# Perancangan *Unified IoT Platform* Untuk Pengembangan Sistem IoT berbasis OPC UA

Sujoko Sumaryono[1)] , Dzakwan Silverdi Hasan[2)] , Dani Adhipta[3)]
*Departemen Teknik Elektro dan Teknologi Informasi, Universitas Gadjah Mada, DI Yogyakarta, Indonesia*
[1]sujoko@ugm.ac.id
[2]dzakwansilverdi@mail.ugm.ac.id
[3]dani@ugm.ac.id

***Abstract*— Based on data obtained by the Indonesian Internet of Think (IoT) Association (ASIOTI), in 2022 the number of IoT devices or sensors used in Indonesia will reach 400 million sensors with a market value of around Rp 350 trillion. To bridge sensors and user applications, an IoT platform is needed that can provide interoperability solutions to the data provided by IoT devices. For this reason, a Konnex IoT platform was designed with the aim of helping the development of IoT systems to bridge sensor data and user applications through the Open Platform Communication Unified Architecture (OPC UA) protocol. The OPC UA protocol has the advantage of being Plug and Play so that the system can access data sent by the OPC UA protocol without the need to manually reconfigure or install device drivers. Based on the performance tests that have been carried out, through a load of 1000 virtual users, the system that has been designed is able to support data requests of 698 QPS (queries per second) to 2886 QPS. The system also supports an average response time of under 300 ms. All processed data requests have been served without any failures.**

***Keywords*— *Internet of Things, OPC UA, IoT Platform*.**

**Intisari— Berdasarkan data yang diperoleh oleh Asosiasi IoT Indonesia (ASIOTI), pada 2022 jumlah perangkat atau sensor IoT (*Internet of Think*) yang digunakan di Indonesia mencapai 400 juta sensor dengan nilai pasar sekitar Rp 350 triliun. Untuk menjembatani sensor dan aplikasi pengguna, dibutuhkan suatu *platform* IoT yang dapat menyediakan solusi interoperabilitas terhadap data yang disediakan oleh piranti IoT. Untuk itu, dirancang sebuah *platform* IoT Konnex dengan tujuan membantu pengembangan sistem IoT untuk menjembatani data sensor dan aplikasi pengguna melalui protokol OPC UA (*Open Platform Communication - Unified Architecture* ). Protokol OPC UA memiliki keunggulan berupa *Plug and Play* sehingga sistem dapat mengakses data yang dikirim oleh protokol OPC UA tanpa perlu melakukan konfigurasi ulang atau pemasangan pengandar (*driver*) peranti secara manual. Berdasarkan pengujian performa yang telah dilakukan, melalui beban sebesar 1000 pengguna virtual, sistem yang telah dirancang mampu mendukung *request* data sebesar 698 QPS (*query per second*) hingga 2886 QPS. Sistem juga telah mendukung rata rata *response time* dibawah 300 ms. Seluruh *request* data yang diproses telah dilayani tanpa adanya kegagalan.**

## I. PENDAHULUAN

Sistem *Internet of Things* (IoT) dapat dibangun dengan menggunakan berbagai piranti yang memiliki konfigurasi dan spesifikasi yang berbeda. Namun, untuk mengintegrasikan berbagai piranti yang memiliki spesifikasi berbeda tersebut, diperlukan proses yang rumit dalam menyesuaikan konfigurasinya. *Open Platform Communication Unified Architecture (OPC-UA)* adalah suatu *platform* yang dapat digunakan untuk mengintegrasikan piranti yang memiliki spesifikasi dan konfigurasi yang berbeda dapat direalisasikan. Berbagai piranti IoT dapat dihubungkan melalui suatu *server* OPC-UA, kemudian data yang diproses dapat dibaca dengan menggunakan aplikasi klien OPC-UA.

Pengembangan sistem IoT berbasis *server* OPC-UA akan mempermudah perancangan suatu *Platform* IoT, sehingga pengembang IoT tidak perlu melakukan proses berulang dalam membangun sistem berbasis OPC-UA. Pembangkan *platform* IoT berbasis teknologi OPC-UA, maka pengembang sistem IoT dapat membangun *server* OPC-UA dan menghubungkan dengan piranti IoT melalui antarmuka interaktif tanpa melakukan pemrograman, sehingga alur pengembangan sistem IoT dapat dipersingkat dengan memanfaatkan fungsionalitas yang tersedia pada *platform* IoT yang akan dibangun.

Makalah ini akan menjelaskan mengenai rancangan berdasar spesifikasi *engineering* dari *project* yang akan dihasilkan. Beberapa fitur akan ditambahkan serta menjelaskan bagaimana piranti IoT dapat terhubung dengan server OPC melalui *platform* IoT. Makalah ini memuat penjelasan mengenai dasar teori pendukung dan analisis studi pustaka kunci, juga akan dipaparkan rancangan detail teknis dari solusi yang diusulkan. Metode dan hasil pengujian untuk memastikan kelayakan rancangan solusi dan kesesuaiannya terhadap spesifikasi teknis yang telah ditetapkan dalam rancangan akan dibahas dalam makalah ini.

## II. LANDASAN TEORI

### A. *Internet of Things*

Sistem IoT merupakan sebuah konsep keterhubungan antara suatu mesin mekanik atau digital, objek, hingga makhluk hidup dengan suatu alat berbasis elektronik, sensor, aktuator, dan koneksi internet yang memiliki kemampuan untuk mengumpulkan dan mentransfer data tanpa membutuhkan campur tangan manusia. Definisi tersebut, menjadikan sistem berbasis IoT merupakan komponen penting dalam pembangunan industri 4.0.

Industri 4.0 dicirikan dengan sebuah pergeseran paradigma dari sistem kontrol tersentralisasi menjadi sistem kontrol desentralisasi yang memungkinkan adanya komunikasi antara manusia, mesin, dan sumber daya. IoT memungkinkan sebuah obyek seperti RFID, sensor, aktuator, smartphone, dengan melalui sebuah protokol

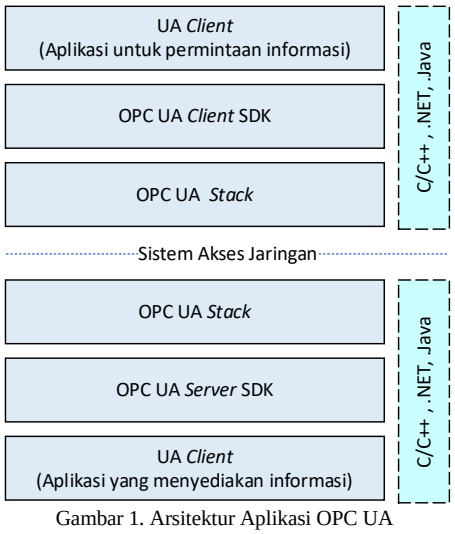
tertentu sehingga dapat berinteraksi dengan sebuah sistem pintar untuk mencapai tujuan tertentu [1].

### *B. Open Platform Communication Unified Architecture (OPC UA)*

*Open Platform Communication* atau OPC merupakan sebuah standar interoperabilitas untuk pertukaran data yang aman dan andal di lingkup otomasi industri. OPC merupakan *platform* yang independen dan memastikan aliran informasi yang lancar di antara perangkat dari beberapa vendor. OPC UA dirilis pada tahun 2008 yang merupakan integrasi antara *platform* individual dari versi klasik OPC yang diantaranya berupa OPC *Data Access* yang mendefinisikan pertukaran data; OPC *Alarms & Events* yang mendefinisikan pesan informasi, *state variable* serta *state management;* dan OPC *Historical Data Access* yang mendefinisikan metode kueri dan analitik yang bisa diaplikasikan untuk data historis atau data berbasis *time-series* [2].

Komponen Fundamental dari OPC UA adalah mekanisme *transport* dan *data modelling*. *Transport* mendefinisikan berbagai mekanisme yang digunakan untuk kasus yang berbeda. Penggunaan protokol TCP biner untuk komunikasi intranet dengan performa tinggi serta sebagai *mapping* ditujukan untuk menerima komunikasi internet standar seperti *Web Services*, XML, dan HTTP. *Data modelling* mendefinisikan aturan atau *building blocks* yang diperlukan untuk mengekspos model informasi melalui OPC UA. Komponen ini juga mendefinisikan *entry points* untuk tempat alamat dan *base types* yang digunakan untuk membuat tipe hirarki [2].

Arsitektur aplikasi OPC UA menggunakan mode komunikasi berbasis *client-server*. Server OPC UA mengekspos dan menyediakan metode ke perangkat dan aplikasi lain, dan klien dapat menerima informasi dari server dan melakukan serangkaian operasi di server. Meski memiliki fungsi yang berbeda, server dan klien OPC UA harus memiliki tiga lapisan perangkat lunak yaitu, specific application function (API), OPC UA Software Development Kit (SDK), dan OPC UA Communication Stack [3]. Gambar 1 memperlihatkan suatu arsitektur aplikasi OPC UA.




Gambar 1. Arsitektur Aplikasi OPC UA

Lapisan aplikasi spesifik (*specific application function*), Klien UA memanipulasi dan mengeksplorasi *address space* yang disediakan oleh server. OPC UA SDK Lapisan *Software Development Kit* (SDK) merupakan lapisan *level* tinggi yang memiliki tugas seperti mengelola koneksi, dan memproses pesan layanan. Pada bagian ini diperlukan implementasi terhadap konsep, model informasi, dan layanan yang berkaitan dengan spesifikasi OPC UA. SDK untuk OPC UA adalah bagian lapisan aplikasi, yang dapat mengurangi beban kerja pengembangan. Bagian ini perlu menerapkan konsep, informasi model, layanan, dan spesifikasi terkait lainnya dari OPC UA. Lapisan OPC UA *Stack* mengimplementasikan berbagai pemetaan *transport* dari OPC UA. Terdapat tiga lapisan utama yang didefinisikan pada OPC UA *Stack*. Di antaranya yaitu, lapisan *encoding*, lapisan keamanan, dan lapisan *transpor*t [4] .

### *C. IoT Platform*

*Platform* IoT merupakan sebuah teknologi *multi-layer* yang menyediakan kumpulan fitur yang siap pakai untuk mempercepat proses pengembangan sistem IoT [5]. Secara umum, arsitektur IoT terdiri atas lima komponen, yang diantaranya berupa *hardware*, *gateway*, pemrosesan data berbasis *cloud*, konektivitas IoT, dan aplikasi antarmuka pengguna. Tanpa adanya *platform* IoT, akan terdapat antara *hardware* dan lapisan aplikasi yang akan menjadi sangat rumit untuk membuat kedua lapisan ini dapat bekerja sama.

Berdasarkan peran dan fitur yang dilayani, *platform* IoT terdiri atas tiga jenis yaitu *Gateway Platform*, *Middleware Platform*, dan *Cloud IoT Platform*. *Gateway* merupakan sebuah piranti keras atau program *software* yang melayani koneksi antara piranti IoT dan layanan *Cloud*. *Middleware Platform* berperan sebagai jembatan penghubung antara komponen yang heterogen serta menyediakan layanan seperti manajemen sumber daya, manajemen data, ekstraksi informasi, hingga pengembangan privasi dan keamanan. *Cloud IoT Platform* merupakan *platform* IoT yang berjalan diatas infrastruktur komputasi awan serta memiliki kelebihan seperti kemampuan untuk skalabilitas, layanan komputasi serta penyimpanan berbasis *cloud*, hingga keandalan infrastruktur yang dikembangkan dan dirawat oleh organisasi teknologi besar seperti Google, Amazon, dan Microsoft [5].

## III. METODOLOGI

### *A. Unified IoT Platform*

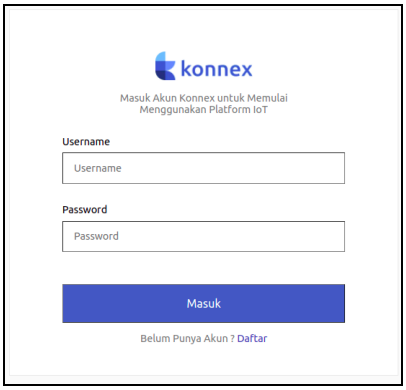
Untuk memperoleh produk aplikasi yang memiliki keandalan dan ketahanan sistem, maka perlu dilakukan uji performa pada aplikasi tersebut. Hasil pengembangan *platform* Konnex ini, diharapkan dapat memberikan layanan dengan baik, apabila diberikan beban hingga 1000 pengguna yang mengakses *platform* Konnex secara bersamaan, maka sistem *platform* ini harus dapat mendukung metrik performa sebagai berikut;

a. minimum 300 *request* per detik untuk fungsionalitas *login*, fungsionalitas *read data*, serta fungsionalitas perolehan klien dan *server* OPC UA,

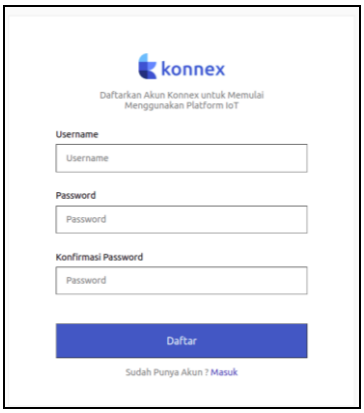
b. rata-rata nilai *response time* kurang dari 300 ms untuk semua *request*,
c. maksimum 1% *request* yang gagal.

*B. Desain Antarmuka*

Desain antarmuka pengguna pada *platform* luaran ini dirancang sedemikian rupa untuk memudahkan penggunaan *platform* IoT untuk pengembangan sistem berbasis OPC UA. Gambar 2 dan Gambar 3 berikut menunjukkan antarmuka *login* dan *register* pengguna.

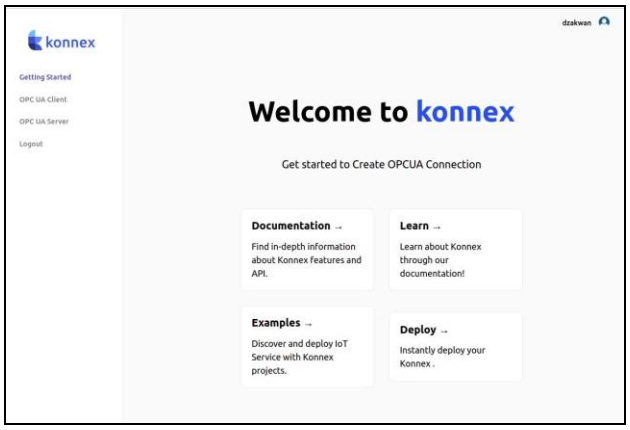



Gambar 2. Antarmuka Login Pengguna



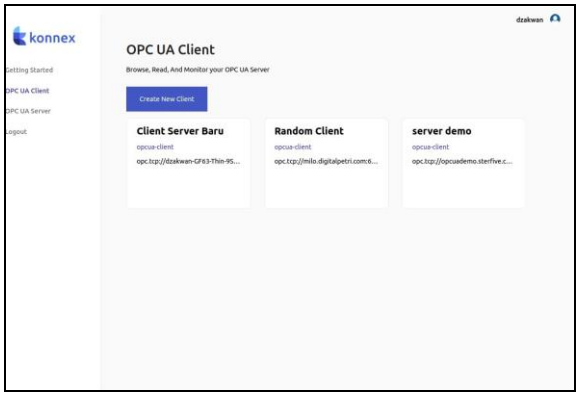

Gambar 3. Antarmuka laman pendaftaran pengguna

Seperti yang ditunjukkan pada Gambar 2 dan 3, Kedua halaman ini berupa tampilan *form* sederhana sebagai masukan data pengguna seperti *username*, dan *password* yang kemudian dikirimkan ke sistem *backend* untuk otentikasi pengguna. Apabila otentikasi sukses, pengguna akan diarahkan ke antarmuka halaman utama *platform* Konnex yang ditunjukkan pada Gambar 4.



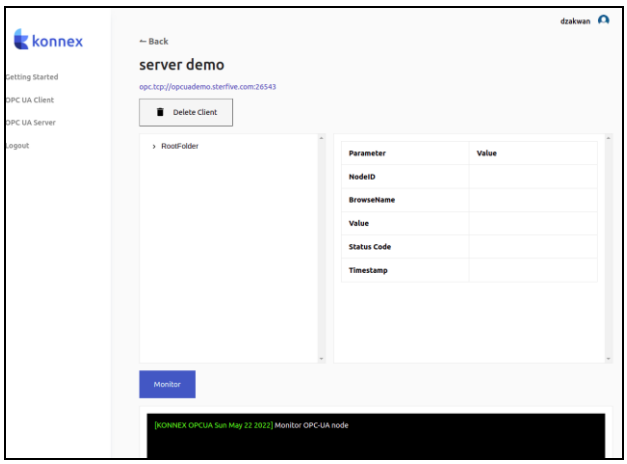

Gambar 4. Antarmuka Halaman Utama

Halaman Utama merupakan halaman yang pertama kali dimasuki pengguna setelah sukses melakukan otentikasi pada halaman login. Halaman ini terdiri atas menu panduan penggunaan dalam membuat sistem IoT pada *platform* Konnex. Pada *platform* Konnex ini selain menu utama, terdapat dua menu lain yaitu laman klien OPC UAdan laman server OPC UA.



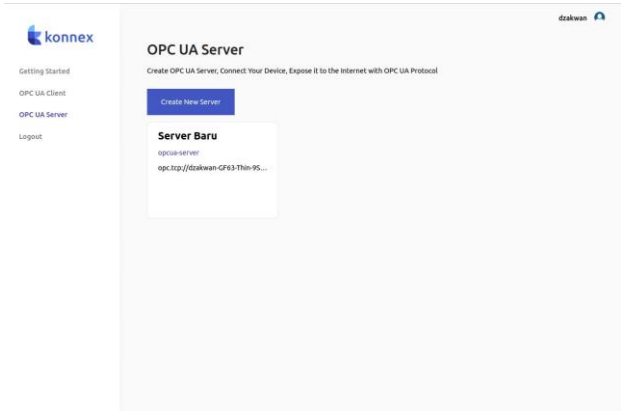

Gambar 5. Antarmuka Halaman Utama Klien OPC UA

Laman Klien OPC UA pada Gambar 5 menampilkan daftar klien OPC UA yang terhubung dengan suatu *server* OPC UA. Pada laman ini juga terdapat tombol pembuatan klien OPC UA yang dapat mengarahkan pengguna ke laman pembuatan klien.



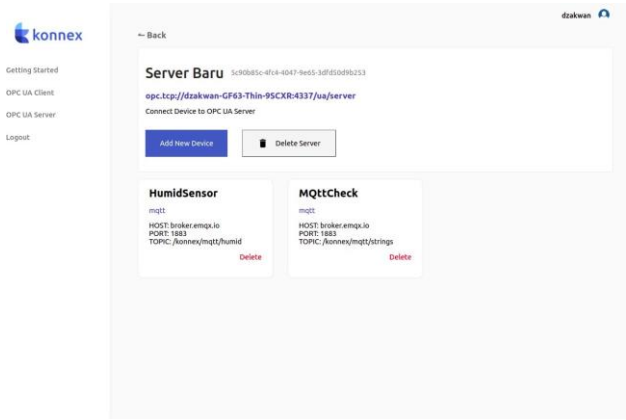

Gambar 6. Antarmuka Halaman Monitor Data OPC UA

Laman Monitor data yang ditunjukkan pada Gambar 6 menampilkan daftar *node* yang tersedia pada suatu *server* OPC UA serta menunjukkan data yang ditunjukkan pada *node* tersebut seperti nilai *value*, *nodeid*, hingga *status code*. Pada laman ini juga dapat *button* monitor untuk memperoleh *streaming* data dari suatu piranti IoT yang terhubung pada *server* OPC UA yang terhubung.




Gambar 7. Antarmuka Halaman Utama *Server* OPC UA

Laman server OPC UA pada Gambar 7 menampilkan daftar *server* yang dibuat melalui *platform* Konnex. Pada laman ini terdapat tombol `Create New Server` yang akan mengarahkan pengguna pada laman pembuatan *server* OPC UA.

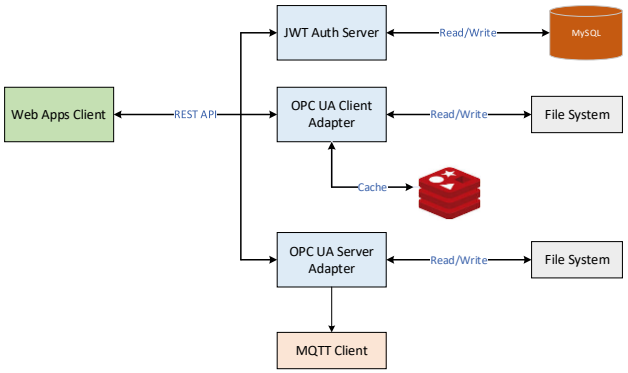



Gambar 8. Antarmuka Dashboard Server OPC UA

Laman *dashboard server* OPC UA pada Gambar 8 menampilkan detail dari setiap server OPC UA yang telah dibuat. Pada halaman ini terdapat daftar kumpulan piranti IoT yang terhubung pada *server* OPC UA. Pada laman ini terdapat tombol fungsionalitas yang akan mengarahkan pengguna ke laman penambahan piranti IoT pada server OPC UA tersebut.

## *C.* Arsitektur Sistem

Pola arsitektur yang digunakan pada perancangan perangkat lunak *platform* IoT ini menggunakan pendekatan *microservices*. *Microservices* merupakan sebuah pendekatan arsitektur yang terdiri atas layanan-layanan kecil dan otonom yang dapat bekerja sama [6]. Pada *platform* IoT ini terdapat tiga layanan yang bekerja, yaitu, layanan otentikasi, layanan adaptor OPC UA *server*, dan layanan adaptor OPC UA *client*. Rancangan arsitektur perangkat lunak *platform* Konnex dapat dilihat pada Gambar 9 berikut.




Gambar 9. Arsitektur *Platform* Konnex

Layanan otentikasi bertugas dalam mengatur sistem otentikasi dan manajemen pengguna. Layanan ini memiliki hubungan *read/write* dengan server basis data MySQL. Layanan adaptor klien OPC UA bertugas sebagai monitor dan penghubungan komunikasi data dengan server OPC UA. Layanan ini menggunakan sistem file sebagai penyimpanan konfigurasi klien OPC UA. Layanan adaptor server OPC UA bertugas sebagai pembuatan server OPC UA serta menyediakan koneksi protokol IoT.

## *D. Application Programming Interface*

Pada pengembangan *platform* Konnex, rancangan API dibagi menjadi tiga bagian yang masing masing merepresentasikan API dari layanan autentikasi pengguna, adaptor klien OPC UA, hingga adaptor *server* OPC UA. Kontrak API untuk layanan autentikasi dapat dilihat pada Tabel 1 berikut.

TABEL 1. API OTENTIKASI PENGGUNA

| Fungsi | Alamat |
|---|---|
| Register | [POST] http://<HOST>:<PORT>/register |
| Authorize | [POST] http://<HOST>:<PORT>/Authorize |

Layanan otentikasi pengguna menyediakan dua API yaitu *register user* dan *authorize user*. API *register user* digunakan untuk mendaftarkan pengguna ke dalam *platform* Konnex dengan mengirim data *e-mail* dan *password* menuju alamat API register. API *authorize user* digunakan sebagai fungsionalitas otentikasi dan otorisasi pengguna dalam menggunakan *platform* Konnex. API ini memanfaatkan teknologi JSON *web token* sebagai enkripsi data pengguna dan verifikasi sistem otentikasi.

TABEL 2. API ADAPTOR KLIEN OPC UA

| Fungsi | Alamat |
|---|---|
| Save Client | [POST] http://<HOST>:<PORT>/register |
| Get Client | [POST] http://<HOST>:<PORT>/Authorize |
| Browse Node | [GET]http://<HOST>:<PORT>/client/browse?id=<client-id>&node=<node-id> |
| Read Node | [GET]http://<HOST>:<PORT>/client/read?id=<clientID>&node=<nodeID> |
| Delete Client | [DELETE]http://<HOST>:<PORT>/client?id=<client ID> |

Layanan klien OPC UA memiliki fungsionalitas yang berhubungan dengan pembuatan klien OPC UA, komunikasi data server OPC UA, hingga monitor data server OPC UA. Berdasarkan pada Tabel 2, layanan adaptor klein OPC UA menyediakan lima API yang terdiri atas API *save client, browse client, get client, read node, delete client.*

API *save client* berfungsi sebagai fungsionalitas pembuatan klein OPC UA dengan mengirimkan data berupa nama klien beserta alamat OPC UA *server* yang hendak dihubungi. API *browse client* berfungsi sebagai fungsionalitas untuk mengeksplorasi setiap node yang berada pada server OPC UA yang telah terhubung. API *get client* berfungsi untuk memperoleh daftar klien OPC UA yang telah dibuat oleh pengguna. API *read node* berfungsi untuk melihat deskripsi dan nilai yang tertera pada suatu node pada *server* OPC UA. API *delete client* berfungsi untuk menghapus klien OPC UA yang dibuat oleh pengguna beserta koneksinya dengan *server* OPC UA yang terhubung.

TABEL 3. API ADAPTOR SERVER OPC UA

| Fungsi | Alamat |
|---|---|
| *Get Server List* | [GET]http://<HOST>:<PORT>/uaserver |
| *Get Server* | [GET]http://<HOST>:<PORT>/uaserver?id=<server ID> |
| *Create Server* | [POST]http://<HOST>:<PORT>/uaserver |
| *Add MQTT Device* | [POST]http://<HOST>:<PORT>/uaserver/variable?type=mqtt |
| *Delete Device* | [DELETE]http://<HOST>:<PORT>/uaserver/variable?id=<serverID>&node=<nodeID> |
| *Delete Server* | [DELETE]http://<HOST>:<PORT>/uaserver?id=<ser verID> |

Layanan *server* OPC UA memiliki fungsionalitas yang berhubungan dengan pembuatan *server* OPC UA, dan pembuatan koneksi dengan piranti IoT. Berdasarkan pada daftar API yang telah ditunjukkan pada Tabel 3, layanan daptor *server* OPC UA memiliki enam API yang terdiri atas API *get server list, get server, create server, add MQTT device, delete device,* dan *delete server*.

API *get server list* memiliki fungsi untuk memperoleh daftar *server* OPC UA yang telah dibuat oleh pengguna. API *get server* memiliki fungsi untuk memperoleh server spesifik sesuai dengan parameter *id server* yang dimasukkan. API *create server* memiliki fungsi untuk membuat server OPC UA dengan memasukkan input *request body* berupa nama *server* dan nomor *port server* yang akan dibuat. API *add MQTT device* berfungsi untuk menyambungkan *server* OPC UA dengan piranti IoT melalui protokol MQTT. API *delete device* berfungsi untuk memutuskan dan menghapus koneksi piranti IoT yang terhubung ke *server* OPC UA. API *delete server* berfungsi untuk menghentikan jalannya *server* OPC UA beserta menghapus obyek *server* OPC UA yang telah dibuat oleh pengguna.

## IV. HASIL DAN PEMBAHASAN

Pengujian performa produk aplikasi dilakukan dengan menggunakan metode *load testing* dengan tujuan sebagai berikut:

1. Menilai kinerja sistem yang ada pada saat dikenai beban tipikal hingga beban puncak,
2. untuk memastikan sistem memenuhi spesifikasi performa dalam berbagai kondisi beban.

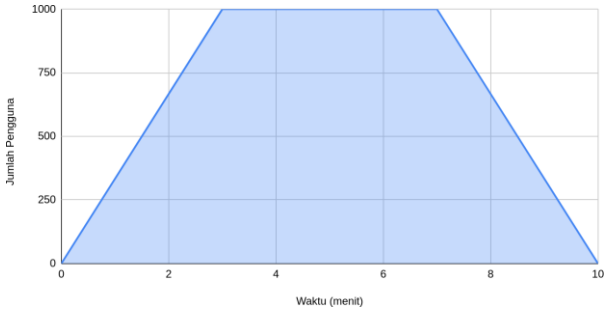
Perangkat yang digunakan pada pengujian ini yaitu K6. K6 merupakan alat *load testing* sumber terbuka yang dikembangkan oleh Grafana. Dengan menggunakan K6, dapat dilakukan pengujian performa dan kehandalan sistem serta mampu memperoleh data berupa kecacatan performa serta isu terkait masalah pada sistem [7].

Dalam pengujian performa, perlu diperhatikan metrik metrik pengujian yang akan dijadikan acuan penilaian. K6 sudah memiliki metrik yang otomatis terkalkulasi melalui proses pengujian. Berdasarkan dokumentasi yang tersedia pada situs dokumentasi K6, metrik pengujian yang akan dilihat dapat ditunjukkan pada Tabel 4.

TABEL 4. METRIK PENGUJIAN PERFORMA

| Metrik | Satuan | Keterangan |
|---|---|---|
| *vus* | *user* | Jumlah pengguna virtual |
| *http_req_duration (response time)* | ms (*milisecon*) | Waktu respon sistem |
| *iteration* | *transaction* | Jumlah Iterasi pengujian |
| *iteration speed* | *transaction per second* (TPS) | Jumlah Iterasi pengujian dalam satu detik |
| *http_reqs* | *query* | Jumlah *request* yang diajukan ke sistem |
| *http_reqs speed* | *query per secon* (QPS) | Jumlah *request* yang diajukan ke sistem dalam satu detik |
| *Failed Request* | persen | Jumlah *request* yang gagal diproses pada sistem |

K6 mengenal konsep *virtual users* (VU), yaitu sebuah pengguna virtual yang dapat meniru skenario penggunaan sistem pada produk aplikasi. Dalam kasus pengujian ini, digunakan VU sebanyak 1000 pengguna, yang akan berjalan secara bersamaan dengan skenario yang ditunjukkan pada Gambar 9 sebagai berikut.




Gambar 9 Skenario Pengguna Virtual

Berdasarkan grafik pada Gambar 9, selama tiga menit pertama, pengguna ditingkatkan dari nol hingga 1000 pengguna, kemudian jumlah pengguna dipertahankan pada 1000 pengguna selama empat menit, dan selanjutnya jumlah pengguna diturunkan hingga nol selama tiga menit terakhir. Sehingga, diperoleh durasi pengujian performa selama 10 menit. Perancangan skenario yang seperti ini, ditujukan untuk meniru skenario pengguna pada kasus nyata.

Pada pengujian performa akan dilakukan pengujian pada lima fungsionalitas penting. Fungsionalitas sistem yang akan dilakukan pengujian diantaranya yaitu fungsionalitas: *login user*, fungsionalitas *get server* dan *get client*, fungsionalitas *read node* (pembacaan informasi *node* OPC UA), hingga fungsionalitas *browse node* (eksplorasi *address space* pada OPC UA). Hasil pengujian pada fungsionalitas tersebut ditunjukkan pada Tabel 5 berikut.

TABEL 5. HASIL PENGUJIAN PERFORMA

| Fungsionalitas | Metrik | Hasil |
|---|---|---|
| *Login* | *Response time* | 2,1 ms |
| | *Query per Sekon* | 698,278 QPS |
| | *Failed Request* | 0 % |
| Get Server | *Response time* | 638,65 ms |
| | *Query per Sekon* | 721,493 QPS |
| | *Failed Request* | 0 % |
| Get Client | *Response time* | 1.47 ms |
| | *Query per Sekon* | 699,171 |
| | *Failed Request* | 0 % |
| Read Node | *Response time* | 2,08 ms |
| | *Query per Sekon* | 2886,31 QPS |
| | *Failed Request* | 0 % |
| Browse Node | *Response time* | 2.17 ms |
| | *Query per Sekon* | 2777,19 |
| | *Failed Request* | 0 % |

Berdasarkan hasil pengujian performa yang telah dilakukan pada lima fungsionalitas sistem, system yang telah dirancang sudah sesuai dengan spesifikasi yang diinginkan. Namun, perlu adanya perbaikan pada sistem *get server* untuk mengurangi nilai *response time* menjadi dibawah 300 ms. Dengan adanya pengujian performa ini, *platform* Konnex sudah dapat dipastikan dapat berjalan untuk 1000 pengguna sekaligus tanpa ada kendala apapun.

## V. KESIMPULAN

Pengembangan *platform* Konnex ditujukan untuk memudahkan pembuatan sistem IoT berbasis OPC UA tanpa memerlukan pemrograman. *Platform* Konnex dirancang dengan memiliki fungsionalitas penting yang diantaranya pembuatan server OPC UA, pembuatan klien OPC UA, penghubungan piranti IoT dengan server OPC UA, hingga pembacaan dan monitor data informasi melalui klien OPC UA.

Perancangan sistem yang telah di implementasi pada *platform* Konnex juga memiliki performa yang baik. Dengan pengujian beban sebanyak 1000 pengguna virtual, *platform* Konnex mampu memenuhi standar spesifikasi performa yang diinginkan yaitu, sistem mampu mendukung transaksi lebih dari 300 QPS (*query per second*), sistem juga mampu mendukung rata rata *response-time* di bawah 300 ms, dan seluruh *request* yang diajukan pada sistem tidak mengalami kegagalan (tidak terdeteksi adanya *Failed Request*).

## REFERENSI

[1] Hermann, H., “Design Principle for Industries 4.0 Scenarios,” 2016, IEEE Explore, ISSN: 1530-1605.
[2] Wolfgang Mahnke, Stefan-Helmut Leitner, and Matthias Damm, *OPC unified architecture*. Berlin ; New York: Springer-Verlag, 2009.
[3] Hui Ren, Yang Liu, Huiqin Wang, “Research on Communication Method of OPC UA Client Based on ARM,” IEEE Explore, 2019.
[4] Mahnke, W. Leitner, S. H., and Damm, M., OPC Unified Architecture. Berlin ; New York: Springer-Verlag, 2009.
[5] Yajuan Guan, Carna Zivkovic, and Christoph Grimm, *IoT platforms, use cases, privacy, and business models : with hands-on examples based on the VICINITY platform*. Cham, Switzerland: Springer, 2021.
[6] Newman, S., *BUILDING MICROSERVICES : designing fine-grained systems.* oreilly, 2018.
[7] Grafana Labs, “k6 Documentation,” *k6.io*. https://k6.io/docs/