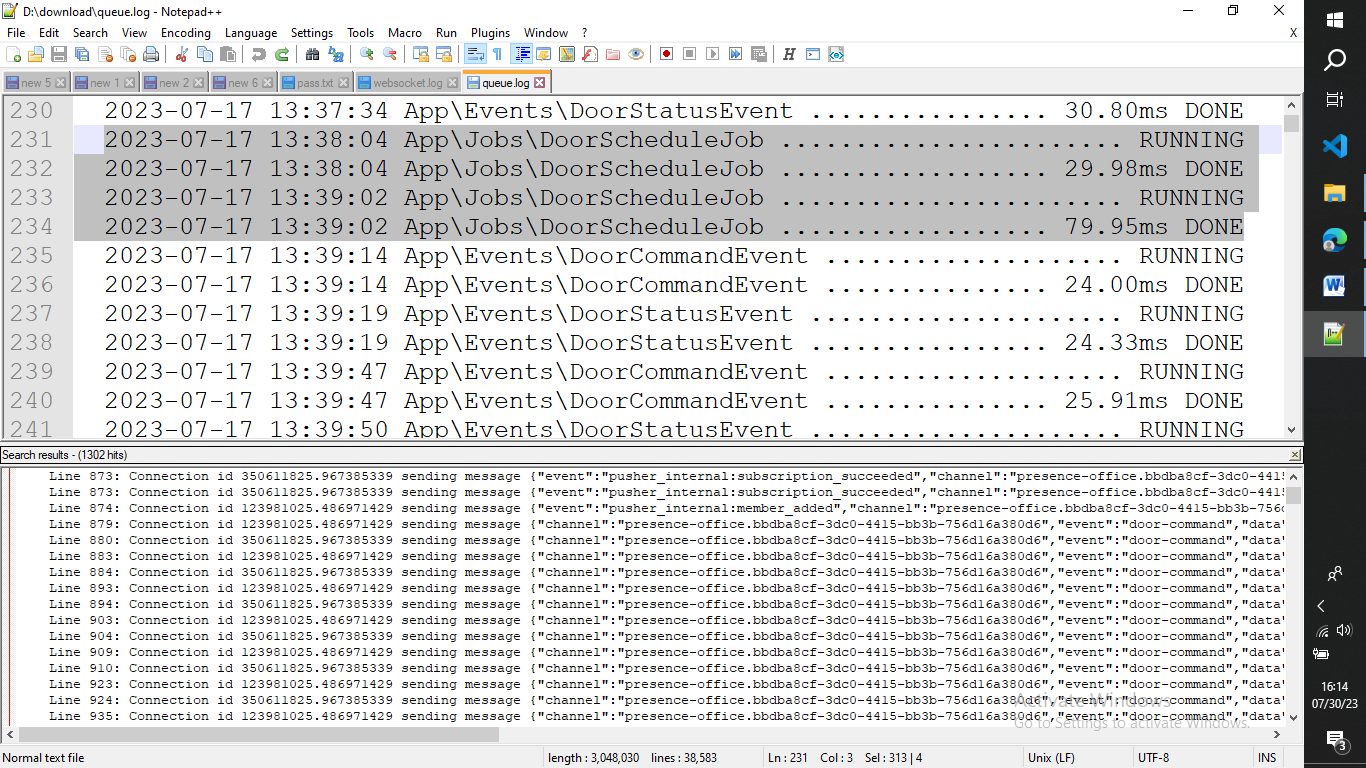


**Gambar 4.25** Aktifitas *Door* *Alert*

Gambar 4.25 diatas merupakan aktifitas dari peringatan pintu pada koneksi websocket. Setiap ada peringatan yang dikirmkan oleh perangkat kunci pintu maka server akan mengirimkan peringatan ke operator baik website maupun aplikasi mobile melalui koneksi websocket sehingga perngatan tersebut dapat diterima oleh operator secara *realtime*.

1. Pengecekan Jadwal

Pada sistem penguncian pintu gedung terdapat fitur penjadwalan yang memungkinkan operator untuk mengatur jadwal membuka atau mengunci pintu secara otomatis pada rentang waktu tertentu. Proses pengecekan jadwal akan dilakukan secara otomatis oleh *server*, jika temukan jadwal yang harus dilaksanakan maka server akan mengirimkan perintah melalui koneksi websosket seperti yang terlihat pada gambar 4.26 dibawah.



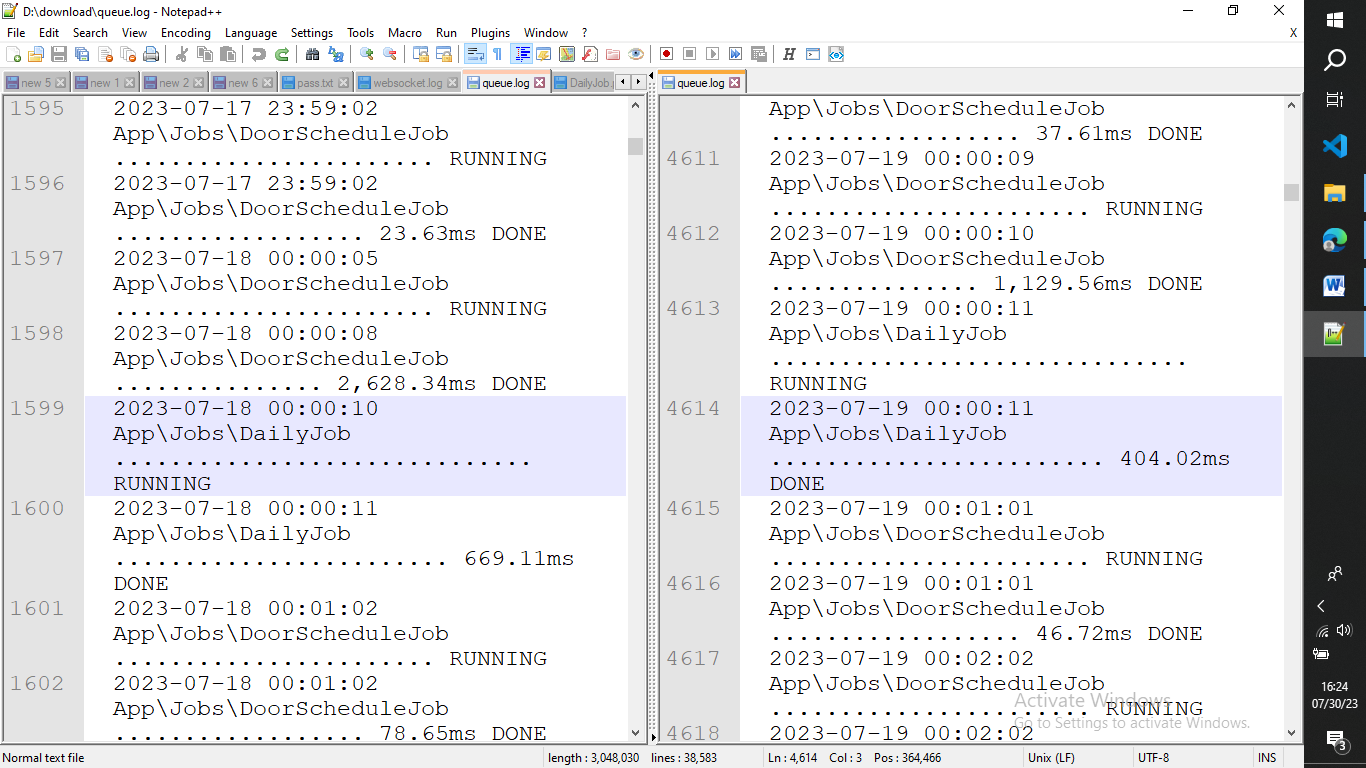
**Gambar 4.26** Pengecekan Jadwal

Pada gambar 4.26 terlihat bahwa pengecekan jadwal menggunakan sebuah *job* dari laravel yaitu *DoorSceduleJob* yang akan memeriksa jadwal pada database setiap 1 menit. Terlihat pada baris 231 (gambar 4.26) pada pukul 13:38:04 server melakukan pengecekan jadwal dan proses pengecekan selanjutkan dilakukan pada pukul 13:39:02 dan terus berulang hal ini menunjukkan bahwa pengecekan jadwal benar benar dilakukan setiap 1 menit.

Pada gambar 4.26 juga memperlihatkan bahwa saat proses pengecekan jadwal server menemukan jadwal yang harus dilaksanakan maka server akan mengirimkan perintah ke perangkat kunci pintu. Seperti yang terlihat pada baris 235 setelah ditemukan jadwal yang harus dilaksakan maka server akan mengirimkan perintah ke perangkat kunci pintu dengan menggunakan sebuah event yaitu *DoorCommandEvent* yang akan disiarkan melalui *channel* websoket.

1. Atur Ulang Jadwal

Setiap jadwal yang sudah dilaksanakan akan dihapus dari database, namun jika jadwal tersebut berulang maka akan diatur ulang sehingga dapat dijalankan kembali. Proses pengeturan jadwal dilaksanakan setiap 24 jam sekali seprti yag terlihat pada gambar 4.27 dibawah.

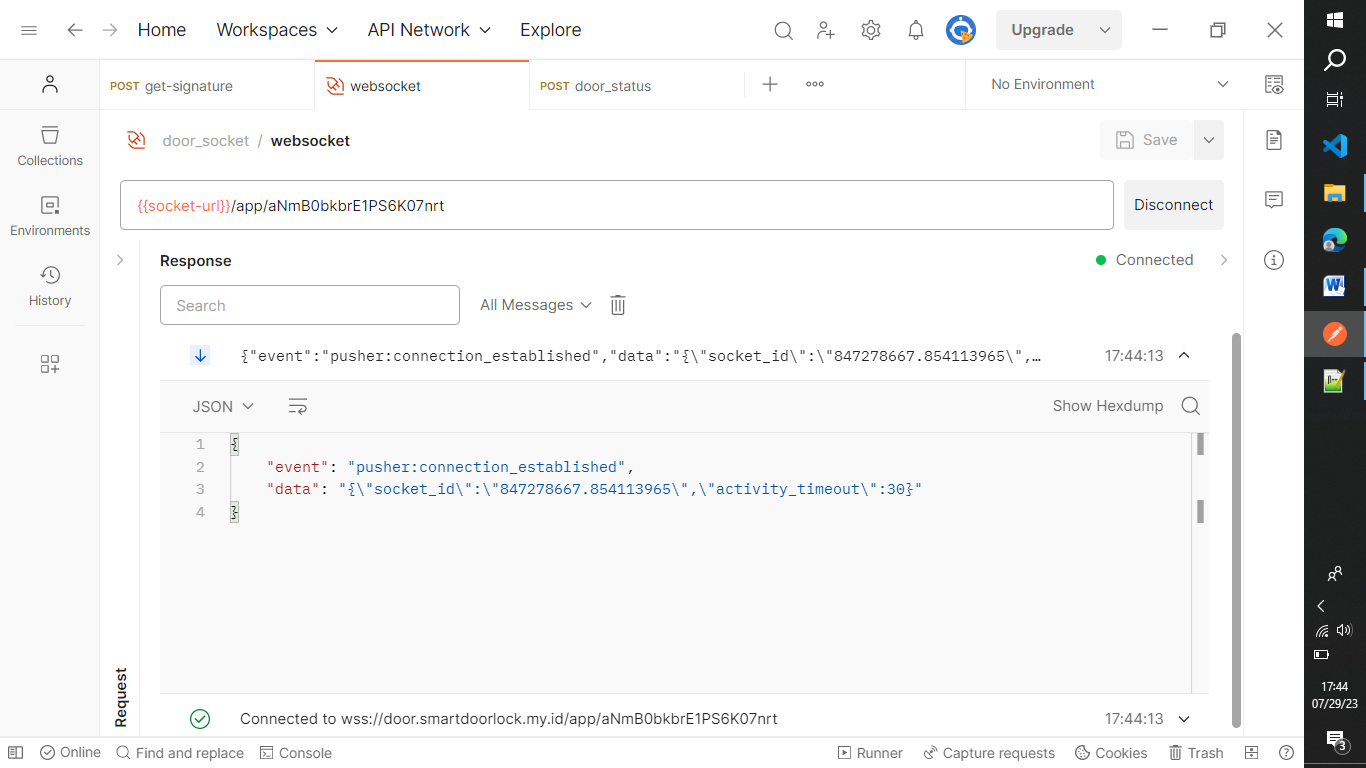


**Gambar 4.27** Atur Ulang Jadwal

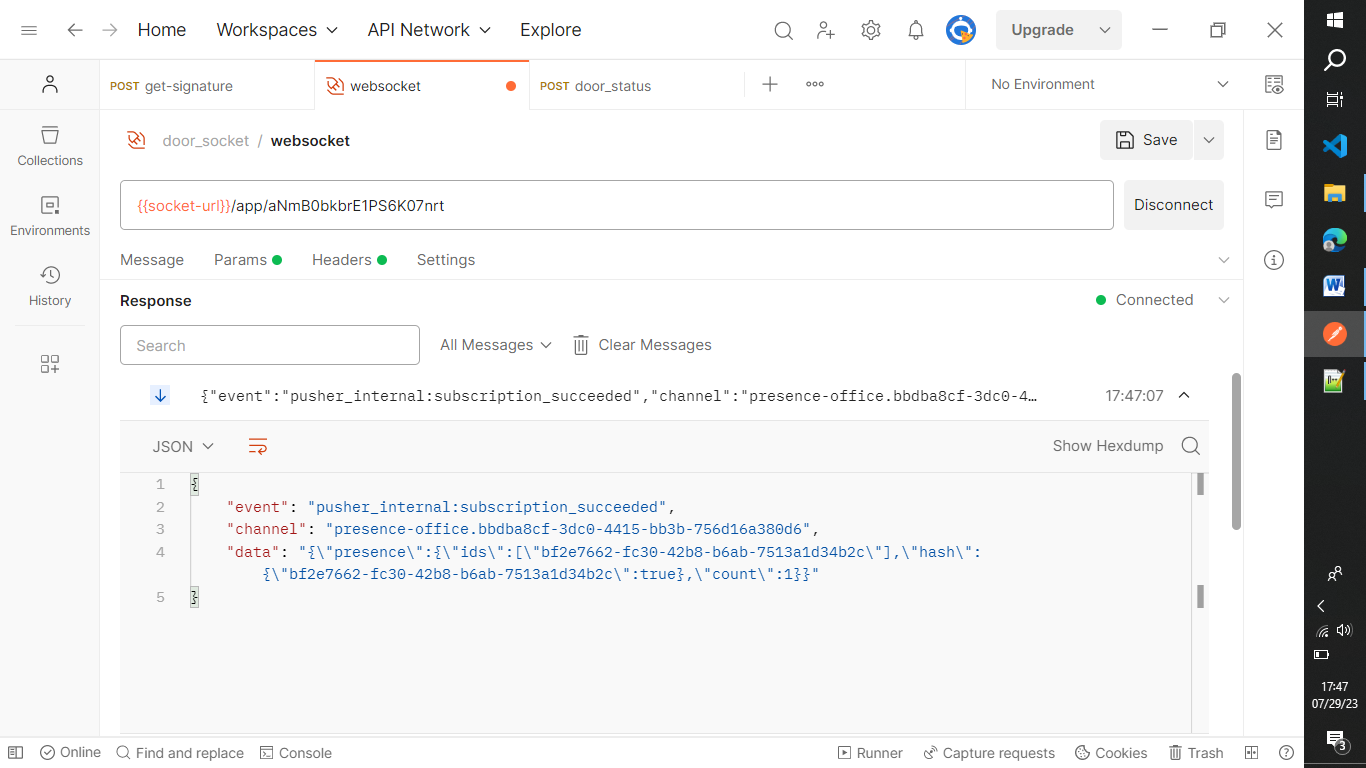
Pada gambar 4.27 terlihat bahwa pengaturan jadwal dilakukan menggunakan sebuah job laravel yaitu DailyJob. Job ini digunakan untuk menghapus jadwal yang sudah dilaksanakan atau mengatur ulang jadwal untuk kembali dilaksanakan pada hari tersebut. Pada gambar 4.27 juga terlihat pada baris 1599 dan baris 4614 memperlihatkan proses pengeturan jadwal dilaksanakan setiap 24 jam dimulai pada tanggal 18 Juli 2023 dan prose pengaturan selanjutnya dilaksanakan pada tanggal 19 Juli 2023.

1. Komunikasi Websocket

Komunikasi websoket dilakukan dengan melakukan permintaan koneksi ke enpoint websoket yang telas disediakan oleh server seperti yang terlihat pada gambar 4.28.



**Gambar 4.28** Koneksi Websocket Melalui *Endpoint*

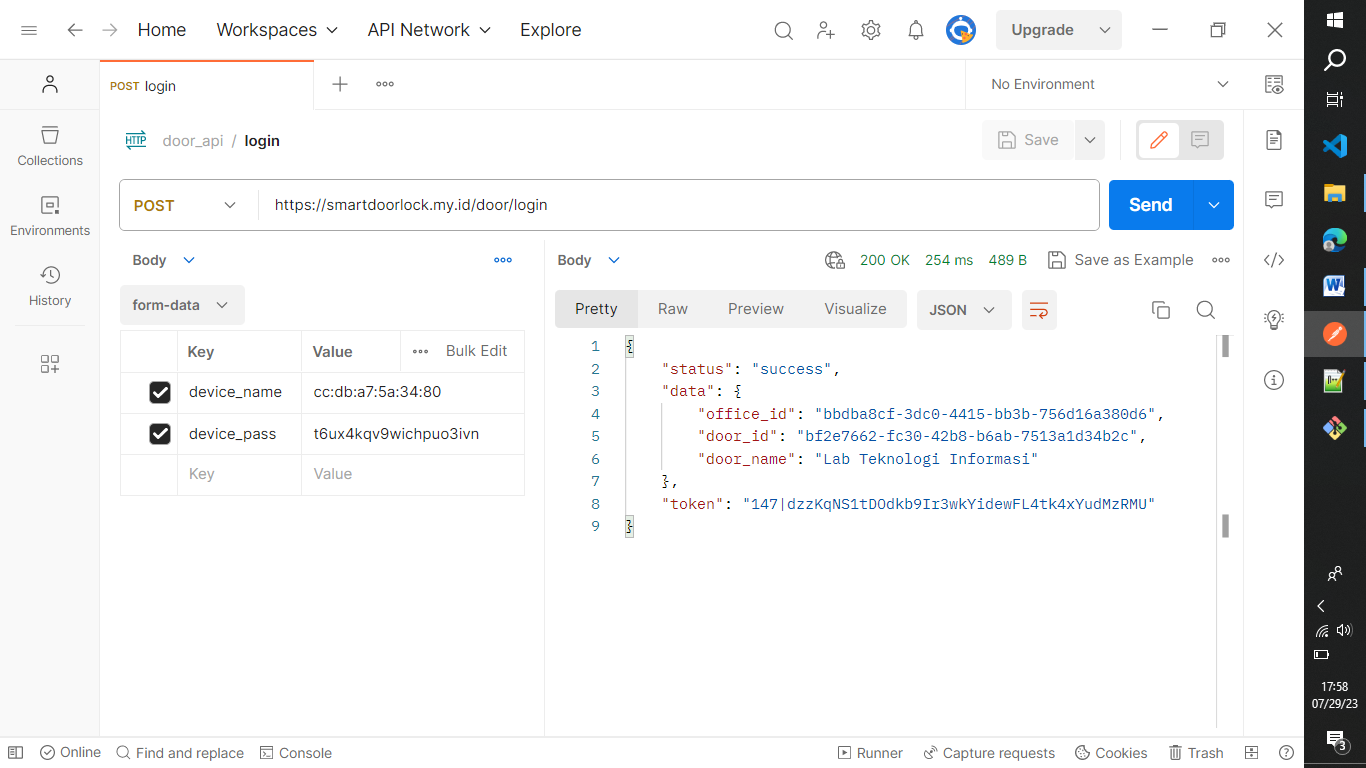


**Gambar 4.29** Proses *Subscribe* ke *Channel* Websocket

Pada gambar 4.28 diatas terlihat proses koneksi websoket dimulasi dengan mengirimkan permintaan ke enpoint websocket. Jika koneksi berhasil maka client akan menerima pesan berupa *event:connection\_establised* yang menandakan koneksi websoket sudah berhasil. Sampai tahap ini *client* tidak akan menerima data perintah yang dikirimkan melalui websocket. *Client* harus men-*subscribe* channel untuk bisa mendapatkan data perintah tersebut. Pada gambar 4.29 diperlihatkan proses *subscribe* ke *channel* dengan mengirimkan pesan *subscriber* melalui websoket dengan melampirkan kode *signature*, jika kode *signature* sesuai maka *server* akan mengirimkan pesan *pusher\_internal:subsrcription\_succeded* yang menandakan proses *subscribe* berhasil.

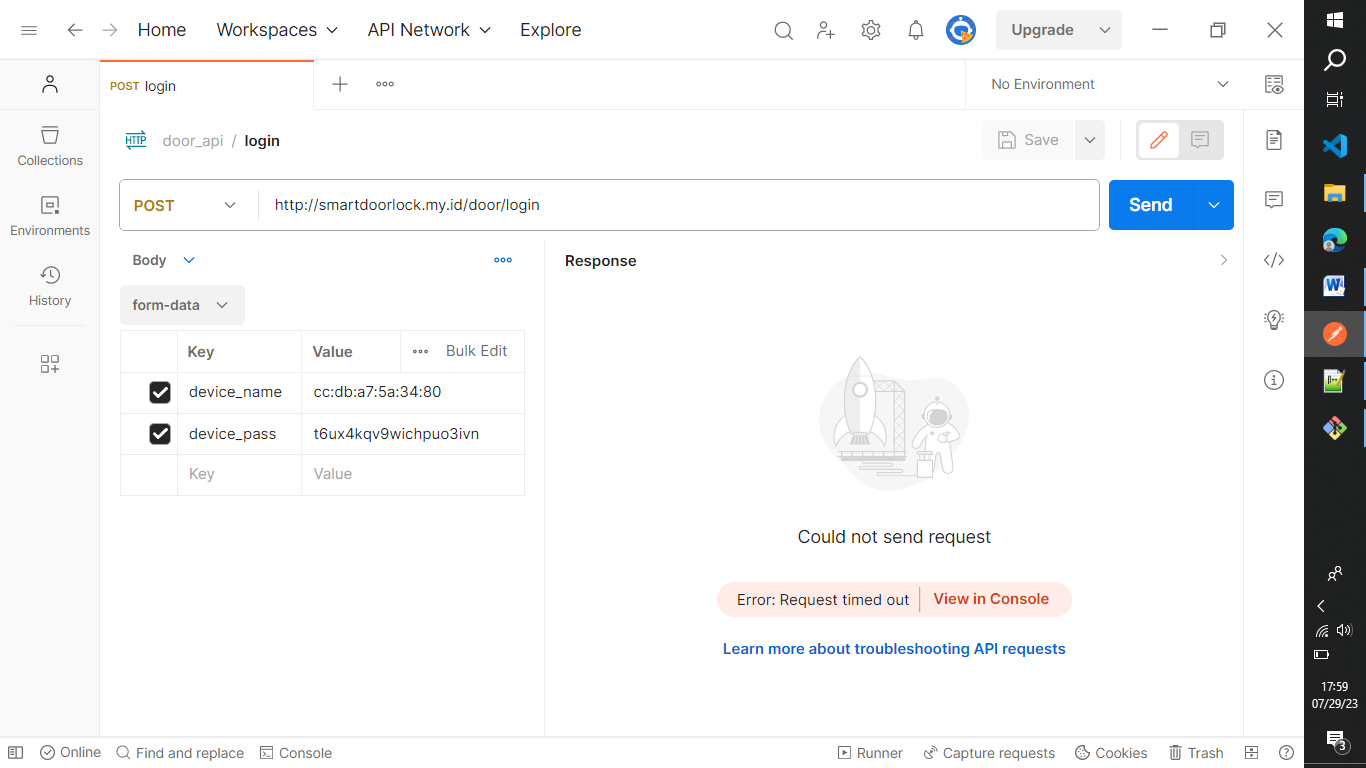
1. Koneksi HTTPS

Setiap pertukaran data pada sistem penguncian pintu gedung ini akan menggunakan koneksi HTTPS yang merupakan koneksi yang aman dengan enkripsi SSL, hal ini untuk menjamin keamanan proses transfer data. Setiap permintaan yang masuk hanya diijinkan melalui port 443 atau koneksi HTTPS selain melalui port tersebut maka permintaan akan ditolak.



**Gambar 4.30** *Request* Menggunakan HTTPS

Pada gambar 4.30 terlihat request pada berhasil diterima oleh server dan server memberikan respon status terhadapt permintaan yang dikirimkan. Dengan mengirimkan permintaan ke <https://smartdoorlock.my.id/api/login> berarti kita melakukan request menggunakan HTTPS sehingga request tersebut ijinkan oleh server.



**Gambar 4.31** *Request* Menggunakan HTTP

Pada gambar 4.31 terlihat bahwa permintaan dikirimkan menggunakan protokol HTTP, dengan menggunakan HTTP maka *server* akan menolak permintaan tersebut dan tidak akan ditanggapi oleh *server* sehingga permintaan tersebut akan *timeout* karena *server* tidak memberikan respon terhadap permintaan tersebut.

1. Pengujian

Pada tahap ini dilakukan proses pengujian pada sistem yang telah dibuat. Pengujian yang dilakukan meliputi pengujian terhadap kebutuhan fungsional maupun kebutuhan nonfungsional. Pengujian kebutuhan fungsional berfokus pada ketersediakan fitur-fitur yang telah dibuat serta kinerja dari fitur tersebut apakah sesuai atau tidak. Pengujian non-fungsional berkokus pada keandalan sistem yang telah dibuat seperti waktu respon dan tingkat keamanan.

1. Pengujian Fungsional

Pada pengujian fungsional berfokur pada fungsi dari fitur-fitur yang ada pada sistem. Pengujian fungsional dilakukan menggunakan metode *blackbox*. Pengujian *blackbox* merupakan sebuah metode pengujian perangkat lunak yang dilakukan tanpa memperhatikan struktuk kode program didalamnya. Tabel 4.1 menjelaskan beberapa fitur hasil implementasi dari kebutuhan fungsional pada sistem penguncian pintu gedung ini.

**Tabel 4.1** Hasil Pengujian Fungsional

|  |  |  |
| --- | --- | --- |
| **No** | **Kebutuhan** | **Hasil** |
| 1 | Autentikasi login dan logout | Tersedia |
| 2 | Ganti password | Tersedia |
| 3 | Reset password | Tersedia |
| 4 | Verifikasi email | Tersedia |
| 5 | Ganti profil | Tersedia |
| 6 | Lihat daftar akses | Tersedia |
| 7 | Verifikasi akses | Tersedia |
| 8 | Lihat riwayat aktifitas | Tersedia |
| 9 | Lihat daftar pintu | Tersedia |
| 10 | Membuka pintu jarak jauh | Tersedia |
| 11 | Register pintu baru | Tersedia |
| 12 | Koneksi websocket untuk pintu | Tersedia |
| 13 | Mendapatkan signature | Tersedia |
| 14 | Update status pintu | Tersedia |
| 15 | Peringatan pintu | Tersedia |

1. Fungsi *Login*

Pengujian pada fungsi *login* dilaksanakan untuk memeriksa kinerja dari fungsi *login*, *login* dikatakan berhasil jika client mendapatan respon *success* dari server disertai dengan dikirimkannya data *client* dan token akses. Hasil dari pengujian fungsi login dapat dilihat pada tabel 4.2.

**Tabel 4.2** Hasil Pengujian Fungsi *Login*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Nama** | **Bentuk Pengujian** | **Respon** | **Hasil** |
| 1 | *Login* dengan data benar | Melakukan *login* menggunakan email dan *password* yang sesuai | *success* | Berhasil |
| 2 | *Login* dengan data salah | Melakukan *login* dengan menggunakan email atau *password* yang salah | *failed* | Gagal |
| 3 | *Login* dengan data kurang | Melakukan *login* dengan menggunakan email saja atau *password* saja | *missing*\_*parameter* | Gagal |
| 4 | *Login* dengan data tidak terdaftar | Melakukan login dengan menggunakan email yang belum terdaftar | *missing*\_*parameter* | Gagal |
| 5 | *Login* dengan format tidak sesuai | Melakukan *login* dengan menggunakan *username* bukan email | *missing*\_*parameter* | Gagal |
| 6 | *Login* dengan email belum terverifikasi | Melakukan *login* dengan menggunakan email yang berlum terverifikasi | *email*\_*unverified* | Gagal |

Dapat dilihat pada tabel 4.2 proses *login* akan berhasil jika menggunakan email dan *password* yang sesuai, proses *login* juga memastikan semua parameter yang digunakan pada autentikasi tersedia dan juga sesuai. Pada proses pengujian menggunakan email yang belum terverifikasi *login* akan tertahan dengan status *email*\_*unverified* dan menunggu *client* untuk melakukan verifikasi email.

1. Fungsi *Logout*

Pengujian pada fungsi *logout* dilaksanakan untuk memeriksa kinerja dari fungsi tersebut. *Logout* dikatakan berhasil jika client menerima respon *succsess* dari *server*. Hasil dari pengujian fungsi *logout* dapat dilihat pada tabel 4.3.

**Tabel 4.3** Hasil Pengujian Fungsi *Logout*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Nama** | **Bentuk Pengujian** | **Respon** | **Hasil** |
| 1 | *Logout* dengan token | Melakukan *logout* menggunakan token yang sesuai | *success* | Berhasil |
| 2 | *Logout*  tanpa token | Melakukan *logout* tanpa menggunakan token | *Unauthenticated* | Gagal |
| 3 | *Login* dengan token salah | Melakukan *logout* dengan menggunakan token yang sudah terhapus | *Unauthenticated* | Gagal |

Dapat dilihat pada tabel 4.3 proses *logout* hanya berhasil jika menggunakan token yang sesuai, jika *client* melakukan *logout* tanpa menggunakan token atau menggunakan token yang sudah terhapus maka *logout* akan gagal dengan respon *unathenticated* atau tidak terautentikasi.

1. Fungsi Verifikasi Email
2. Fungsi Ganti Email
3. Fungsi Lupa Password
4. Fungsi update Profil
5. Fungsi Update Avatar
6. Fungsi Lihat Akses
7. Fungsi Lihat Pintu
8. Fungsi Lihat Riwayat Akses
9. Fungsi Verifikasi Akses
10. Permintaan Signature
11. Komunikasi Websocket
12. Update Status Pintu
13. Peringatan Pintu
14. Pengujian Non-Fungsional
15. Pengujian Waktu Respon Akses
16. Pengujian Keamanan HTTPS

BAB V

**PENUTUP**

1. Kesimpulan

Berdasarkan hasil pengerjaan dan analisis yang telah dilakukan, dapat disimpulkan beberapa hal sebagai berikut :

1. Saran

Setelah dilakukan analisan, perancangan dan pengujian didapatkan beberapa saran antara lain :

**DAFTAR PUSTAKA**

**BIODATA MAHASISWA**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Nama | : | Muhammad Khoiril Wafi |
| NIM | : | 21060119140133 |
| Konsentrasi | : | Teknologi Informasi |
| Tempat Tanggal Lahir | : | Demak, 4 Maret 2001 |
| Alamat Sekarang | : | Jl. Baskoro Raya No. 61, RT.03 / RW.07, Tembalang, Kec. Tembalang, Kota Semarang, Jawa Tengah 50275 |
| Alamat E-mail | : | [khoirilwafi123@gmail.com](mailto:khoirilwafi123@gmail.com) |
| Nama Orang Tua | : | Muarifin |
| Alamat Orang Tua | : | Jl. Putrajaya No. 06 02/06 Desa Karangrejo Kec. Bonang Kab. Demak |
| IPK | : | - |

**Pengalaman dan Prestasi yang Pernah Diraih:**

|  |  |
| --- | --- |
| **Nama pengalaman / prestasi** | **Tahun** |
|  |  |
|  |  |
|  |  |

Semarang, 23 Juni 2023

(Muhammad Khoiril Wafi)

NIM. 21060119140133

**LAMPIRAN A**

**SENARAI PROGRAM**

**LAMPIRAN B**

**DATA TRAINNING**

**LAMPIRAN C**

**MAKALAH TUGAS AKHIR**