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

Muhammad Khoiril Wafi *), M. Arfan dan Imam Santoso

Program Studi Sarjana Departemen Teknik Elektro, Universitas Diponegoro
Jl. Prof Sudharto, SH, Kampus UNDIP Tembalang, Semarang 50275, Indonesia

*)*Email : khoirilwafi@students.undip.ac.id*

Abstrak

Banyak gedung yang masih menggunakan sistem penguncian manual dengan menggunakan kunci fisik sehingga dalam satu gedung akan mempunyai banyak kunci untuk masing-masing pintu hal tersebut menjadikan proses pengelolaan akses pintu kurang optimal. Untuk mengatasi hal tersebut maka dilakukan perancangan mengenai sistem penguncian pintu gedung yang dapat mengoptimalkan pengelolaan akses pintu serta meningkatkan efisiensi penguncian pintu gedung. Sistem yang dibuat menggunakan konsep IoT dimana terdapat beberapa perangkat kunci untuk masing-masing pintu yang terhubung ke sebuah server sebagai pusat kendali dan data yang mengatur semua pintu serta memberikan informasi kepada pihak pengelola gedung tersebut mengenai kondisi pintu pada gedung tersebut. Beberapa fitur seperti pengawasan langsung, pindai kode QR, penjadwalan dan kendali jarak jauh akan disediakan untuk mengoptimalkan pengawasan dan pengelolaan pintu. Sistem yang dibangun terdiri dari backend API menggunakan Laravel*, database menggunakan MySQL, penjadwalan menggunakan Cron Job serta komunikasi menggunakan* Websocket *dimana semua komponen tersebut akan dipasang pada sebuah Virtual Private Server dengan menggunakan sistem operasi Ubuntu.*

***Kata Kunci**: Akses Pintu, kode QR, IoT, Server, Database, API.*

Abstract

Many buildings still use a manual locking system using physical keys so that in one building there will be many keys for each door, making the door access management process less than optimal. To overcome this, a design is carried out regarding a building door locking system that can optimize door access management and increase the efficiency of locking building doors. The system created uses the concept of IoT where there are several key devices for each door connected to a server as control and data that regulates all doors and provides information to the building manager about the condition of the doors in the building. Several features such as live monitoring, QR code scanning, scheduling and remote control will be provided to optimize door monitoring and management. The system consists of a backend API using Laravel*, a database using MySQL, scheduling using Cron Job and communication using* Websocket *where all components will be installed on a Virtual Private Server using Ubuntu operating system.*

***Keywords**: Door Access, QR code, IoT, Server, Database, API.*

I. Pendahuluan

1.1 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]. Masalah keamanan ruangan dalam sebuah gedung dan efektivitas dapat diselesaikan dengan menggunakan sebuah sistem penguncian cerdas yang terorganisasi dan terkoneksi ke sebuah sistem manajemen kunci pintu (*access control*) yang memiliki tingkat keamanan 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 aktivitas 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.

1.2 Tujuan

Tugas akhir ini bertujuan untuk merancang sistem *database* dan *server* serta sistem keamanan kunci pintu gedung dengan *access control*.

1.3 Batasan Masalah

Tugas akhir ini hanya akan membahas tentang perancangan *database* dan *server* serta sistem keamanan kunci pintu gedung dengan *access control* dengan memberikan solusi perancangan yang sesuai dengan kaidah keilmuan rekayasa.

II. Kajian Pustaka

2.1 *Access Control*

Kendali akses atau *access control* merupakan sebuah mekanisme pengaturan kebijakan yang digunakan untuk membatasi dan mengatur hak akses pengguna terhadap suatu sumber daya atau fasilitas tertentu. Kendali akses atau access control merupakan sebuah mekanisme pengaturan kebijakan yang digunakan untuk membatasi dan mengatur hak akses pengguna terhadap suatu sumber daya atau fasilitas tertentu. Dengan menggunakan kendali akses kita bisa mengatur dan membatasi akses pengguna sehingga hanya pengguna tertentu yang diizinkan yang bisa mengakses sumber daya yang dilindungi[6].

2.2 *Internet of Things*

IoT merupakan sebuah konsep yang digunakan untuk mengembangkan konektivitas 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].

2.3 *Database* MySQL

Database atau basis data merupakan sekumpulan data yang terintegrasi dan diatur sedemikian rupa sehingga data tersebut dapat dicari, diambil, ditambahkan, dan diolah dengan tepat[6].

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].

2.4 Laravel

Laravel merupakan kerangka kerja pemrograman web yang menggunakan bahasa pemrograman PHP yang terbuka dan gratis, Laravel diperuntukkan untuk pengembangan aplikasi berbasis *website* maupun API dengan menggunakan pendekatan pola MVC atau *Model View Controller*[18]. Arsitektur 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].

2.5 Websocket

Websocket merupakan sebuah protokol komunikasi web berbasis *client-server*, keberadaan *websocket* dinilai dapat menggantikan teknologi 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 terjadi 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.

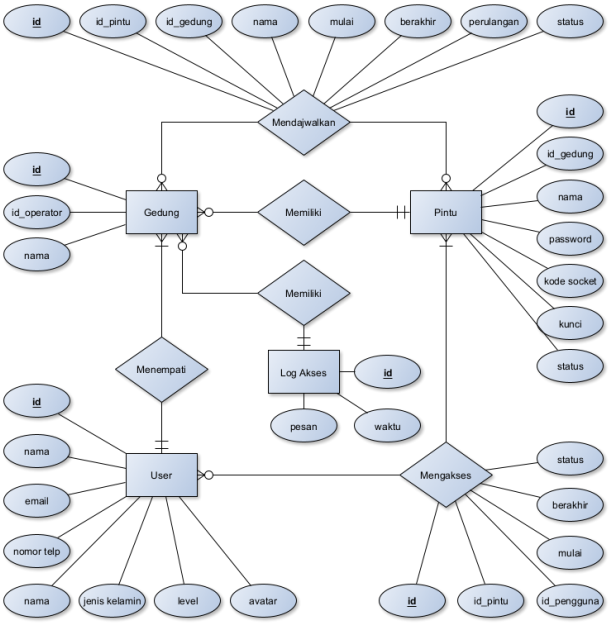
2.6 *Virtual Private Server*

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].

III. Perancangan

3.1 Database

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.



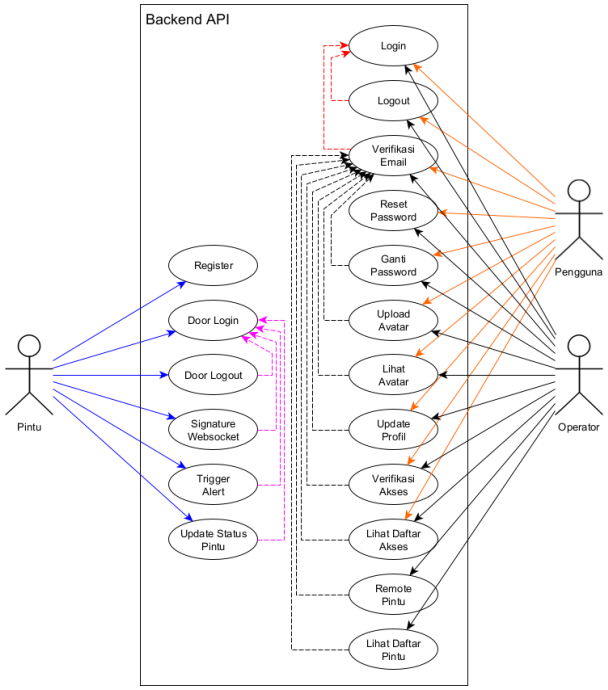
Gambar 1 ERD *database*

Pada Gambar 1 terlihat relasi antar entitas di dalam *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 menyimpan data yang akan digunakan di dalam pengelolaan sistem dengan isi data sesuai dengan atribut dari masing-masing entitas. Di dalam sistem *database* juga terdapat relasi, relasi menunjukkan hubungan antar entitas.

3.2 *Backend* API

3.2.1 *Use Case*

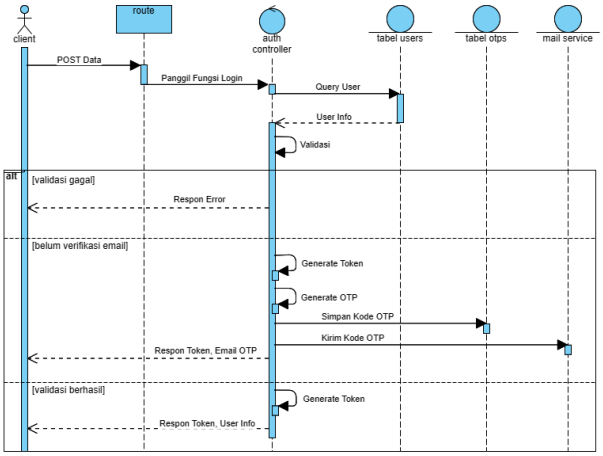
Terdapat 3 aktor yang berinteraksi dengan sistem melalui API yaitu pintu, pengguna dan operator. Pintu merupakan perangkat IoT yang digunakan untuk melakukan penguncian pada gedung, sedangkan pengguna dan operator merupakan aplikasi *mobile* yang digunakan sebagai antarmuka sistem, beberapa metode pada API mungkin membutuhkan akses *login* untuk autentikasinya. Diagram *Use Case* dari *Backend* API dapat dilihat pada Gambar 2 di bawah.



Gambar 2 Diagram *use case* API

3.2.2 Login

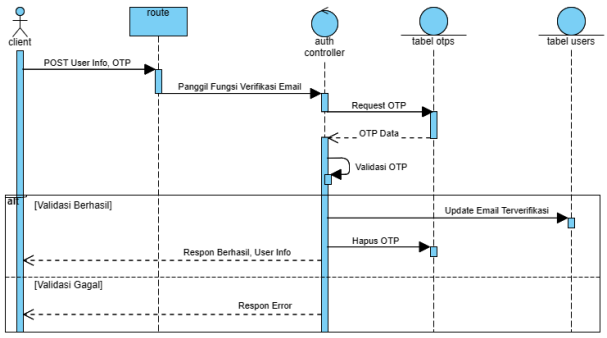
API *login* digunakan untuk melakukan autentikasi *client* melalui *username* dan *password* yang dikirimkan, API *login* juga memeriksa apakah pengguna dan operator sudah melakukan verifikasi *email*, jika belum maka *login* akan tertahan sampai pengguna melakukan verifikasi *email*. Diagram fungsi *login* dapat dilihat pada Gambar 3.



Gambar 3 API *login*

3.2.3 Verifikasi Email

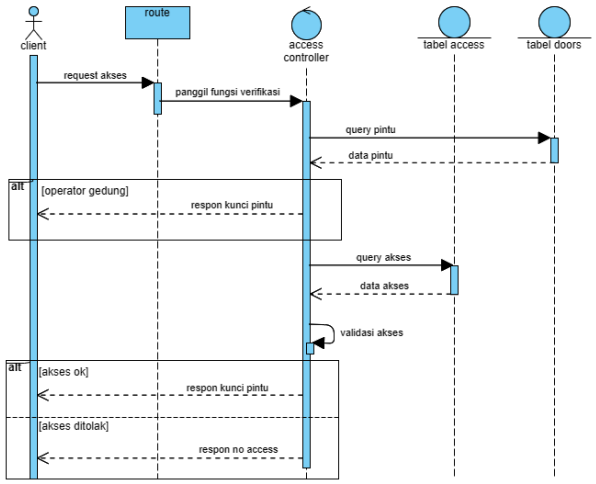
API verifikasi *email* digunakan untuk memastikan bahwa pengguna dan operator memiliki *email* yang valid dan aktif, dengan adanya verifikasi *email* maka akan meningkatkan keamanan dengan hanya mengizinkan pengguna dan operator yang terpercaya untuk mengakses sumber daya yang ada. Verifikasi *email* dilakukan dengan cara mengirimkan kode OTP (*One Time Password*) ke *email* yang telah didaftarkan. Diagram dari API verifikasi *email* dapat dilihat pada Gambar 4.



Gambar 4 API verifikasi *email*

3.2.4 Verifikasi Akses

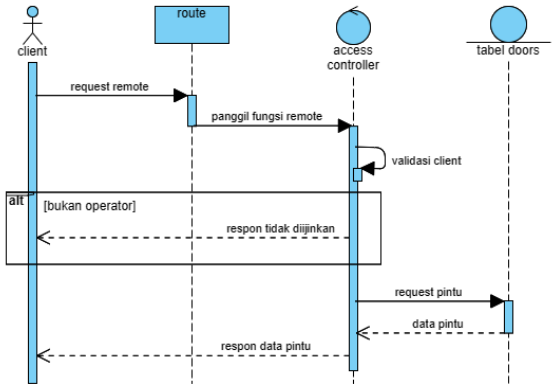
Verifikasi akses digunakan oleh pengguna dan operator untuk memverifikasi diri mereka dan untuk mendapatkan akses terhadap suatu pintu dengan cara memindai kode QR dengan perangkat *mobile*. Diagram dari API verifikasi akses dapat dilihat pada Gambar 5.



Gambar 5 API verifikasi akses

3.2.5 *Remote* Pintu

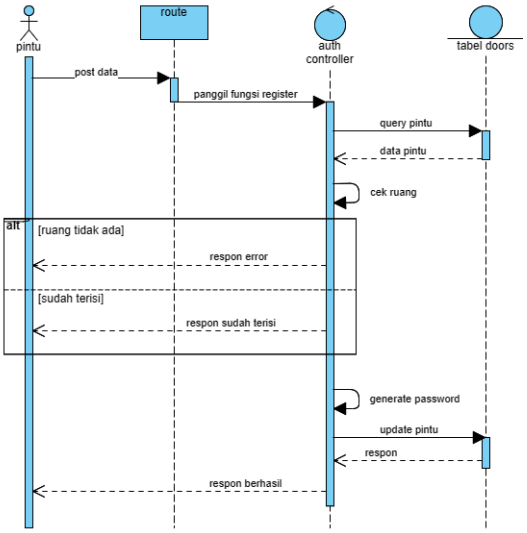
API *remote* pintu digunakan oleh operator untuk membuka atau mengunci pintu secara jarak jauh melalui aplikasi *mobile*. Dengan adanya fitur ini operator dapat mengendalikan pintu melalui aplikasi *mobile* dimana saja dan kapan saja tanpa harus berapa di ruangan operasional dengan menggunakan komputer. Diagram dari API *remote* pintu dapat dilihat pada Gambar 6.



Gambar 6 API *remote* pintu

3.2.6 Door *Register*

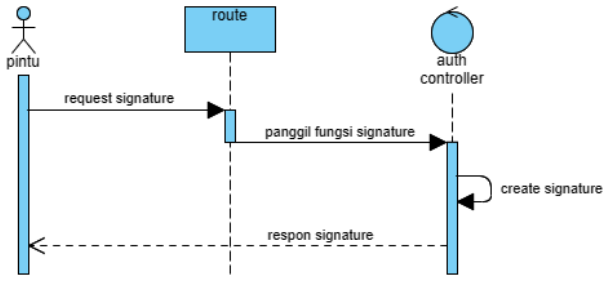
API door *register* digunakan untuk menambahkan perangkat penguncian yang baru ke dalam pintu. Pada saat operator menambahkan pintu baru melalui *dashboard website* operator maka pintu tersebut belum terpasang perangkat kunci pintu sehingga harus ditambahkan secara manual melalui prosedur pendaftaran. Diagram dari API *door register* dapat dilihat pada Gambar 7.



Gambar 7 API *door register*

3.2.7 *Door Signature*

API *door signature* digunakan untuk mendapatkan kode unik yang digunakan untuk melakukan *subscribe* ke *channel* Pusher. 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. Setelah mendapatkan nilai signature kemudian kontroler akan mengembalikan respon kode *signature* ke perangkat kunci pintu. Diagram dari API door *signature* dapat dilihat pada Gambar 8.

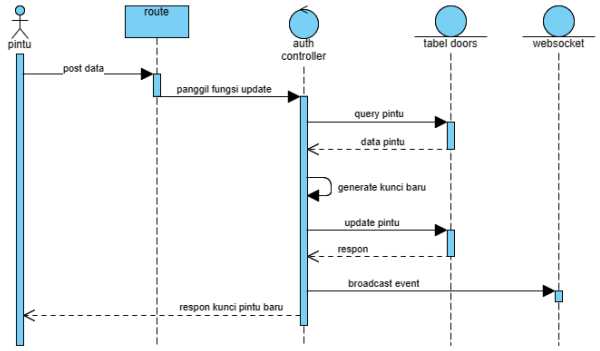


Gambar 8 API *door signature*

3.2.8 *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 *update* ini kunci pintu juga akan diperbarui sehingga meningkatkan keamanan karena kode kunci

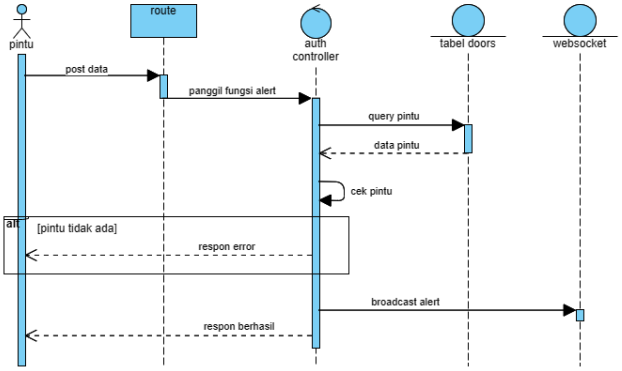
selalu berubah secara dinamis. Diagram dari API *door update* status dapat dilihat pada Gambar 9.



Gambar 9 API *Door update* status

3.2.9 *Door Alert*

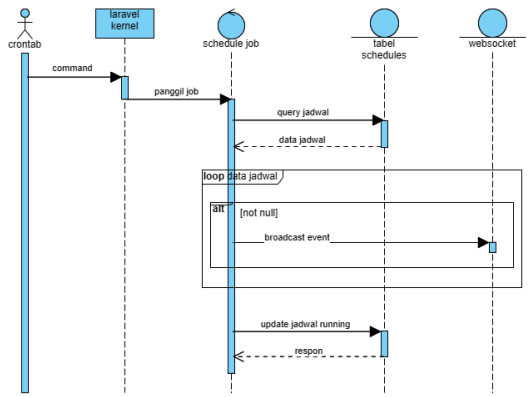
API *door alert* digunakan oleh perangkat kunci pintu untuk memberikan peringatan kepada operator bahwa pintu dalam kondisi yang tidak aman seperti terbuka tanpa autentikasi yang sah. Diagram dari API *door alert* dapat dilihat pada Gambar 10.



Gambar 10 API *door alert*

3.3 Penjadwalan

Proses penjadwalan 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 11.

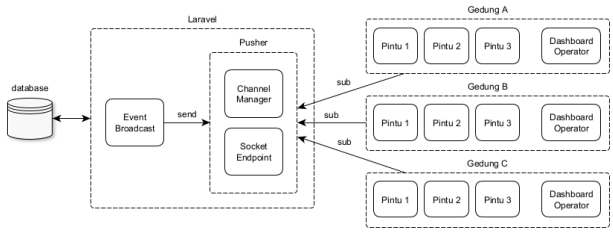


Gambar 11 Proses penjadwalan

3.4 Komunikasi Websocket

Websocket digunakan sebagai jalur komunikasi yang menghubungkan antara *server* dengan perangkat 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 12.

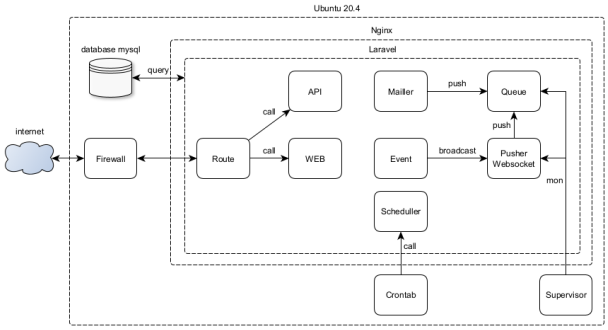


Gambar 12 Komunikasi websocket

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 di dalam gedung tersebut, untuk mengawasi semua aktivitas yang terhubung ke channel dengan mudah maka konfigurasi *channel* menggunakan *presence channel*, dengan menggunakan *presence channel* maka aktivitas semua perangkat kunci pintu seperti perangkat kunci melakukan *subscribe* dan koneksi terputus dapat diketahui secara langsung.

3.5 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 13.



Gambar 13 Konfigurasi *server*

Server dibangun menggunakan sistem operasi Ubuntu 20.04, Ubuntu merupakan bagian dari sistem operasi linux yang biasa digunakan baik untuk perangkat desktop maupun *server* karena *open source* dan ringan. Di dalam sistem operasi Ubuntu 20.04 dipasang Nginx yang digunakan sebagai *web server* untuk menjalankan aplikasi Laravel. Di dalam 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.

IV. Hasil dan Pembahasan

4.1 Pengujian

Pengujian dilakukan untuk memeriksa hasil dari perancangan dan untuk memastikan semua bagian berfungsi secara baik.

Tabel 1 Pengujian *login*

Nama	Pengujian	Respon
Login dengan data benar	Melakukan *login* menggunakan *email* dan *password* yang sesuai	*success*
Login dengan data salah	Melakukan *login* dengan menggunakan *email* atau *password* salah	*failed*
Login dengan data kurang	Melakukan *login* dengan menggunakan *email* saja atau *password* saja	*missing_parameter*
Login dengan data tidak terdaftar	Melakukan *login* dengan menggunakan *email* yang belum terdaftar	*missing_parameter*
Login dengan format tidak sesuai	Melakukan *login* dengan menggunakan *username* bukan *email*	*missing_parameter*
Login dengan *email* belum terverifikasi	Melakukan *login* dengan menggunakan *email* yang belum terverifikasi	*email_unverified*

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*.

Tabel 2 Pengujian verifikasi email

Nama	Pengujian	Respon
Verifikasi *email* tanpa token	Melakukan verifikasi *email* tanpa menggunakan token	*Unauthenticated*
Verifikasi *email* tidak sesuai	Melakukan verifikasi *email* menggunakan kode OTP yang salah	*otp_not_match*
Verifikasi *email* kadaluarsa	Melakukan verifikasi *email* menggunakan kode OTP yang sudah kadaluarsa	*otp_expired*
Verifikasi *email* sesuai	Melakukan verifikasi *email* menggunakan kode OTP yang sesuai	*success*
Verifikasi *email* token salah	Melakukan verifikasi *email* menggunakan token yang sudah terhapus	*Unauthenticated*

Proses verifikasi *email* hanya berhasil jika client mengirimkan kode OTP yang sesuai disertai dengan token yang sesuai. Jika proses verifikasi *email* menggunakan kode OTP yang salah atau sudah kadaluarsa maka proses verifikasi *email* akan gagal.

Tabel 3 Pengujian verifikasi akses

Nama	Pengujian	Respon
Verifikasi akses tanpa token	Melakukan *request* lihat verifikasi akses tanpa menggunakan token	*Unauthenticated*

Tabel 3 (lanjutan)

Nama	Pengujian	Respon
Verifikasi akses dengan token	Melakukan *request* verifikasi akses menggunakan token yang sesuai	*success*
Verifikasi akses dengan id pintu salah	Melakukan *request* verifikasi akses menggunakan identitas pintu yang salah	*no_data*

Proses verifikasi akses berhasil jika permintaan disertai dengan token dan identitas pintu sesuai, jika identitas pintu tidak sesuai maka proses verifikasi akses akan gagal karena pintu tidak ditemukan. Jika permintaan tidak disertai dengan token maka permintaan tersebut akan ditolak.

Tabel 4 Pengujian *signature*

Nama	Pengujian	Respon
Signature tanpa token	Melakukan *request signature* tanpa menggunakan token	*Unauthenticated*
Signature dengan token	Melakukan *request signature* menggunakan token dan data lengkap	*success*
Signature data kurang	Melakukan *request signature* menggunakan data yang kurang	*missing_parameter*

Proses fungsi *signature* berhasil jika permintaan yang dikirimkan disertai dengan token dan data yang dikirimkan lengkap, jika permintaan yang dikirimkan tanpa menggunakan token atau ada data yang kurang maka permintaan akan gagal.

Tabel 5 Pengujian *websocket*

Nama	Pengujian	Respon
Subscription dengan *signature*	Melakukan *subscription* ke *channel* websocket menggunakan signature sesuai	*Subscription Succeed*

Tabel 5 (lanjutan)

Nama	Pengujian	Respon
Subscription dengan *signature*	Melakukan *subscription* ke *channel* websocket menggunakan signature sesuai	*Subscription Succeed*
Subscription tanpa *signature*	Melakukan *subscription* ke *channel websocket* tanpa menggunakan *signature*	*Invalid Signature*
Subscription dengan *signature* salah	Melakukan *subscription* ke *channel websocket* menggunakan *signature* yang tidak sesuai	*Invalid Signature*

Proses *subscription* pada *channel websocket* berhasil jika menggunakan kode *signature* yang sesuai, jika tidak menggunakan kode *signature* atau menggunakan kode *signature* yang salah maka proses *subscription* akan gagal.

Tabel 6 Pengujian *update* status pintu

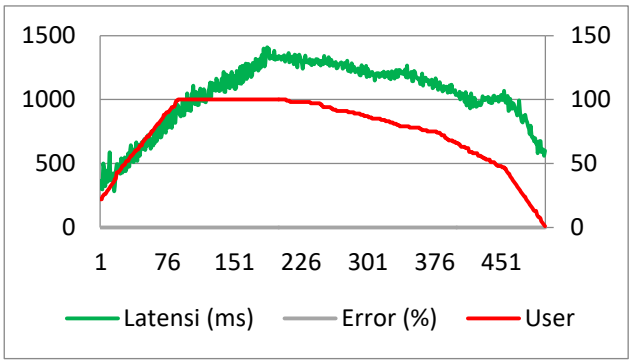
Nama	Pengujian	Respon
Update status tanpa token	Melakukan *update* status pintu tanpa token	*Unauthenticated*
Update status dengan token	Melakukan *update* status pintu dengan menggunakan token dan data lengkap	*success*
Update status data tidak lengkap	Melakukan *update* status pintu dengan menggunakan data yang tidak lengkap	*missing_parameter*
Update status id pintu salah	Melakukan *update* status pintu menggunakan data identitas pintu yang salah	*failed*

Proses *update* status pintu berhasil jika request dikirimkan dengan token dan data yang lengkap, jika ada data yang kurang lengkap atau tidak disertai dengan token akan request tersebut akan gagal.

Tabel 7 Pengujian peringatan pintu

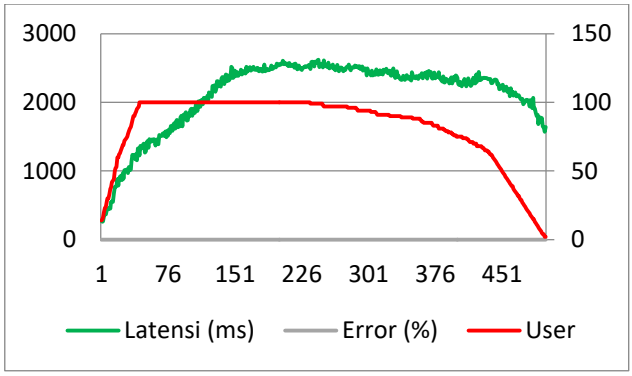
Nama	Pengujian	Respon
Peringatan pintu tanpa token	Mengirim peringatan pintu tanpa menggunakan token	*Unauthenticated*
Peringatan pintu sesuai	Mengirim peringatan pintu dengan menggunakan token dan data lengkap	*success*
Peringatan pintu data tidak lengkap	Mengirim peringatan pintu dengan menggunakan data yang tidak lengkap	*missing_parameter*
Peringatan pintu id pintu salah	Mengirim peringatan pintu menggunakan data identitas pintu yang salah	*failed*

Proses peringatan pintu berhasil jika *request* dikirimkan dengan token dan data yang lengkap, jika ada data yang kurang lengkap atau tidak disertai dengan token akan request tersebut akan gagal.



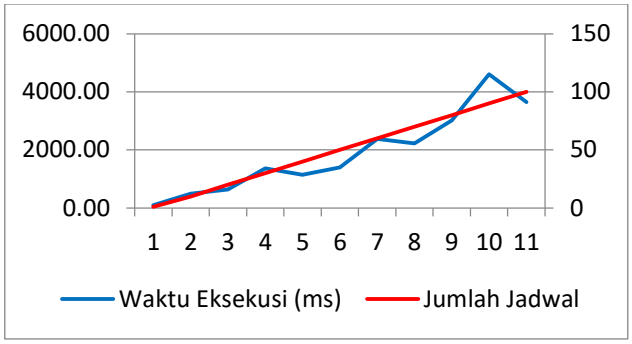
Gambar 14 Performa verifikasi akses

Pengujian performa verifikasi akses dilakukan dengan mensimulasikan beban permintaan verifikasi akses sebanyak 100 pengguna berbeda secara bersamaan. Dari hasil pengujian didapatkan bahwa pada puncak jumlah pengguna sistem membutuhkan waktu rata-rata 1.3 detik untuk memberikan respon ke pengguna dengan rasio *error* 0.0%.



Gambar 15 Performa *update* status pintu

Pengujian performa *update* status pintu dilakukan menggunakan Jmeter. Proses pengujian mendapatkan hasil bahwa pada beban 100 pintu berbeda melakukan update status secara bersamaan maka sistem memerlukan waktu rata-rata 2.5 detik dengan rasio *error* 0.0%.



Gambar 16 Performa penjadwalan

Pengujian dilakukan dengan mencatat waktu pemeriksaan untuk setiap jadwal, yaitu terdapat 11 titik pengujian dimulai dari 1 jadwal sampai 100 jadwal dengan masing-masing jadwal terdapat 20 pintu. Dapat dilihat pada Gambar 4.34 di atas, semakin besar jumlah jadwal yang ada maka waktu yang diperlukan pada proses penjadwalan akan semakin lama dimana untuk 1 jadwal memerlukan waktu 96.00 milidetik dan 100 jadwal memerlukan waktu 3641.14 milidetik. Dari hasil pengujian menggunakan 100 jadwal waktu yang diperlukan yaitu 3.6 detik dimana nilai tersebut masih di bawah dari periode pengecekan jadwal (1 menit) sehingga sistem masih dapat menangani 100 proses penjadwalan dengan aman.

4.2 Pembahasan

Pengembangan sistem *database* dan *server* pada sistem keamanan kunci pintu gedung dengan akses kontrol ini merupakan bagian dari pengembangan sistem penguncian gedung berbasis IoT yang dibangun dengan tujuan untuk meningkatkan keamanan dan efisiensi pengelolaan kunci pada sebuah gedung dimana di dalam

satu gedung tersebut terdapat banyak ruangan. Sistem yang dibangun dapat meningkatan keamanan dan efisiensi penguncian dengan menggunakan beberapa fitur seperti kode QR, kendali jarak jauh, peringatan penerobosan dan penjadwalan.

Perancangan *database* dan *server* pada sistem keamanan kunci pintu menggunakan metode *waterfall*, pengembangan dilakukan secara berurutan dimulai dari proses pengumpulan kriteria kebutuhan, implementasi sampai ke pengujian dan pengiriman. Sistem *server* dibangun menggunakan sistem operasi Ubuntu 20.04 dengan dengan *database* MySQL dan *backend* API Laravel. Penggunaan Ubuntu sebagai *server* memberikan banyak kemudahan seperti adanya dukungan Cron Job sebagai mekanisme penjadwalan dan Nginx sebagai HTTP *server* yang ringan dan cepat. Laravel dan MySQL digunakan sebagai bagian dari *backend* dengan memanfaatkan beberapa fitur seperti autentikasi, penyiaran, penjadwalan yang disediakan oleh Laravel serta *database* relasional yang disediakan oleh MySQL. Pada tahap pengujian menggunakan alat pengujian seperti Jmeter dan Postman, sistem dapat memberikan kinerja yang diinginkan dan juga dapat menangani banyak pengguna secara bersamaan.

V. Penutup

5.1 Kesimpulan

Berdasarkan hasil perancangan dan pengujian yang telah dilakukan, dapat disimpulkan beberapa hal sebagai berikut:

1. Metode *access control* yang digunakan pada penelitian ini dapat meningkatkan keamanan penguncian pada pintu gedung dibuktikan dengan hasil pengujian pada proses verifikasi akses yang mana hanya pengguna tertentu yang dapat membuka kunci pintu sesuai dengan izin yang diberikan oleh operator.
2. Metode penjadwalan yang dirancang pada penelitian ini dapat meningkatkan efisiensi penguncian pada sebuah gedung dibuktikan pada hasil pengujian penjadwalan untuk membuka 20 pintu hanya membutuhkan waktu 96.00 milidetik.
3. API yang telah dikembangkan menggunakan Laravel pada penelitian ini secara umum dapat menangani permintaan secara bersamaan dengan waktu respon sekitar 140 milidetik dengan rasio *error* 0%.
4. Pada *endpoint* API yang kemungkinan besar akan diakses secara bersamaan seperti verifikasi akses memerlukan waktu respon 1.3 detik pada beban 100 pengguna sedangkan untuk *endpoint update* status pintu memerlukan waktu 2.5 detik untuk beban 100 pintu.
5. Proses penjadwalan 100 jadwal pintu memerlukan waktu eksekusi sebesar 3.6 detik sehingga proses penjadwalan dapat berjalan dengan aman menggunakan periode pengecekan jadwal 1 menit sekali.

5.2 Saran

Setelah dilakukan proses perancangan dan pengujian didapatkan beberapa hal yang dapat diperbaiki atau ditingkatkan yaitu perlu adanya perancangan lebih lanjut untuk mempercepat respon *server* terutama pada beban *request* secara bersama-sama.

DAFTAR PUSTAKA

[1] K. Y. Sun, Y. Pernando, and M. I. Safari, "Perancangan Sistem IoT pada Smart Door Lock Menggunakan Aplikasi BLYNK," *JUTSI (Jurnal Teknol. dan Sist. Informasi)*, vol. 1, no. 3, pp. 289–296, 2021, doi: 10.33330/jutsi.v1i3.1360.

[2] I. Hermawan, D. Arnaldy, P. Oktivasari, and D. A. Fachrudin, "Development of Intelligent Door Lock System for Room Management Using Multi Factor Authentication," vol. 16, no. 1, pp. 1–14, 2023.

[3] F. As *et al.*, "Design and Construction of a Smart Door Lock With an Embedded Spy-Camera," *J. Multidiscip. Eng. Sci. Technol.*, vol. 8, no. October, pp. 2458–9403, 2021, [Online]. Available: https://www.researchgate.net/publication/354872757

[4] A. Jain, V. L. Kalyani, and B. Nogiya, "RFID and GSM Based Attendance Monitoring System using door locking / unlocking system and Its Hardware Implementation," *J. Manag. Eng. Inf. Technol.*, vol. 2, no. 3, pp. 5–10, 2015.

[5] M. Kasyful Anwar, "Perancangan Database IoT Berbasis Cloud dengan Restful API Cloud-Based IoT Database Design with Restful API," vol. 20, no. 2, pp. 268–279, 2021.

[6] S. Artikel, "Jurnal Nasional Teknologi dan Sistem Informasi Pembangunan Auto Backup SQL Database Server Menggunakan Raspberry Pi : Studi Kasus," vol. 03, pp. 130–137, 2018.

[7] P. Simanjuntak, C. E. Suharyanto, and Jamilah, "Analisis Penggunaan Access Control List (Acl) Dalam Jaringan Komputer Di Kawasan," *Isd*, vol. 2, no. 2, pp. 122–128, 2017.

[8] Y. Efendi, "Internet Of Things (Iot) Sistem Pengendalian Lampu Menggunakan Raspberry Pi Berbasis Mobile," *J. Ilm. Ilmu Komput.*, vol. 4, no. 2, pp. 21–27, 2018, doi: 10.35329/jiik.v4i2.41.

[9] R. F. Ramadhan and R. Mukhaiyar, "Penggunaan Database Mysql dengan Interface PhpMyAdmin sebagai Pengontrolan Smarthome Berbasis Raspberry Pi," vol. 1, no. 2, pp. 129–134, 2020.

[10] Nirsal, Rusmala, and Syafriadi, "Desain Dan Implementasi Sistem Pembelajaran Berbasis E-

Learning Pada Sekolah Menengah Pertama Negeri 1 Pakue Tengah," *J. Ilm. d'Computare*, vol. 10, pp. 30–37, 2020, [Online]. Available: http://www.elsevier.com/locate/scp

[11] C. Bestari Gea, K. Juri Damai Lase, M. Syamsudin, P. Studi Informatika, F. Sains dan Komputer, and U. Kristen Immanuel Yogyakarta, "Implementasi Virtual Private Server untuk Mini Hosting," *J. InFact Sains dan Komput.*, vol. 7, no. 02, pp. 5–9, 2023.

[12] B. Robert and E. B. Brown, "Pengembangan Distro Ubuntu untuk Aplikasi Game Center," no. 1, pp. 1–14, 2010.

[13] F. Nabawi and A. B. Susanto, "Perancangan Sistem Keamanan Server Linux Ubuntu 18 . 04 dengan Metode Ufw Firewall , Hardening , Chmod dan Chown pada UNUSIA Jakarta Abstrak Pendahuluan," vol. 7, no. 4, 2022.

[14] A. Aziz and T. Tampati, "Analisis Web Server untuk Pengembangan Hosting Server Institusi : Pembandingan Kinerja Web Server Apache dengan Nginx," vol. 1, no. 2, pp. 12–20, 2015.

[15] M. Meng, S. Steinhardt, and A. Schubert, "Application Programming Interface Documentation : Application Programming Interface Documentation : What Do Software Developers Want ?," no. March, 2019, doi: 10.1177/0047281617721853.

[16] A. B. Warsito, A. Ananda, and D. Triyanjaya, "Penerapan Data JSON Untuk Mendukung Pengembangan Aplikasi Pada Perguruan Tinggi Dengan Teknik Restfull Dan Web Service," *Technomedia J.*, vol. 2, no. 1, pp. 26–36, 2017, doi: 10.33050/tmj.v2i1.313.

[17] S. Kosasi, I. D. Ayu, E. Yuliani, and G. Syarifudin, "Implementasi Arsitektur Model View Controller pada Website Toko Online Implementation of Model View Controller Architecture on Online Store Website," vol. 3, no. 2, pp. 135–150, 2021, doi: 10.30812/bite.v3i2.1566.

[18] D. Purnama Sari and R. Wijanarko, "Implementasi Framework Laravel pada Sistem Informasi Penyewaan Kamera (Studi Kasus di Rumah Kamera Semarang)," *J. Inform. dan Rekayasa Perangkat Lunak*, vol. 2, no. 1, p. 32, 2020, doi: 10.36499/jinrpl.v2i1.3190.

[19] A.-I. A.B., "Implementasi Teknologi Websocket dalam Pengembangan Sistem Berbagi Lokasi Berbasis Web," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 1, no. 9, pp. 950–959, 2017, [Online]. Available: http://j-ptiik.ub.ac.id

[20] C. Asiminidis, G. Kokkonis, and S. Kontogiannis, "Database Systems Performance Evaluation for IoT Applications," *SSRN Electron. J.*, no. November 2019, 2019, doi: 10.2139/ssrn.3360886.

BIODATA

Muhammad Khoiril Wafi Lahir di Kabupaten Demak, 4 Maret 2001. Telah menempuh pendidikan mulai dari SD Negeri Karangrejo 2 selama 6 tahun, SMP Negeri 1 Bonang selama 3 tahun dan SMK Negeri 2 Demak selama 3 tahun. Saat ini penulis sedang menyelesaikan pendidikan S1-Teknik Elektro di Fakultas Teknik Universitas Diponegoro, Konsentrasi Teknologi Informasi angkatan 2019.

Saya menyatakan bahwa segala informasi yang tersedia di makalah ini adalah benar, merupakan hasil karya sendiri, bebas dari plagiat dan semua karya orang lain telah dikutip dengan benar.

Muhammad Khoiril Wafi
NIM. 21060119140133

Telah disetujui untuk diajukan pada Seminar Tugas Akhir
Semarang, 8 September 2023

Pembimbing 1	Pembimbing 2
M. Arfan, S.Kom., M.Eng.	**Imam Santoso, S.T., M.T.**
NIP. 198408172015041002	NIP. 197012031997021001