Rancang Bangun Platform sebagai Sistem Pemantauan untuk Pemodelan Pembangkit Listrik Virtual

Muhammad Djati Pradana
1606829680
Electrical Engineering Department
Computer Engineering, University of Indonesia
Depok, Indonesia
muhammad.djati@ui.ac.id

***Abstract* — Perkembangan teknologi merubah sistem penyediaan listrik tersentralisasi menjadi desentralisasi dengan menggunakan pembangkit listrik virtual. Konsep pembangkit listrik virtual menggunakan pembangkit listrik dengan energi terbarukan yang terdistribusi sehingga memungkinkan klien dapat berperan menjadi prosumer yang terhubung dengan jaringan listrik. Skripsi ini bertujuan untuk merancang platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual. Sistem pemantauan memungkinkan klien dapat memantau aktivitas pembangkit dan beban meliputi daya yang dihasilkan dari setiap pembangkit, kapasitas baterai, daya pada beban serta total energi yang didapatkan dari akumulasi daya per hari pada beban dan pembangkit. Pemodelan ini menggunakan 3 klien yang terdiri dari 2 klien memiliki pembangkit dan beban serta 1 klien memiliki baterai. Pengujian sistem dilakukan terhadap penggunaan *node editor* dan *usability testing*. Hasil dari pengujian menunjukan bahwa penggunaan *node editor* dapat mendukung konfigurasi pada platform yang telah dimodelkan serta penilaian terhadap platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual oleh responden dapat dikategorikan cukup baik dengan *Mean of Survey* (MoS) berdasarkan responden memahami pembangkit listrik virtual yaitu 4,1 dalam skala 5 sedangkan untuk responden tidak memahami pembangkit lsitrik virtual yaitu 3,93 dalam skala 5.**

***Keywords* — Energi listrik, Pembangkit listrik virtual, Pemantauan, Pemodelan.**

I. PENDAHULUAN

Dalam memberikan pasokan energi listrik yang sesuai permintaan konsumen (optimal) maka diperlukan waktu yang tidak sedikit untuk membangun suatu pembangkit tenaga listrik. Pembangunan sistem tenaga listrik dengan skala besar sering terkendala besarnya investasi dan jangka waktu pembangunan yang lama pada pusat-pusat tenaga listrik dibandingkan pembangunan industri yang lain maka perlu diusahakan agar dapat memenuhi kebutuhan tenaga listrik tepat pada waktunya.

Sampai saat ini sistem pengelolaan energi listrik masih tersentralisasi yang berpusat pada PLN dengan memberikan pasokan listrik dari pembangkit konvensional yang diteruskan ke rumah-rumah sehingga penyediaan listrik tidak optimal untuk memenuhi permintaan konsumen. Pembangkit listrik yang dimiliki oleh PLN secara umum menggunakan energi yang termasuk tidak terbaharui seperti batubara dan bahan bakar fosil lainnya. Untuk dapat memenuhi kebutuhan energi yang terus meningkat maka diperlukan pembangkit tenaga listrik dengan memanfaatkan energi terbarukan (*renewable energy*) seperti cahaya matahari, angin, air dan hidrogen.

Perkembangan teknologi menghasilkan sebuah konsep penyediaan listrik desentralisasi dengan menggunakan pembangkit listrik virtual. Dengan konsep pembangkit listrik virtual ini, memungkinkan pemilik rumah dapat mengontrol penggunaan energi listrik dengan menjadi produsen dan konsumen energi listrik (prosumer) secara bersamaan yang dapat terhubung dengan jaringan listrik (*grid*). Selain itu, juga dapat membantu PLN dalam mengatasi pemadaman bergilir karena ada gangguan distribusi serta mengurangi beban puncak terhadap kebutuhan energi listrik (*peak load*) baik di rumah tangga maupun industri.

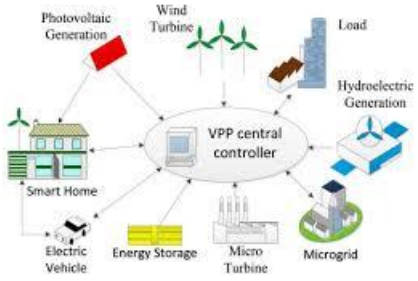
Konsep pembangkit listrik virtual membutuhkan sistem pemantauan untuk dapat mengetahui aktivitas pembangkit dalam produksi energi serta aktivitas beban pada jaringan listrik sesuai dengan algoritma yang telah didefinisikan sehingga diperlukan platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual. Klien dapat mengetahui energi listrik yang dihasilkan dari setiap pembangkit, daya yang digunakan oleh beban, kapasitas baterai serta total energi yang dihasilkan dari akumulasi daya per hari pada beban dan pembangkit.

II. LANDASAN TEORI

A. Pembangkit Listrik Virtual

Pembangkit Listrik Virtual merupakan sebuah konsep yang digunakan untuk manajemen energi dengan mengagregasi energi yang dihasilkan dari pembangkit energi terdistribusi yang tersebar di berbagai titik berbeda dan terhubung dengan jaringan pusat pengontrol yang terdiri atas instalasi skala kecil dan menengah baik yang berfungsi untuk mengonsumsi ataupun

memproduksi listrik [1]. Untuk lebih memperjelas mengenai pembangkit listrik virtual dapat dilihat pada Gambar 1.

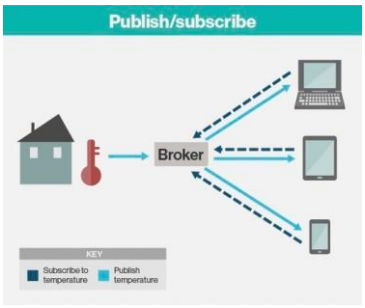


Gambar 1. Skema Umum Sistem Pembangkit Listrik Virtual [2]

Hal penting dari pembangkit listrik virtual adalah adanya pusat kendali yang akan mengatur pembangkit dengan berbagai macam sumber energi yang terlibat dalam pembangkit listrik virtual [3]. Pusat kendali ini akan mengatur pengoperasian dari pembangkit seperti kapan waktu yang tepat untuk melakukan pengisian baterai, berapa daya yang dihasilkan per hari, berapa beban yang digunakan per hari, kapan energi listrik yang dihasilkan dari pembangkit akan digunakan, kapan waktu yang tepat untuk menyalankan barang-barang elektronik berdasarkan penjadwalan otomatis dan lain-lain.

B. MQTT

MQTT (*Message Queuing Telemetry Transport*) merupakan protokol komunikasi *publish* dan *subscribe* berdasarkan topik yang sangat sederhana dan ringan yang hanya mengirimkan paket dengan ukuran yang kecil sehingga tidak membebani *resource* jaringan internet serta didesain untuk alat yang memiliki kemampuan terbatas dengan *bandwidth* yang rendah dan latensi yang tinggi atau jaringan yang kurang dapat diandalkan [4]. Dalam mengetahui konsep dasar dari MQTT maka dapat dilihat pada Gambar 2.

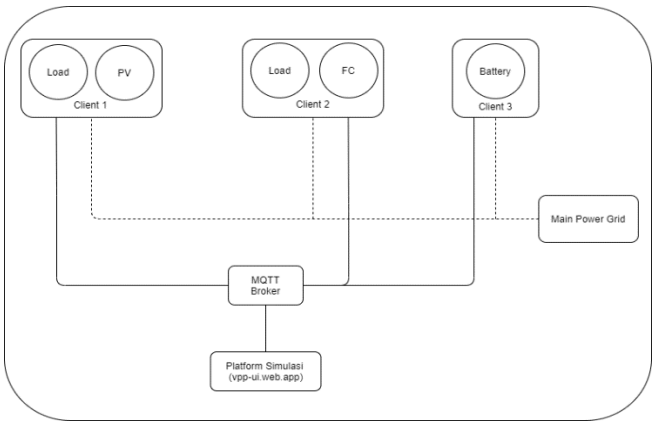


Gambar 2. Konsep Dasar MQTT [5]

Berdasarkan Gambar 2, terdapat dua bagian utama pada konsep dasar MQTT yaitu MQTT *Client* dan MQTT Broker. MQTT Broker adalah sebuah bagian yang dijadikan sebagai server atau *cloud* dengan program berjalan yang berfungsi untuk menerima pesan dari *client* dan meneruskan pesan antara *client*. Sementara *Client* adalah bagian yang melakukan pertukaran data. Pada protokol MQTT tidak membutuhkan suatu alamat pada setiap *client*. Fungsi dari alamat ini digantikan oleh "*Topic*". Sehingga setiap *client* yang akan melakukan pengiriman informasi kepada *client* lain dapat dilakukan dengan mendaftarkan diri pada *topic* tersebut. Pengiriman dan penerimaan informasi antara MQTT *Client* dan MQTT Broker dilakukan dengan menggunakan skema *publish* dan *subscribe* [6]. *Publish* adalah konsep yang menjadikan *client* akan mengirimkan informasi. Informasi tersebut akan dikirimkan dengan melabeli infromasi tertentu dengan suatu *topic*. *Subscribe* adalah konsep yang menjadikan *client* akan menerima informasi sesuai dengan *topic* yang diminta oleh *client* tersebut.

III. PERANCANGAN SISTEM

A. Pemodelan Konsep Pembangkit Listrik Virtual

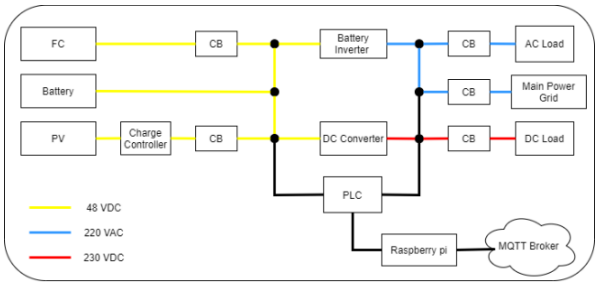


Gambar 3. Pemodelan Konsep Pembangkit Listrik Virtual

Pada Gambar 3, memperlihatkan pemodelan konsep pembangkit listrik virtual yang akan saling menghubungkan antar klien pada *main power grid*. Setiap klien memungkinkan dapat bertindak sebagai prosumer yang dapat menghasilkan energi dari pembangkit listrik yang dimiliki dan memakai energi listrik, namun juga dapat berperan sebagai konsumen yang hanya memakai energi listrik. Dalam pemodelan konsep pembangkit listrik virtual yang digunakan untuk penyelesaian skripsi ini menggunakan 3 klien yang terdiri dari 1 klien yang memiliki baterai (ESS) sebagai tempat penyimpanan energi serta 2 klien yang bertindak sebagai prosumer dengan masing-masing prosumer memiliki pembangkit listrik yaitu *photovoltaic* dan *fuel cell* dan juga beban tersendiri.

Komunikasi dan pertukaran informasi antar klien pada *main power grid* dapat melalui beberapa protokol, namun dalam pemodelan ini menggunakan protokol MQTT. Klien akan mengirimkan beberapa data pada MQTT Broker seperti daya yang dihasilkan oleh pembangkit listrik, daya yang digunakan oleh beban, kapasitas baterai dalam persentase dan daya maksimal pada baterai ketika *discharge* ke *grid* kemudian dapat ditampilkan pada platform yang telah dirancang (vpp-ui.web.app).

Salah satu klien merupakan *Sofwan House* TREC atau biasanya disebut *container* yang terdapat di Fakultas Teknik, Universitas Indonesia. Sistem yang ada di *container* merupakan sistem yang sudah berjalan dengan data-data yang sudah ditentukan.

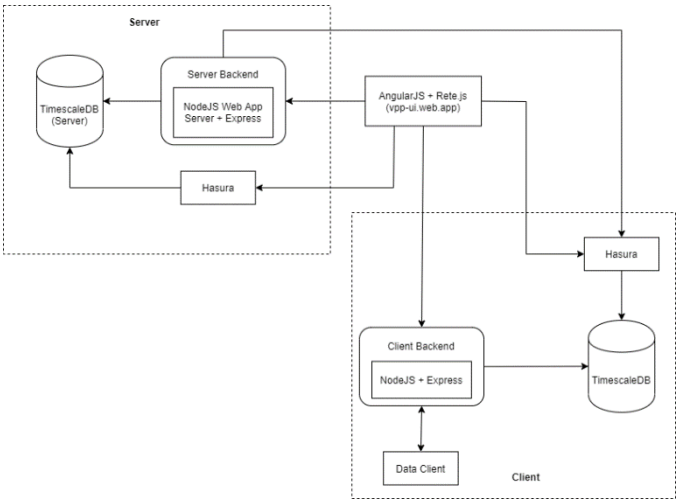


Gambar 4. Perancangan Sistem pada *Sofwan House* TREC FTUI

Berdasarkan Gambar 4, memperlihatkan perancangan sistem pada *container*. Dalam sistem di *container* terdapat pembangkit listrik seperti *photovoltaic* dan *fuel cell* serta terdapat baterai yang digunakan untuk menyimpan energi listrik. Sistem pada *container* menggunakan PLC sebagai sistem kendali yang akan terhubung dengan raspberry pi melalui protokol modbus TCP dan raspberry pi akan terhubung dengan internet untuk mengirimkan data ke broker melalui protokol MQTT. Namun pada sistem yang berjalan saat ini, pembangkit yang dapat dipantau hanya *photovoltaic* sedangkan *fuel cell* untuk saat ini belum dapat dipantau oleh sistem dikarenakan belum ada sensor yang terpasang pada *fuel cell* sehingga belum dapat diotomatisasi serta baterai tidak dapat dipantau dengan baik dikarenakan terdapat sensor yang rusak. Selain *photovoltaic*, terdapat beberapa perangkat yang dapat dipantau seperti *battery inverter*, *DC converter* dan *circuit breaker* pada pembangkit dan beban. *Photovoltaic*, *battery inverter* dan *DC converter* memiliki parameter yang dapat dicatat yaitu tegangan (volt), arus (ampere), daya (watt), energi (kWh) sedangkan pada *circuit breaker* memiliki data yang dapat dicatat yaitu status atau keadaan dari *circuit breaker* tersebut.

B. Pengembangan Platform untuk Pemodelan Pembangkit Listrik Virtual

Platform ini digunakan untuk memodelkan konsep pembangkit listrik virtual serperti melakukan pemantauan, kendali dan juga manajemen energi berdasarkan logika yang sudah dibentuk yaitu mengagregasi daya yang dihasilkan oleh seluruh jenis pembangkit serta daya yang digunakan oleh beban sehingga daya dari pembangkit dan beban yang telah diagregasi akan dibandingkan untuk mengetahui keputusan yang dibuat dalam menentukan pembangkit mana yang akan aktif serta menentukan apakah ekspor ke *grid* atau impor dari *grid*. Klien pada pemodelan ini memiliki kemampuan untuk menghasilkan energi dari pembangkit yang dimiliki, mengonsumsi energi listrik, menyimpan energi mengunakan baterai. Platform ini memungkinkan pengguna dapat memantau aktivitas pembangkit dalam memproduksi energi dan total energi yang dihasilkan dari akumulasi daya per hari pada beban dan pembangkit untuk masing-masing klien.



Gambar 5. Arsitektur Platform untuk Pemodelan Pembangkit Listrik Virtual

Berdasarkan Gambar 5, terdapat tiga bagian utama dalam platform ini yaitu server, website dan klien. Pada server memiliki *backend* yang menyediakan layanan *web service* menggunakan Node.js dan Express sehingga dapat berkomunikasi atau terhubung menggunakan RESTFUL API. Server dalam platform ini bertugas untuk menjalankan *request* yang diberikan oleh website vpp-ui.web.app seperti autentikasi pada *login*, mengelola akun pengguna, menambah klien, mengubah atau menghapus data klien serta membuat keputusan untuk menentukan pembangkit yang akan aktif berdasarkan logika yang didapatkan dari agregasi terhadap total daya pada pembangkit dan beban dari setiap klien. Data yang di-*request* oleh pengguna melalui website akan diterima oleh server dan data tersebut akan disimpan dalam database menggunakan TimescaleDB. Data yang disimpan pada server meliputi data akun pengguna (username, password) baik admin ataupun klien, data klien (nama, ID, lokasi, alamat URL *client backend*, URL layanan Hasura) serta data agregasi daya pada pembangkit dan beban.

Pada platform ini, website dibentuk dengan menggunakan AngularJS yang di-*hosting* menggunakan layanan pada Firebase dan bertindak sebagai *frontend* atau antarmuka yang menghubungkan server dan klien serta untuk menampilkan data dari setiap klien agar lebih menarik dengan tabel dan grafik, mengatur *interfacing* antara platform dengan perangkat seperti pembangkit dan baterai dengan menggunakan *node editor* pada Rete.js sebagai *virtual programming* serta melakukan konfigurasi sesuai dengan keinginan pengguna pada platform ini.

Klien dalam platform ini memiliki kemampuan untuk menghasilkan energi listrik melalui pembangkit yang terdapat pada masing-masing klien, menggunakan energi listrik dan menyediakan baterai sebagai alat penyimpanan energi. Setiap klien juga memungkinkan terdapat beberapa perangkat selain pembangkit seperti inverter dan konveter. Pada setiap klien akan memiliki *backend* yang dikembangkan menggunakan Node.js dan Express sehingga dapat berkomunikasi dengan *frontend* melalui RESTFUL API. Selain itu, setiap klien juga

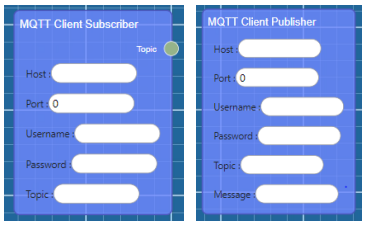
dilengkapi dengan *database* untuk menyimpan data dari masing-masing klien. *Client backend* berfungsi untuk menjalankan fungsi sesuai dengan konfigurasi *node editor* seperti mengambil data dari broker menggunakan protokol MQTT, melakukan kalkulasi terhadap nilai yang didapatkan dari *node* MQTT, menyimpan data yang telah diproses ke dalam TimescaleDB, melakukan akumulasi daya dari pembangkit dan beban selama sehari untuk mendapatkan nilai total energi yang digunakan per hari. Data yang disimpan pada klien meliputi data hasil pemantauan terhadap pembangkit, baterai dan beban pada klien akan disimpan pada TimescaleDB. Pada klien, penggunaan TimescaleDB dikarenakan mendukung penyimpanan rekaman data berdasarkan *timeseries*. Dalam penggunaan TimescaleDB perlu membuat *hypertable* yang secara otomatis akan membentuk tabel *chunk* (*automatic chunking*) sehingga dapat melakukan partisi data sesuai dengan interval yang ditentukan pada tabel *chunk*. Untuk manajemen data pada klien perlu dilakukan rotasi data dengan menghapus data lama (lebih lama dari 7 hari). Penghapusan data lama tersebut dilakukan dengan menghapus atau men-*drop* tabel *chunk* secara langsung berdasarkan interval tabel *chunk* pada TimescaleDB. Dalam pengembangan *node editor* untuk mendukung konfigurasi pada setiap klien maka skema *node* yang telah dibuat pada aplikasi web (vpp-ui.web.app) akan di kirimkan dalam bentuk JSON ke *client backend*. Untuk mengirim data berbentuk JSON ke *client backend* dapat dilakukan dengan menggunakan salah satu fitur yang disediakan oleh website yaitu dengan men-*deploy* konfigurasi tersebut yang secara otomatis akan menyimpan konfigurasi *node* tersebut pada *database* di server. Penyimpanan konfigurasi *node* tersebut dilakukan agar konfigurasi tidak hilang ketika pengguna sudah selesai melakukan konfigurasi.

Dalam melakukan pengaksesan terhadap *database* menggunakan layanan Hasura yang di-*deploy* menggunakan Heroku. Penggunaan fungsi *subscription* pada layanan Hasura berfungsi untuk mengembalikan data *realtime* dari *database* dalam bentuk JSON melalui protokol *WebSocket.* Data yang sudah didapatkan tersebut kemudian ditampilkan dengan menggunakan tabel dan grafik.

C. Perancangan Node Editor pada Platform Pemodelan Pembangkit Listrik Virtual

Dalam merancang *node editor*, penulis menggunakan Rete.js sebagai *framework* Javascript yang mendukung pembuatan *node-based* editor. Rete.js pada platform ini digabungkan dengan *front-end* dari website yang dibuat. *Node* yang dibuat untuk mendukung platform antara lain *node* MQTT (*Subscriber* dan *Publisher*), kalkulasi (*Addition* dan *Multiplication*), *database*, akumulator (*Load* dan *Generator*), Spesifikasi (*Battery* dan *Fuel Cell*) dan kontrol.

1) Node MQTT (Subscriber dan Publisher)

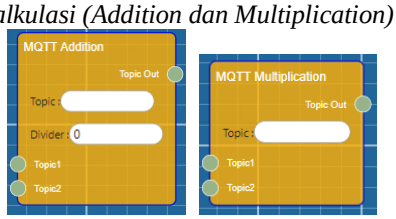


Gambar 6. *Node* MQTT *Client Subscriber* dan *Publisher*

Berdasarkan Gambar 6, kedua *node* MQTT dibentuk untuk *interfacing* antara platform dan perangkat dengan menggunakan MQTT sebagai protokol komunikasi. *Node* MQTT *Client Subscriber* digunakan untuk mendapatkan dan mengambil pesan atau nilai dari broker berdasarkan topik melalui *credentials* broker yang telah didefinisikan. Pada *node* ini terdapat komponen *control* dengan tipe data yang telah didefinisikan meliputi alamat broker, *username*, *password* dan *topic* dengan tipe data string serta *port* dengan tipe data *number* untuk menerima input sesuai yang diberikan oleh pengguna. Selain itu, terdapat juga komponen *output* yang akan mengirimkan pesan yang telah didapatkan berdasarkan topic kepada *node* lain sesuai dengan koneksi yang dibuat (*Node* Kalkulasi atau *Node* MQTT *Database*). *Node* MQTT *Client Publisher* digunakan untuk mengirimkan pesan kepada broker sesuai topik melalui *credentials* broker yang telah didefinisikan. Pada *node* ini terdapat komponen *control* yang hampir sama dengan *Node* MQTT *Client Subscriber* namun terdapat perbedaan jumlah komponen *control* yaitu adanya kolom editor *message* dengan tipe data string untuk menerima input sesuai yang diberikan oleh pengguna.

Dalam pembuatan *backend* dari *Node* MQTT *Client Subscriber* dan *Node* MQTT *Client Publisher* untuk menjalankan fungsi MQTT maka diperlukan pembuatan fungsi khusus untuk mengirimkan atau mengambil nilai pada broker sesuai dengan *credentials* broker yang telah didefinisikan pada *node* tersebut. Untuk membuat fungsi MQTT menggunakan *library* MQTT.js. Dalam *library* tersebut, terdapat fungsi bawaan seperti *connect* untuk menghubungkan *client* dengan broker, *close* untuk memutus koneksi antara *client* dengan broker, *subscribe* untuk mengambil nilai dari broker berdasarkan topik yang telah didefinisikan pada *node* dan untuk mengirim nilai ke broker sesuai topik yang telah didefinisikan pada *node*. Pengiriman dan pengambilan nilai pada broker pada fungsi MQTT yang dibuat menggunakan penjadwalan setiap 5 detik sekali untuk menjalankan fungsi *subscribe* dan *publish*.

2) Node Kalkulasi (Addition dan Multiplication)



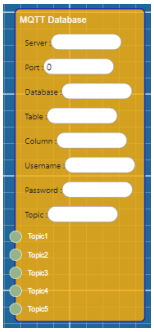
Gambar 7. *Node* MQTT *Addition* dan *Multiplication*

Pada Gambar 7, menujukan bahwa kedua *node* tersebut digunakan untuk kalkulasi. *Node* MQTT *Addition* dibentuk

untuk melakukan kalkulasi berupa penambahan dari pesan yang didapatkan dari *node* MQTT *Client Subscriber* serta pembagian dari pesan yang didapatkan dari *node* lain dengan pembagi berupa bilangan yang diberikan oleh pengguna. Pada *node* ini terdapat komponen *control* dengan tipe data yang telah didefinisikan meliputi *divider* dengan tipe data *number* untuk menerima input sesuai yang diberikan oleh pengguna serta *topic* dengan tipe data string untuk menerima pesan yang diberikan oleh *node* lain (*node* MQTT *Client Subscriber*) melalui komponen *input*. Terdapat juga komponen *output* yang akan mengirimkan hasil kalkulasi dari pesan yang telah didapatkan kepada *node* lain sesuai dengan koneksi yang dibuat (*Node Database*). *Node* MQTT *Multipication* dibentuk untuk untuk melakukan kalkulasi berupa perkalian dari pesan yang didapat dari *node* MQTT *Client Subscriber*. Pada *node* ini terdapat komponen *input* dan *output* yang memiliki fungsi yang sama dengan *Node* MQTT *Addition.* Terdapat juga komponen *control* yang hampir sama dengan *Node* MQTT *Addition* namun terdapat perbedaan jumlah komponen *control* yaitu tidak adanya kolom editor *divider*.

Dalam pembuatan *backend* dari *Node* MQTT *Addition* dan *Node* MQTT *Multipication* maka diperlukan pembuatan fungsi khusus untuk mengkalkulasikan (menambahkan dan mengalikan) nilai yang didapat dari *Node* MQTT *Client Subscriber*. Sebelum melakukan kalkulasi, nilai yang didapatkan tersebut bertipe data *string* dan kemudian mengkonversi nilai tersebut dari tipe data *string* menjadi *float*. Setelah nilai sudah dalam bentuk tipe data *float* maka kemudian akan ditambahkan atau dikalikan dengan nilai lainnya. Selain itu, untuk *Node* MQTT *Addition* juga dapat melakukan pembagian terhadap nilai yang sudah bertipe data *float* dengan pembagi berupa bilangan yang telah didefinisikan pada *node* tersebut.

3) Node Database

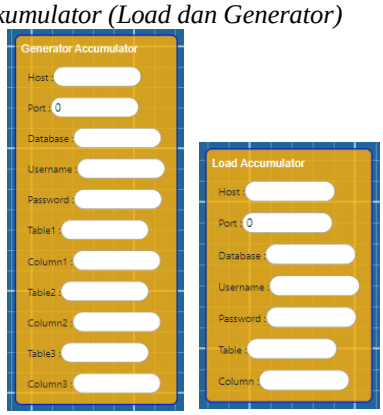


Gambar 8. *Node* MQTT *Database*

Berdasarkan Gambar 8, terlihat bahwa *node* MQTT *Database* dibentuk untuk menyimpan pesan atau nilai yang didapat dari *node* lain (*Node* MQTT *Client Subscriber* atau *Node* Kalkulasi) melalui komponen *input*. Pada *node* ini terdapat komponen *control* dengan tipe data yang telah didefinisikan meliputi alamat server d*atabase*, *database*, *table*, *column*, *username*, *password* dengan tipe data string dan *port* dengan tipe data *number* untuk menerima input sesuai yang diberikan oleh pengguna serta *topic* dengan tipe data string untuk menerima pesan yang diberikan oleh *node* lain (*Node* MQTT *Client Subscriber* atau *Node* Kalkulasi) melalui komponen *input*.

Dalam pembuatan *backend* dari *Node* MQTT *Database* maka diperlukan pembuatan fungsi khusus untuk memasukan rekaman data atau nilai secara *realtime* berdasarkan *timeseries* ke dalam *database*. Sebelum memasukan data *realtime* ke dalam *database* maka diperlukan pengaturan *database* TimescaleDB yaitu instalasi ekstensi TimescaleDB pada layanan PostgreSQL, membuat tabel dengan kolom terakhir pada tabel bertipe data *timestamp* dan kemudian membuat *hypertable* untuk tabel tersebut serta mengatur interval pada tabel *chunk*. Dalam membuat fungsi untuk memasukan nilai ke dalam *database* maka digunakan *library* pg. Dalam *library* tersebut, terdapat fungsi bawaan *Client* atau *Pool* untuk mendefinisikan *connection string* yang berisi *credentials database* dan *connect* yang digunakan untuk membentuk koneksi dengan *database* dan *query* untuk mengeksekusi *query* yang diinginkan. Pembuatan *query* pada TimescaleDB yang ingin dieksekusi tidak jauh berbeda dengan pembuatan *query* pada PostgreSQL. Data *realtime* yang dimasukan ke dalam *database* tidak lebih dari 5 data (tidak termasuk data *timestamp*) sesuai dengan komponen *input* pada *node* tersebut. Proses memasukan data *realtime* ke dalam *database* dilakukan dengan penjadwalan setiap 5 detik sekali untuk menjalankan *query insert* berdasarkan *credentials database* yang telah didefinisikan pada *node* tersebut.

4) Node Akumulator (Load dan Generator)



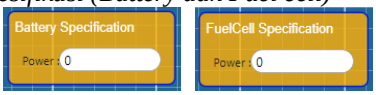
Gambar 9. *Node Generator* dan *Load Accumulator*

Pada Gambar 9, menunjukan bahwa *node Generator* dan *Load Accumulator* dibentuk untuk menghitung total energi yang digunakan selama sehari pada generator dan beban. Total energi yang dihitung berdasarkan akumulasi daya per hari yang didapatkan dari database kemudian diakumulasikan lagi dengan total energi pada hari sebelumnya sehingga menyebabkan total enegi dalam kWh akan bertambah terus menerus setiap harinya. Pada *Node Load Accumulator* terdapat komponen *control* dengan tipe data yang telah didefinisikan yaitu alamat host *database*, *database*, *table*, *column*, *username* dan *password* dengan tipe data string serta *port* dengan tipe data *number* untuk menerima input sesuai yang diberikan oleh pengguna. Pada *Node* Generator *Accumulator* terdapat komponen *control* yang hampir sama dengan *Node Load Accumulator* namun terdapat perbedaan jumlah komponen *control* untuk kolom editor *table*

dan *column*. Hal tersebut dimaksudkan untuk klien yang memiliki jumlah tabel pembangkit lebih dari satu.

Dalam pembuatan *backend* dari *Node Generator* dan *Load Accumulator* maka diperlukan pembuatan fungsi khusus untuk mengakumulasi data berupa daya per hari untuk menghasilkan total energi selama sehari kemudian diakumulasikan lagi dengan total energi pada hari sebelumnya sehingga menyebabkan total enegi dalam kWh akan bertambah terus menerus setiap harinya. Perhitungan total energi dengan mengakumulasi daya dilakukan berdasarkan *timestamp* pada tabel yang didefinisikan pada *node* tersebut. Hal yang dilakukan untuk mendapatkan total energi per hari yaitu mencari daya rata-rata per jam dengan mengelompokan data berdasarkan waktu sehingga dihasilkan energi per jam dalam bentuk Wh kemudian daya yang sudah dirata-ratakan akan dibagi dengan 1000 untuk mendapatkan nilai energi per jam dalam bentuk kWh. Setelah itu, energi per jam akan diakumulasikan selama sehari untuk menghasilkan total energi per hari dan kemudian total energi tersebut akan diakumulasikan lagi dengan total energi pada hari sebelumnya sehingga menyebabkan total enegi dalam kWh akan bertambah terus menerus setiap harinya. Proses perhitungan total energi dilakukan dengan penjadwalan menggunakan fungsi *Cron Job* setiap pukul 23:59.

5) Node Spesifikasi (Battery dan Fuel cell)

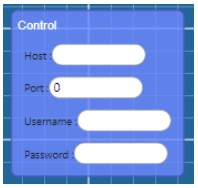


Gambar 10. *Node Battery* and *FuelCell Specification*

Pada Gambar 10, terlihat bahwa *node Battery Specification* dibentuk untuk mendefinisikan kapasitas daya yang diekspor oleh baterai. *Node FuelCell Specification* digunakan untuk mendefinisikan kapasitas daya pada *fuel cell* yang dimiliki oleh pengguna. Kedua *node* ini diperlukan agar agregasi dan logika penentuan keputusan yang telah dibuat dapat berjalan sesuai dengan fungsinya. Pada *node* tersebut juga terdapat komponen *control* dengan tipe data yang telah didefinisikan yaitu *power* dengan tipe data string untuk menerima input sesuai yang diberikan oleh pengguna.

Pembuatan *backend* dari *Node Battery* and *FuelCell Specification* diperlukan untuk mendefinisikan kapasitas dari *fuel cell* dan baterai. Pendefinisian kapasitas dari *fuel cell* dan baterai digunakan oleh algoritma penentuan keputusan untuk mengetahui daya yang akan dihasilkan ketika suatu pembangkit aktif dan kemudian akan menentukan pembangkit yang akan aktif.

6) Node Kontrol



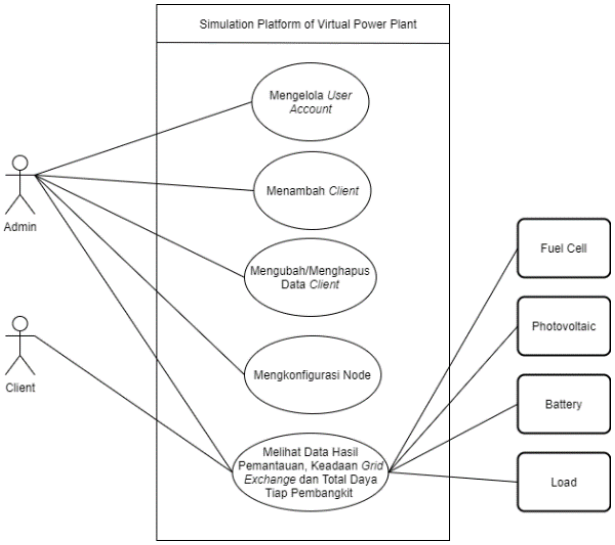
Gambar 11. *Node Control*

Berdasarkan Gambar 11, menunjukan bahwa *node control* digunakan untuk mengontrol pembangkit melalui *credentials* broker yang diberikan oleh pengguna. *Node* ini akan mengirimkan pesan berupa status atau keadaan dari pembangkit (aktif atau tidak aktif) ke broker dengan topik dan pesan yang sudah didefinisikan pada logika yang telah dibuat pada server. Pada *node* ini terdapat komponen *control* dengan tipe data yang telah didefinisikan yaitu alamat broker, *username*, *password* dengan tipe data string serta *port* dengan tipe data *number* untuk menerima input sesuai yang diberikan oleh pengguna.

Pembuatan *backend* dari *Node Control* diperlukan untuk mendefinisikan *credentials* broker yang akan digunakan untuk mengontrol pembangkit. Pendefinisian *credentials* broker digunakan oleh algoritma penentuan keputusan untuk mengirimkan status dari suatu pembangkit ke broker dengan topik dan nilai atau pesan yang sudah didefinisikan pada algoritma yang telah dibuat pada server. Status pembangkit yang dikirimkan ke broker yaitu *fuel cell* dengan status "1" (aktif menghasilkan energi listrik) atau "0" (tidak aktif menghasilkan energi listrik) serta baterai dengan status "3" (*discharge*), "2" (*charge*), dan "1" (tidak aktif atau tidak melakukan *charge* maupun *discharge*).

D. Use Case Diagram pada Platform untuk Pemodelan Pembangkit Listrik Virtual

Untuk mendukung perancangan platform ini, maka penulis menyertakan *use case* diagram untuk menggambarkan interaksi komponen sistem secara keseluruhan yang dapat dilihat pada Gambar 3.13.



Gambar 12. *Use Case* Diagram pada Platform

Pada Gambar 12, terlihat bahwa platform untuk pemodelan pembangkit listrik virtual memiliki dua *role* yaitu sebagai admin dan klien. Dalam platform ini, admin bertindak sebagai *superuser* yang dapat melakukan apapun terhadap platform ini. Hal-hal yang dapat dilukakan oleh admin meliputi mengelola akun pengguna, menambah klien, mengubah atau menghapus data klien, mengkonfigurasi *node*, melihat data tiap klien, total daya tiap pembangkit serta keadaan dari *grid exchange*

berdasarkan selisih dari agregasi daya pada seluruh jenis pembangkit dan beban. Ketika berperan sebagai klien maka hanya dapat melihat data klien yang bersangkutan, total daya tiap pembangkit serta keadaan dari *grid exchange*.

IV. Pemodelan, Pengujian dan Analisis Sistem

A. Pemodelan Sistem

Dalam pemodelan yang dilakukan pada platform sebagai sistem pemantauan untuk pembangkit listrik virtual menggunakan tiga klien yang terdiri dari dua klien sebagai prosumer dengan masing-masing pembangkit (PV dan FC) dan beban yang dimiliki serta satu klien sebagai konsumen yang hanya memiliki baterai sebagai alat untuk menyimpan energi.

Pada pemodelan ini, server dan klien menggunakan *instance* Amazon EC2. Server pada platform untuk pemodelan pembangkit listrik virtual menggunakan Ubuntu Server 18.04 berbasis x86 (64-bit). Server ini digunakan untuk menjalankan *service* utama dan *database* utama dari platform ini. Spesifikasi *instance* yang digunakan pada server sebagai berikut.

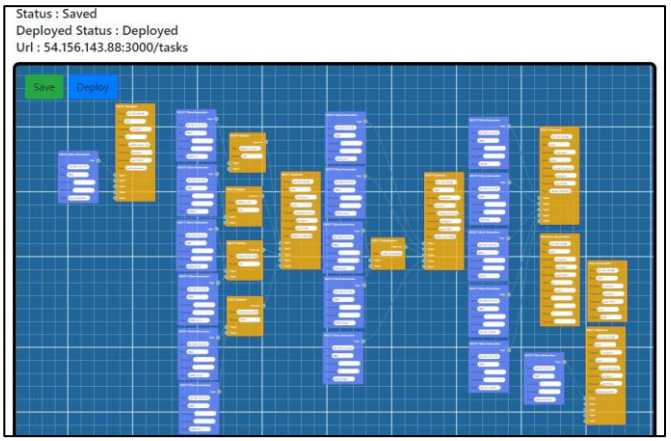
- Jumlah virtual CPU: 1
- RAM: 2 GB
- *Network Bandwidth*: Up to 10 Gbps
- Kapasitas Penyimpanan: 8 Gb

Klien menggunakan Ubuntu Server 18.04 berbasis ARM sebagai representasi dari raspberry pi pada setiap klien. Klien akan menjalankan layanan *backend* yang mampu menjalankan fungsi dari konfigurasi *node editor* yang telah dibuat pada platform. Selain itu, pada setiap klien juga memiliki *database* yang akan menyimpan data-data sesuai dengan klien tersebut. Spesifikasi *instance* pada setiap klien sebagai berikut.

- Jumlah virtual CPU: 1
- RAM: 2 GB
- *Network Bandwidth*: Up to 10 Gbps
- Kapasitas Penyimpanan: 8 Gb

1) Pemodelan Klien Pertama (Photovoltaic)

Pada klien yang pertama menggunakan sistem yang sudah berjalan pada *container* di Fakultas Teknik, Universitas Indonesia dengan data-data sudah ditentukan. Pada sistem tersebut, data-data yang dapat dipantau yaitu data *photovoltaic*, *battery inverter*, *DC converter* serta status atau keadaan dari *circuit breaker* pada pembangkit dan beban. Namun pada sistem yang sudah berjalan, data *photovoltaic* tidak dapat mendukung logika yang sudah didefinisikan dengan baik dikarenakan sensor pada *photovoltaic* rusak atau daya pada baterai sudah penuh yang menyebabkan *photovoltaic* tidak dapat menghasilkan daya yang maksimal sehingga data *photovoltaic* diganti dengan data *dummy* yang mendukung pemodelan ini agar logika dapat terlihat berfungsi dengan baik. Data *dummy* untuk *photovoltaic* telah dimodelkan dalam *script* dengan *photovoltaic* memiliki kapasitas daya produksi tertinggi yaitu 2400 Wp. Dengan demikian, Pada *photovoltaic* memiliki data yang dapat dicatat yaitu daya sedangkan untuk *DC converter* dan *battery inverter* yang dicatat yaitu tegangan, arus, daya dan energi. Selain itu, pada *circuit breaker* yang dicatat adalah status atau keadaan dari *circuit breaker* itu sendiri. Selain itu, pemantauan juga dilakukan pada total energi yang didapatkan dari akumulasi daya per hari pada pembangkit dan beban. Data beban diperlukan untuk mendukung pemodelan ini. Data beban yang digunakan berasal dari data *dummy* dengan beban maksimal 1100 Watt yang dibuat seolah-olah terdapat beban pada sistem di *container*.



Gambar 13. Tampilan Konfigurasi *Node* pada Klien Pertama

Pada Gambar 13, *node* yang digunakan antara lain *node* MQTT *Client Subscriber*, *node* kalkulasi baik *node Addition* maupun *Multiplication*, *node* MQTT *database* serta *node* akumulator untuk generator dan *load*. Ketika konfigurasi *node* telah dilakukan maka konfigurasi *node* tersebut dapat di-*deploy* untuk dikirimkan dalam bentuk JSON ke *client backend*. Pada proses *deploy* juga akan secara otomatis menyimpan konfigurasi *node* tersebut dalam *database* server sehingga konfigurasi *node* tersebut tidak akan hilang ketika pengguna sudah selesai melakukan konfigurasi dan *logout* dari website.

Gambar 14. Tampilan Potongan JSON pada Klien Pertama

Ketika JSON seperti yang dapat dilihat pada Gambar 14 sudah diterima oleh *client backend* maka klien akan menjalankan fungsi sesuai dengan konfigurasi *node* yang telah dibuat. Kemudian data hasil pemataauan akan disimpan pada *database* di klien. Pada *database* TimescaleDB perlu membuat *hypertable* terlebih dahulu yang secara otomatis akan membentuk tabel *chunk* (*automatic chunking*). *Hypertable* yang dibentuk dapat dilihat pada Gambar 15.

id [PK] integer	schema_name name	table_name name	associated_schema_name name	associated_table_prefix name
2	public	dcon	_timescaledb_internal	_hyper_2
1	public	pv_container	_timescaledb_internal	_hyper_1
4	public	state	_timescaledb_internal	_hyper_4
5	public	pyranometer	_timescaledb_internal	_hyper_5
12	public	pv	_timescaledb_internal	_hyper_12
8	public	load	_timescaledb_internal	_hyper_8
9	public	inverter	_timescaledb_internal	_hyper_9
10	public	kwh_generator	_timescaledb_internal	_hyper_10
11	public	kwh_load	_timescaledb_internal	_hyper_11

Gambar 15. Tampilan Hypertable pada Klien Pertama

Data *realtime* tersebut kemudian akan dipartisi sesuai interval yang sudah didefinisikan pada tabel *chunk* sehingga tabel *chunk* akan terus betambah secara otomatis jika timeseries pada data sudah melewati *range* waktu yang sudah ditentukan untuk setiap tabel *chunk*. Daftar tabel *chunk* yang telah terbentuk dapat dilihat pada Gambar 16.

chunk_id integer	chunk_table text	ranges text[]	table_size text
41	_timescaledb_internal._hyper_12_41_chunk	{"['2020-06-04 07:00:00+07','2020-06-11 07:00:00+07')"}	4208 kB
28	_timescaledb_internal._hyper_2_28_chunk	{"['2020-05-28 07:00:00+07','2020-06-04 07:00:00+07')"}	4960 kB
37	_timescaledb_internal._hyper_2_37_chunk	{"['2020-06-04 07:00:00+07','2020-06-11 07:00:00+07')"}	6104 kB
31	_timescaledb_internal._hyper_9_31_chunk	{"['2020-05-28 07:00:00+07','2020-06-04 07:00:00+07')"}	4960 kB
40	_timescaledb_internal._hyper_9_40_chunk	{"['2020-06-04 07:00:00+07','2020-06-11 07:00:00+07')"}	6104 kB
30	_timescaledb_internal._hyper_4_30_chunk	{"['2020-05-28 07:00:00+07','2020-06-04 07:00:00+07')"}	4960 kB
39	_timescaledb_internal._hyper_4_39_chunk	{"['2020-06-04 07:00:00+07','2020-06-11 07:00:00+07')"}	6104 kB
32	_timescaledb_internal._hyper_8_32_chunk	{"['2020-05-28 07:00:00+07','2020-06-04 07:00:00+07')"}	3744 kB
35	_timescaledb_internal._hyper_8_35_chunk	{"['2020-06-04 07:00:00+07','2020-06-11 07:00:00+07')"}	4600 kB
33	_timescaledb_internal._hyper_11_33_chunk	{"['2020-05-28 07:00:00+07','2020-06-04 07:00:00+07')"}	40 kB
43	_timescaledb_internal._hyper_11_43_chunk	{"['2020-06-04 07:00:00+07','2020-06-11 07:00:00+07')"}	40 kB
34	_timescaledb_internal._hyper_10_34_chunk	{"['2020-05-28 07:00:00+07','2020-06-04 07:00:00+07')"}	40 kB
42	_timescaledb_internal._hyper_10_42_chunk	{"['2020-06-04 07:00:00+07','2020-06-11 07:00:00+07')"}	40 kB

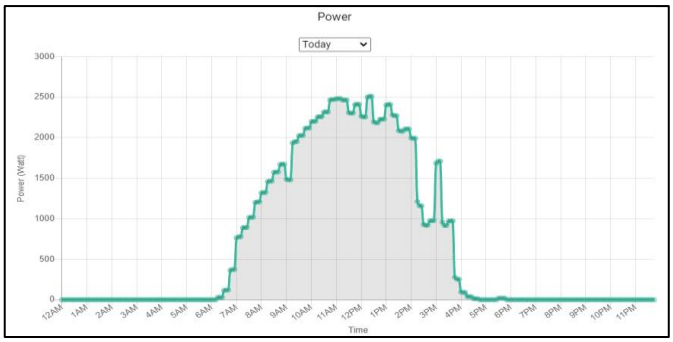
Gambar 16. Daftar Tabel *Chunk* pada Klien Pertama

Data yang sudah disimpan pada *database* kemudian data tersebut ditampilkan pada website melalui tabel dengan 100 data terakhir yang memiliki interval 5 detik dan grafik dengan interval 5 menit. Data yang dapat dilihat pada grafik yaitu data hari ini, data hari kemarin, data dua hari yang lalu dan data tiga hari yang lalu. Salah satu tabel dan grafik yang ditampilkan pada klien pertama yaitu data berupa daya yang dihasilkan oleh *photovoltaic* yang dapat dilihat pada Gambar 17 dan 18.

PV | DCON | Inverter | State | Load | Total Load | Total Power Generation

Power (W)	Input_time
919.1489361	2020-6-18 14:34:22
919.1489361	2020-6-18 14:34:17
919.1489361	2020-6-18 14:34:12
919.1489361	2020-6-18 14:34:7
919.1489361	2020-6-18 14:34:2
919.1489361	2020-6-18 14:33:57
919.1489361	2020-6-18 14:33:52
919.1489361	2020-6-18 14:33:47

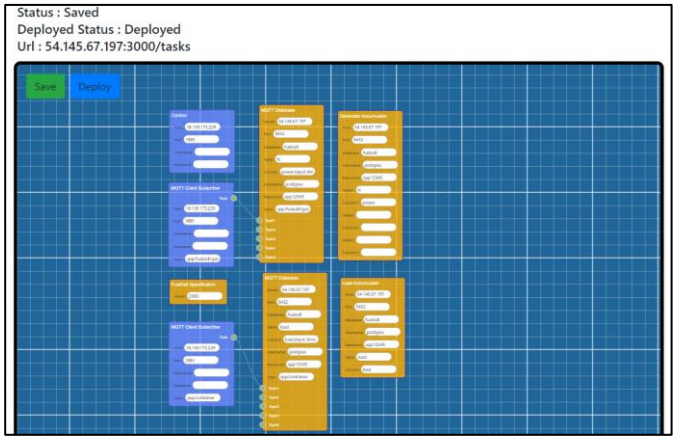
Gambar 17. Tampilan Tabel *Photovoltaic* pada Klien Pertama



Gambar 18. Tampilan Grafik *Power* pada *Photovoltaic* (Klien Pertama)

2) Pemodelan Klien Kedua (Fuel Cell)

Pada klien kedua menggunakan pembangkit *fuel cell* yang ditujukan sebagai pembangkit yang fleksibel yaitu dapat dihidupkan dan dimatikan sesuai dengan logika yang sudah dibuat. Data yang digunakan oleh *fuel cell* merupakan data *dummy* yang dibuat untuk mendukung platform pemodelan ini. Data *dummy* untuk *fuel cell* telah dimodelkan dalam *script* dengan *fuel cell* memiliki kapasitas daya yang dihasilkan untuk diekspor ke *grid* yaitu 2000 Watt. Sama seperti klien pertama, data yang dipantau juga terdapat total energi yang dihasilkan dari akumulasi daya per hari pada pembangkit dan beban. Pada klien kedua, beban yang digunakan sama seperti beban pada klien pertama. Data beban yang digunakan berasal dari data *dummy* dengan beban maksimal yaitu 1100 Watt.



Gambar 19. Tampilan Konfigurasi *Node* pada Klien Kedua

Berdasarkan Gambar 19, terlihat bahwa *node* yang digunakan antara lain *node* MQTT *Client Subscriber*, *node* MQTT *database*, *node* kontrol serta *node* akumulator untuk generator dan *load*. Pada klien ini, perangkat yang dapat dipantau antara lain *fuel cell* dengan data yang dapat dicatat yaitu daya yang dihasilkan. Kemudian konfigurasi *node* tersebut di-*deploy* yang berguna untuk mengirimkan data konfigurasi dalam bentuk JSON ke *client backend* serta data konfigurasi akan secara otomatis disimpan dalam *database* server.

```
{
  "id": "demo@0.2.0",
  "nodes": {
    "243": {
      "id": 243,
      "data": {
        "host": "18.139.173.229",
        "port": 1883,
        "username": "",
        "password": "",
        "topic": "vpp/fuelcell/gen"
      },
      "inputs": {},
      "outputs": {
        "topic": {
          "connections": [
            {
              "node": 244,
              "input": "topicMergel",
              "data": {}
            }
          ]
        }
      },
      "position": [
        153.171211654997,
        272.2190268714912
      ],
      "name": "MQTT Client Subscriber"
    },
    "244": {
      "id": 244,
      "data": {
        "server": "54.145.67.197",
        "portDB": 5432,
        "database": "fuelcell",
        "table": "fc",
        "column": "power",
        "usernameDB": "postgres",
        "passwordDB": "vpp12345",
        "topicStr": "vpp/fuelcell/gen"
      },
      "inputs": {
        "topicMergel": {
```


Gambar 20. Tampilan Potongan JSON pada Klien Kedua

Client backend akan menerima JSON seperti yang terlihat pada Gambar 20 dan menjalankan fungsi sesuai dengan konfigurasi *node* yang telah dibuat. Kemudian data hasil pemantaun akan disimpan pada *database* di klien. Pada *database* TimescaleDB perlu membuat *hypertable* terlebih dahulu yang secara otomatis akan membentuk tabel *chunk* (*automatic chunking*). *Hypertable* yang telah dibentuk dapat dilihat pada Gambar 21.

id [PK] integer	schema_name name	table_name name	associated_schema_name name	associated_table_prefix name
1	public	load	_timescaledb_internal	_hyper_1
3	public	kwh_generator	_timescaledb_internal	_hyper_3
4	public	kwh_load	_timescaledb_internal	_hyper_4
2	public	fc	_timescaledb_internal	_hyper_2

Gambar 21. Tampilan *Hypertable* pada Klien Kedua

Data tersebut kemudian akan dipartisi sesuai interval yang sudah didefinisikan pada tabel *chunk* sehingga tabel *chunk* akan terus betambah secara otomatis jika timeseries pada data sudah melewati *range* waktu yang sudah ditentukan untuk setiap tabel *chunk*. Daftar tabel *chunk* yang telah terbentuk dapat dilihat pada Gambar 22.

chunk_id integer	chunk_table text	ranges text[]	table_size text
9	_timescaledb_internal._hyper_2_9_chunk	{"['2020-05-28 07:00:00+07','2020-06-04 07:00:00+07')"}	4680 kB
13	_timescaledb_internal._hyper_2_13_chunk	{"['2020-06-04 07:00:00+07','2020-06-11 07:00:00+07')"}	4832 kB
8	_timescaledb_internal._hyper_1_8_chunk	{"['2020-05-28 07:00:00+07','2020-06-04 07:00:00+07')"}	4680 kB
12	_timescaledb_internal._hyper_1_12_chunk	{"['2020-06-04 07:00:00+07','2020-06-11 07:00:00+07')"}	4824 kB
10	_timescaledb_internal._hyper_4_10_chunk	{"['2020-05-28 07:00:00+07','2020-06-04 07:00:00+07')"}	40 kB
14	_timescaledb_internal._hyper_4_14_chunk	{"['2020-06-04 07:00:00+07','2020-06-11 07:00:00+07')"}	40 kB
11	_timescaledb_internal._hyper_3_11_chunk	{"['2020-05-28 07:00:00+07','2020-06-04 07:00:00+07')"}	40 kB
15	_timescaledb_internal._hyper_3_15_chunk	{"['2020-06-04 07:00:00+07','2020-06-11 07:00:00+07')"}	40 kB

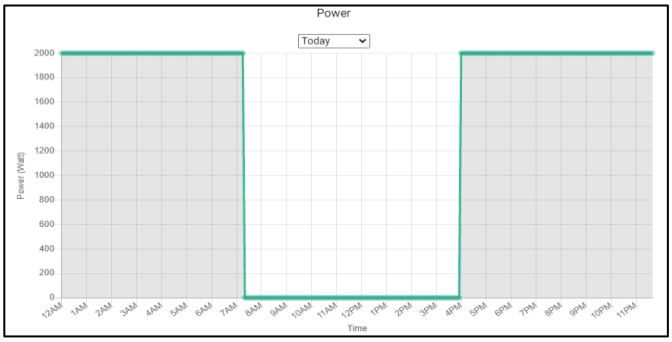
Gambar 22. Daftar Tabel *Chunk* pada Klien Kedua

Data *realtime* yang sudah disimpan pada *database* di klien kemudian akan ditampilkan pada website melalui tabel dengan 100 data terakhir yang memiliki interval 5 detik dan grafik dengan interval 5 menit. Data yang dapat dilihat pada grafik yaitu data hari ini, data hari kemarin, data dua hari yang lalu dan data tiga hari yang lalu. Salah satu tabel dan grafik yang ditampilkan pada klien kedua yaitu data berupa daya yang dihasilkan oleh *fuel cell* dapat dilihat pada Gambar 23 dan 24.

Fuelcell | Load | Total Load | Total Power Generation

Power (W)	Input_time
2000	2020-6-18 14:36:47
2000	2020-6-18 14:36:42
2000	2020-6-18 14:36:37
2000	2020-6-18 14:36:32
2000	2020-6-18 14:36:27
2000	2020-6-18 14:36:22
2000	2020-6-18 14:36:17
2000	2020-6-18 14:36:12

Gambar 23. Tampilan Tabel *Fuel Cell* pada Klien Kedua

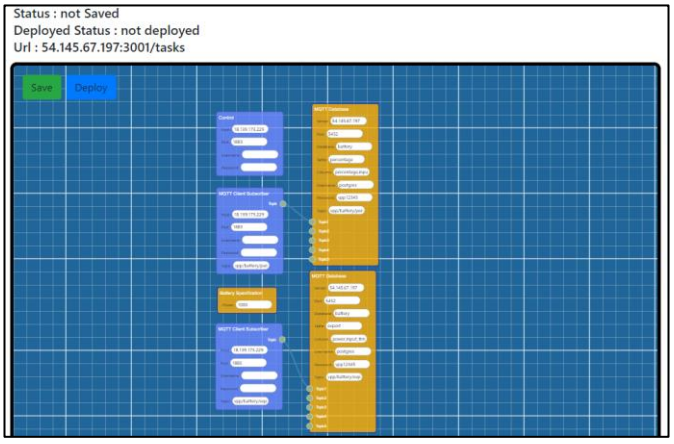


Gambar 24. Tampilan Grafik Power pada Fuel Cell (Klien Kedua)

3) Pemodelan Klien Ketiga (Baterai)

Klien ketiga dalam pemodelan ini mempunyai baterai sebagai alat penyimpanan energi. Baterai mengisi daya (*charging*) ketika daya tersedia dalam *grid* atau total daya yang dihasilkan oleh pembangkit lebih besar daripada total beban yang terdapat pada *grid.* Baterai akan melakukan *discharge* untuk membantu mengurangi beban pada *grid* ketika daya tidak menncukupi dalam *grid* atau total beban lebih besar daripada total daya yang dihasilkan oleh pembangkit. Pada pemodelan ini, menggunakan data *dummy* dengan parameter pada baterai

yang akan dipantau yaitu kapasitas baterai dalam persentase dan daya maksimal pada baterai yang di-*discharge* ke *grid*. Data *dummy* pada baterai telah dimodelkan dalam *script* dengan baterai memiliki kapasitas total yaitu 19200 Wh dengan daya yang diekspor oleh baterai sebesar 1000 Watt.



Gambar 25. Tampilan Konfigurasi *Node* pada Klien Ketiga

Berdasarkan Gambar 25, menunjukan bahwa *node* yang digunakan sama seperti pada klien kedua yaitu *node* MQTT *Client Subscriber*, *node* MQTT *database*, *node* kontrol serta *node* akumulator untuk generator dan load. Pada klien ini, perangkat yang dapat dipantau antara lain baterai dengan data yang dapat dicatat yaitu daya yang diekspor oleh baterai dan kapasitas baterai dalam persentase. Konfigurasi *node* tersebut akan di-*deploy* untuk mengirimkan data konfigurasi dalam bentuk JSON ke *client backend* serta data konfigurasi secara otomatis disimpan dalam *database* server sehingga konfigurasi *node* tersebut tidak akan hilang.

Gambar 26. Tampilan Potongan JSON pada Klien Ketiga

Client backend akan menerima konfigurasi *node* berupa JSON seperti yang terlihat pada Gambar 26 kemudian menjalankan fungsi sesuai dengan konfigurasi *node* yang telah dibuat. Data hasil pemantaun akan disimpan pada *database* di klien. Pada *database* TimescaleDB perlu membuat *hypertable* terlebih dahulu yang secara otomatis akan membentuk tabel *chunk* (*automatic chunking*). *Hypertable* yang telah dibentuk dapat dilihat pada Gambar 27.

id [PK] integer	schema_name name	table_name name	associated_schema_name name	associated_table_prefix name
1	public	percentage	_timescaledb_internal	_hyper_1
2	public	export	_timescaledb_internal	_hyper_2

Gambar 27. Tampilan Hypertable pada Klien Ketiga

Data tersebut kemudian akan dipartisi sesuai interval yang sudah didefinisikan pada tabel *chunk* sehingga tabel *chunk* akan terus betambah secara otomatis jika timeseries pada data sudah melewati *range* waktu yang sudah ditentukan untuk setiap tabel *chunk*. Daftar tabel *chunk* dapat dilihat pada Gambar 28.

chunk_id integer	chunk_table text	ranges text[]	table_size text
3	_timescaledb_internal._hyper_1_3_chunk	{"['2020-05-28 07:00:00+07','2020-06-04 07:00:00+07')"}	4680 kB
5	_timescaledb_internal._hyper_1_5_chunk	{"['2020-06-04 07:00:00+07','2020-06-11 07:00:00+07')"}	4576 kB
chunk_id integer	**chunk_table text**	**ranges text[]**	**table_size text**
4	_timescaledb_internal._hyper_2_4_chunk	{"['2020-05-28 07:00:00+07','2020-06-04 07:00:00+07')"}	4680 kB
6	_timescaledb_internal._hyper_2_6_chunk	{"['2020-06-04 07:00:00+07','2020-06-11 07:00:00+07')"}	4568 kB

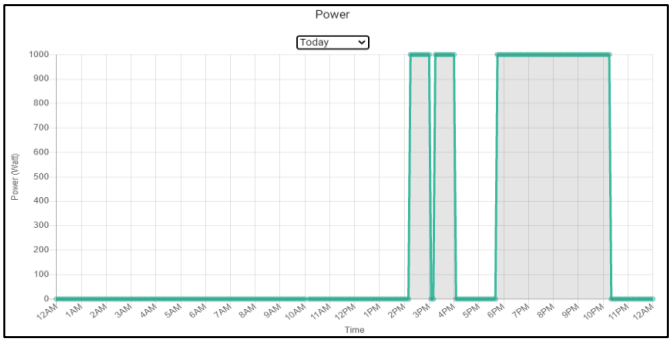
Gambar 28. Daftar Tabel Chunk pada Battery Generation

Data *realtime* yang sudah disimpan pada *database* di klien akan ditampilkan pada website menggunakan tabel dengan 100 data terakhir yang memiliki interval 5 detik dan grafik dengan interval 5 menit. Data yang dapat dilihat pada grafik yaitu data hari ini, data hari kemarin, data dua hari yang lalu dan data tiga hari yang lalu. Salah satu tabel dan grafik yang ditampilkan pada klien ketiga yaitu data *battery generation* yang merupakan daya untuk melakukan *discharge* ke *grid* yang dapat dilihat pada Gambar 29 dan**Error! Reference source not found.**.

Battery Generation | Battery Percentage

Power (W)	Input_time
1000	2020-6-10 19:54:47
1000	2020-6-10 19:54:42
1000	2020-6-10 19:54:37
1000	2020-6-10 19:54:32
1000	2020-6-10 19:54:27
1000	2020-6-10 19:54:22
1000	2020-6-10 19:54:17
1000	2020-6-10 19:54:12

Gambar 29. Tampilan Tabel *Battery Generation* pada Klien Ketiga



Gambar 30. Tampilan Grafik *Power* pada *Battery Generation* (Klien Ketiga)

B. Pengujian Sistem

Berdasarkan perancangan dan implementasi sistem yang telah dilakukan, maka diperlukan pengujian sistem yang bertujuan untuk mengetahui dan menilai fleksibilitas dalam penggunaan atau pengoperasian platform pemodelan. Pada pengujian sistem dibagi menjadi dua bagian yaitu pengujian *node editor* dan *usability testing*.

1) Pengujian Node Editor

Pengujian yang dilakukan pada *node editor* bertujuan untuk mengetahui kemampuan adaptasi *node editor* dalam mendukung platform. Dalam pengujian ini, terdapat beberapa parameter yang akan diuji yaitu sebagai berikut:

1. *Node* dapat ditambahkan melalui menu yang tersedia
2. *Node* dapat dihapus (*delete*) dan diduplikasi (*clone*)
3. *Node* dapat dicari melalui fitur *search*
4. *Node* dapat dibesarkan atau dikecilkan
5. *Node* dapat dihubungkan dengan *node* lain

2) Usability Testing

Pada Usability Testing, pengujian dilakukan dengan memberikan kuesioner mengenai penilaian pengguna terhadap platform untuk pemodelan pembangkit listrik virtual. Sebelum mengisi kuesioner, para responden telah diminta untuk mencoba terlebih dahulu dengan dipandu dalam mengoperasikan platform ini. Para responden dibagi menjadi dua kelompok yang terdiri dari responden yang mengenal atau memahami konsep pembangkit listrik virtual serta responden yang tidak memahami konsep pembangkit listrik virtual. Penilaian dilakukan terhadap penggunaan *node editor* dan tampilan antarmuka pada setiap klien di platform ini. Responden diminta untuk memberikan penilaian dalam angka dengan skala 1-5 yang masing-masing angka mewakili kategori atau kriteria sebagai berikut:

1 = Sangat Tidak Setuju (STS)
2 = Kurang Setuju (KS)
3 = Cukup (C)
4 = Setuju (S)
5 = Sangat Setuju (SS)

Dalam menguji penggunaan *node editor* pada platform ini, terdapat poin-poin yang dijadikan parameter pengujian yaitu:

1. Kemudahan mempelajari alur kerja dari platform.
2. Kejelasan dalam memahami fitur yang terdapat pada *node editor*.
3. Penggunaan *node editor* dapat mendukung konfigurasi pada platform.
4. Kemudahan mempelajari penggunaan *node editor*.
5. Dibutuhkan training dalam mengoperasikan *node editor*.

Sedangkan untuk menguji tampilan antarmuka pada setiap klien pada platform ini, terdapat poin-poin yang dijadikan parameter pengujian yaitu:

1. Tampilan antarmuka pada setiap klien secara keseluruhan menarik.
2. Tampilan *node editor* secara keseluruhan mudah dipahami dan menarik.
3. Kejelasan detail informasi yang ditampilkan pada tabel dan grafik.

C. Hasil dan Analisis Pengujian Sistem

Bedasarkan pengujian yang telah dilakukan maka diperlukan analisis untuk melihat hasil dan kesimpulan yang didapatkan dari parameter yang diuji yaitu hasil dan analisis terhadap pengujian *node editor* dan *usability testing*.

1) Hasil dan Analisis Pengujian Node Editor

Berdasarkan pengujian *node editor* yang telah dilakukan, menunjukan bahwa perancangan *node editor* dengan menggunakan *framework* Rete.js sebagai *visual programming* dapat memberikan fitur - fitur dan komponen yang mendukung platform untuk pemodelan pembangkit listrik virtual sehingga dapat berjalan sesuai rancangan. Untuk merangkum pengujian *node editor* yang telah dilakukan, maka terlihat pada Tabel 1.

TABEL I. PENGUJIAN *NODE EDITOR*

Parameter	Berhasil	Gagal
Node dapat ditambahkan melalui menu yang tersedia	✓	
Node dapat dihapus (*delete*) dan diduplikasi (*clone*)	✓	
Node dapat dicari melalui fitur *search*	✓	
Node dapat dibesarkan atau dikecilkan	✓	
Node dapat dihubungkan dengan *node* lain	✓	

2) Hasil dan Analisis Usability Testing

Analisis diperlukan untuk mengetahui hasil dan kesimpulan dari pengujian yang telah dilakukan melalui *usability testing.* Dalam pengujian ini, *usability testing* dilakukan dengan memberikan kuesioner kepada 20 responden yang bertujuan untuk menilai penggunaan platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual terutama tentang penggunaan *node editor* dan tampilan antarmuka pada setiap klien. Data hasil kuesioner ini dibagi menjadi dua jenis pengelompokan berdasarkan memahami dan tidak memahami konsep pembangkit listrik virtual. Data hasil kuesioner yang telah dikumpulkan, kemudian akan digunakan untuk mencari *Mean of Survey* (MoS) dan standar deviasi dari tiap parameter penilaian. Untuk mencari standar deviasi dapat dilihat pada Persamaan 4.1 [7].

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \qquad (4.1)$$

Keterangan:

- σ = standar deviasi
- x_i = nilai sampel ke-i
- $\bar{x}$ = rata-rata
- N = jumlah sampel

Namun berdasarkan data hasil kuesioner hanya sebanyak 20 responden yang melakukan penilaian sehingga hasil yang didapat dari penilaian masih kurang mewakili untuk populasi

data yang lebih besar. Hal ini membutuhkan perhitungan *confidence interval* untuk mengukur ketepatan atau keakuratan dari nilai rata-rata data sampel yang didapat dari kuesioner (*Mean of Survey*) apakah dapat mewakili nilai rata-rata pada populasi data sesungguhnya. Dalam menghitung *confidence interval* dapat dilihat pada Persamaan 4.2 [8].

$$CI = \bar{x} \pm T_{\alpha/2} \frac{\sigma}{\sqrt{n}} \qquad (4.2)$$

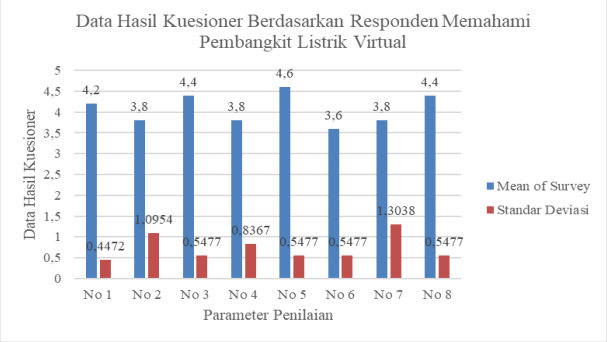
Keterangan:

- CI = *confidence interval*
- T = nilai T
- α = alpha
- σ = standar deviasi
- $\bar{x}$ = rata-rata
- n = jumlah sampel

Terdapat batas kesalahan maksimum baik batas atas maupun batas bawah yang dapat ditoleransi atau diterima (*margin of error*) pada data sampel sehingga dapat memengaruhi hasil dari perhitungan *confidence interval*.

a) Responden Memahami Pembangkit Listrik Virtual

Berdasarkan hasil kuesioner dengan 20 responden terdapat 5 responden yang memahami konsep pembangkit listrik virtual. Dari 5 responden tersebut, maka didapatkan *Mean of Survey* dan standar deviasi dari tiap parameter yang dihasilkan dari penilaian terhadap penggunaan *node editor* dan tampilan antarmuka pada setiap klien yang dapat dilihat pada Gambar 31.



Gambar 31. Grafik Data Hasil Kuesioner untuk Penilaian Berdasarkan Responden Memahami Pembangkit Listrik Virtual

Mean of Survey dan standar deviasi digunakan untuk menghitung *confidence interval* pada setiap parameter penilaian. Dalam menghitung *confidence interval* maka digunakan *confidence level* 80% yang menghasilkan nilai T yaitu 1,533. Berikut ini adalah analisis hasil penilaian responden pada masing-masing parameter yang diuji meliputi:

1. Parameter pertama: Mampu mempelajari alur kerja dari platform dengan singkat
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 4,2 dan 0,4472. Selain itu, juga didapatkan *confidence interval* 4,2 ± 0,3066 dengan rentang 3,8934 - 4,5066. Berdasarkan *Mean of Survey* untuk parameter ini yaitu 4,2 maka dapat dikategorikan "Setuju". Hal ini menunjukan bahwa alur kerja pada platform ini mudah dipahami dan dipelajari bagi para responden meskipun mereka baru pertama kali mencoba mengoperasikan platform ini.
2. Parameter kedua: Mampu memahami fitur yang terdapat pada *node editor*
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 3,8 dan 1,0954. *Confidence interval* yang dihasilkan yaitu 3,8 ± 0,751 dengan rentang 3,049 – 4,551. Berdasarkan *Mean of Survey* 3,8 maka dapat dikategorikan "Cukup". Hal ini menunjukan bahwa fitur yang terdapat pada *node editor* seperti *search, delete* dan *clone* cukup dapat dipahami namun bagi beberapa responden kurang dapat dipahami karena diperlukan panduan.
3. Parameter ketiga: Penggunaan *node editor* dapat mendukung konfigurasi pada platform
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 4,4 dan 0,5477. Selain itu, juga didapatkan *confidence interval* 4,4 ± 0,3755 dengan rentang 4,0245 – 4,7755. Berdasarkan *Mean of Survey* 4,4 maka dapat dikategorikan "Setuju". Hal ini menunjukan bahwa responden merasa terbantu dengan adanya *node editor* yang dapat mendukung konfigurasi pada platform ini meskipun mereka baru mencoba mengoperasikan platform ini.
4. Parameter keempat: Mampu mempelajari penggunaan *node editor* dengan singkat
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 3,8 dan 0,8367. *Confidence interval* menghasilkan nilai 3,8 ± 0,5736 dengan rentang 3,2264 – 4,3736. Berdasarkan *Mean of Survey* 3,8 maka dapat dikategorikan "Cukup". Hal ini menunjukan bahwa penggunaan *node editor* kurang dapat dipahami dengan singkat bagi beberapa responden karena diperlukan petunjuk seperti halaman yang menjelaskan bagaimana cara pengoperasian *node editor* pada platform ini.
5. Parameter kelima: Dibutuhkan *training* dalam mengoperasikan *node editor*
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 4,6 dan 0,5477. *Confidence interval* menghasilkan nilai 4,6 ± 0,3755 dengan rentang 4,2245 – 4,9755. Berdasarkan *Mean of Survey* 4,6 maka dapat dikategorikan "Setuju". Hal ini menunjukan bahwa responden memerlukan *training* dalam mengoperasikan *node editor* pada platform ini sehingga diperlukan waktu untuk dapat memahami pengoperasian platform secara keseluruhan.
6. Parameter keenam: Tampilan antarmuka pada setiap klien secara keseluruhan menarik
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 3,6 dan 0,5477.

Confidence interval yang dihasilkan yaitu 3,6 ± 0,3755 dengan rentang 3,2245 – 3,9755. Berdasarkan *Mean of Survey* 3,6 maka dapat dikategorikan "Cukup". Hal ini menunjukan bahwa tampilan antarmuka pada setiap klien cukup menarik namun beberapa responden menganggap masih kurang menarik dan terkesan kaku.

7. Parameter ketujuh: Tampilan *node editor* secara keseluruhan mudah dipahami dan menarik
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 3,8 dan 1,3038. Selain itu, juga didapatkan *confidence interval* 3,8 ± 0,8939 dengan rentang 2,9061 – 4,6939. Berdasarkan *Mean of Survey* 3,8 maka dapat dikategorikan "Cukup". Hal ini menunjukan bahwa tampilan *node editor* secara keseluruhan cukup dapat dipahami dan menarik namun beberapa responden menganggap kurang dapat dipahami dan kurang menarik.
8. Parameter kedelapan: Detail informasi yang ditampilkan pada tabel dan grafik sudah jelas
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 4,4 dan 0,5477. *Confidence interval* menghasilkan nilai 4,4 ± 0,3755 dengan rentang 4,0245 – 4,7755. Berdasarkan *Mean of Survey* 4,4 maka dapat dikategorikan "Setuju". Hal ini menunjukan bahwa responden dapat memahami dengan jelas detail informasi yang ditampilkan pada tabel dan grafik.

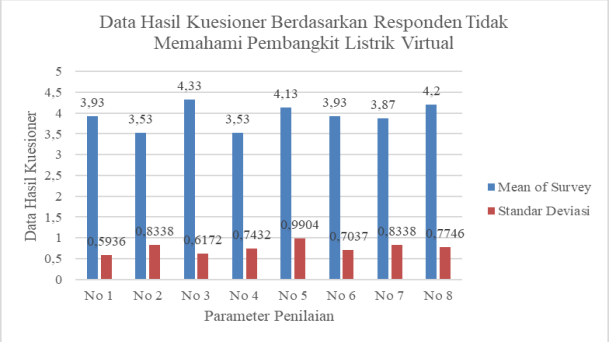
Untuk lebih memperjelas, *confidence interval* pada setiap parameter dapat dilihat pada Tabel 2.

TABEL II. CONFIDENCE INTERVAL BERDASARKAN RESPONDEN MEMAHAMI PEMBANGKIT LISTRIK VIRTUAL

No	Parameter Penilaian	*Mean of Survey*	*Margin of Error*	*Confidence Interval*
1.	Mampu mempelajari alur kerja dari platform dengan singkat	4,2	0,3066	3,8934 - 4,5066
2.	Mampu memahami fitur yang terdapat pada *node editor*	3,8	0,751	3,049 – 4,551
3.	Penggunaan *node editor* dapat mendukung konfigurasi pada platform	4,4	0,3755	4,0245 – 4,7755
4.	Mampu mempelajari penggunaan *node editor* dengan singkat	3,8	0,5736	3,2264 – 4,3736
5.	Dibutuhkan training dalam mengoperasikan *node editor*	4,6	0,3755	4,2245 – 4,9755
6.	Tampilan antarmuka pada setiap klien secara keseluruhan menarik	3,6	0,3755	3,2245 – 3,9755
7.	Tampilan *node editor* secara keseluruhan mudah dipahami dan menarik	3,8	0,8939	2,9061 – 4,6939
8.	Detail informasi yang ditampilkan pada tabel dan grafik sudah jelas	4,4	0,3755	4,0245 – 4,7755

b) Responden Tidak Memahami Pembangkit Listrik Virtual

Berdasarkan hasil kuesioner dengan 20 responden terdapat 15 responden yang tidak memahami konsep pembangkit listrik virtual (orang awam). Dari 15 responden tersebut, maka didapatkan *Mean of Survey* dan standar deviasi dari tiap parameter yang dihasilkan dari penilaian terhadap penggunaan *node editor* dan tampilan antarmuka pada setiap klien yang dapat dilihat pada Gambar 32.



Gambar 32. Grafik Data Hasil Kuesioner untuk Penilaian Berdasarkan Responden Tidak Memahami Pembangkit Listrik Virtual

Mean of Survey dan standar deviasi yang yang telah diolah akan digunakan untuk menghitung *confidence interval* pada setiap parameter. Dalam mencari *confidence interval* maka digunakan *confidence level* 80% yang menghasilkan nilai T yaitu 1,345. Berikut ini adalah analisis hasil penilaian responden pada masing-masing parameter yang diuji meliputi:

1. Parameter pertama: Mampu mempelajari alur kerja dari platform dengan singkat
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 3,93 dan 0,5936. *Confidence interval* memiliki nilai 3,93 ± 0,2061 dengan rentang 3,7272 - 4,1395. Berdasarkan *Mean of Survey* untuk parameter ini yaitu 3,93 maka dapat dikategorikan "Cukup". Hal ini menunjukan bahwa beberapa responden sebagai orang awam kurang memahami alur kerja pada platform ini sehingga diperlukan panduan yang menggambarkan kerja sistem pada platform secara keseluruhan seperti tambahan menu navigasi.
2. Parameter kedua: Mampu memahami fitur yang terdapat pada *node editor*
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 3,53 dan 0,8338. *Confidence interval* yang dihasilkan yaitu 3,53 ± 0,2896 dengan rentang 3,2438 – 3,8229. Berdasarkan *Mean of Survey* 3,53 maka dapat dikategorikan "Cukup". Hal ini menunjukan bahwa fitur yang terdapat pada *node editor* seperti *search, delete* dan *clone* kurang dapat dipahami bagi beberapa responden karena diperlukan panduan.
3. Parameter ketiga: Penggunaan *node editor* dapat mendukung konfigurasi pada platform

Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 4,33 dan 0,6172. *Confidence interval* yang dihasilkan yaitu 4,33 ± 0,2143 dengan rentang 4,119 – 4,5477. Berdasarkan *Mean of Survey* 4,33 maka dapat dikategorikan "Setuju". Hal ini menunjukan bahwa responden yang merupakan orang awam merasa terbantu dengan adanya *node editor* yang dapat mendukung konfigurasi pada platform ini meskipun ketika mengoperasikannya perlu waktu untuk menjelaskan lebih lanjut agar mudah dimengerti.

4. Parameter keempat: Mampu mempelajari penggunaan *node editor* dengan singkat
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 3,53 dan 0,7432. *Confidence interval* yang menghasilkan nilai 3,53 ± 0,2581 dengan rentang 3,2752 – 3,7914. Berdasarkan *Mean of Survey* 3,53 maka dapat dikategorikan "Cukup". Hal ini menunjukan bahwa penggunaan *node editor* kurang dapat dipahami dengan singkat bagi responden sebagai orang awam karena diperlukan petunjuk seperti halaman yang menjelaskan bagaimana cara pengoperasian *node editor* pada platform ini.
5. Parameter kelima: Dibutuhkan *training* dalam mengoperasikan *node editor*
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 4,13 dan 0,9904. *Confidence interval* 4,13 ± 0,344 dengan rentang 3,7894 – 4,4773. Berdasarkan *Mean of Survey* 4,13 maka dapat dikategorikan "Setuju". Hal ini menunjukan bahwa responden memerlukan *training* dalam mengoperasikan *node editor* pada platform ini sehingga diperlukan waktu untuk dapat memahami pengoperasian platform secara keseluruhan.
6. Parameter keenam: Tampilan antarmuka pada setiap klien secara keseluruhan menarik
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 3,93 dan 0,7037. *Confidence interval* memiliki nilai 3,93 ± 0,2444 dengan rentang 3,6889 – 4,1777. Berdasarkan *Mean of Survey* 3,93 maka dapat dikategorikan "Cukup". Hal ini menunjukan bahwa tampilan antarmuka pada setiap klien cukup menarik namun beberapa responden sebagai orang awam menganggap masih kurang menarik karena masih terdapat ruang kosong pada halaman tertentu.
7. Parameter ketujuh: Tampilan *node editor* secara keseluruhan mudah dipahami dan menarik
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 3,87 dan 0,8338. Selain itu, juga didapatkan *confidence interval* 3,87 ± 0,2896 dengan rentang 3,5771 – 4,1562. Berdasarkan *Mean of Survey* 3,87 maka dapat dikategorikan "Cukup". Hal ini menunjukan bahwa tampilan *node editor* secara keseluruhan cukup dapat dipahami dan menarik namun beberapa responden (orang awam) menganggap kurang dapat dipahami dan kurang menarik.
8. Parameter kedelapan: Detail informasi yang ditampilkan pada tabel dan grafik sudah jelas
   Pada parameter ini terlihat bahwa *Mean of Survey* dan standar deviasi yang didapatkan yaitu 4,2 dan 0,7746. *Confidence interval* yang dihasilkan yaitu 4,2 ± 0,269 dengan rentang 3,931 – 4,469. Berdasarkan *Mean of Survey* 4,2 maka dapat dikategorikan "Setuju". Hal ini menunjukan bahwa responden dapat memahami dengan jelas detail informasi yang ditampilkan pada tabel dan grafik.

Untuk lebih memperjelas, *confidence interval* pada setiap parameter dapat dilihat pada Tabel 3.

TABEL III. CONFIDENCE INTERVAL BERDASARKAN RESPONDEN TIDAK MEMAHAMI PEMBANGKIT LISTRIK VIRTUAL

No	Parameter Penilaian	*Mean of Survey*	*Margin of Error*	*Confidence Interval*
1.	Mampu mempelajari alur kerja dari platform dengan singkat	*3,93*	*0,2061*	*3,7272 - 4,1395*
2.	Mampu memahami fitur yang terdapat pada *node editor*	3,53	0,2896	3,2438 – 3,8229
3.	Penggunaan *node editor* dapat mendukung konfigurasi pada platform	4,33	0,2143	4,119 – 4,5477
4.	Mampu mempelajari penggunaan *node editor* dengan singkat	3,53	0,2581	3,2752 – 3,7914
5.	Dibutuhkan training dalam mengoperasikan *node editor*	4,13	0,344	3,7894 – 4,4773
6.	Tampilan antarmuka pada setiap klien secara keseluruhan menarik	3,93	0,2444	3,6889 – 4,1777
7.	Tampilan *node editor* secara keseluruhan mudah dipahami dan menarik	3,87	0,2896	3,5771 – 4,1562
8.	Detail informasi yang ditampilkan pada tabel dan grafik sudah jelas	4,2	0,269	3,931 – 4,469

c) Hasil dan Analisis Keseluruhan Usability Testing

Menurut analisis yang sudah dilakukan, maka dapat disimpulkan bahwa keseluruhan *Mean of Survey* berdasarkan responden memahami pembangkit listrik virtual yaitu 4,1 dalam skala 5 sedangkan untuk responden tidak memahami pembangkit lsitrik virtual yaitu 3,93 dalam skala 5. Hal ini menunjukan bahwa responden merasa cukup puas dengan pengalaman dalam mengoperasikan platform ini baik dalam segi penggunaan *node editor* maupun tampilan antarmuka pada setiap klien. Namun beberapa responden memerlukan *training* atau panduan dalam mengoperasikan platform ini.

Confidence interval dengan *confidence level* 80% untuk setiap parameter ini memiliki interval yang cukup lebar dengan

margin of error yang bervariasi. Hal tersebut dipengaruhi oleh ukuran data sampel yang sedikit. Pada pengelompokan berdasarkan responden memahami pembangkit listrik virtual memiliki *margin of error* yaitu 0,3066 – 0,8939 sedangkan pengelompokan berdasarkan responden tidak memahami pembangkit listrik virtual memiliki *margin of error* yaitu 0,2061 – 0,2896.

V. KESIMPULAN

Dalam merancang platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual maka penulis telah berhasil membangun *node editor* dengan Rete.js sebagai *visual programming* untuk membantu pengguna dalam melakukan seluruh konfigurasi yaitu *interfacing* antara platform dengan pembangkit menggunakan protokol MQTT, melakukan kalkulasi, menyimpan data hasil pemantauan serta menghitung total energi pada pembangkit dan beban. Selain itu, penulis telah berhasil membangun sistem pemantauan untuk membantu klien dalam melihat seluruh aktivitas pembangkit dan beban meliputi daya pada pembangkit dan beban, kapasitas baterai, serta total energi yang dihasilkan dari akumulasi daya per hari pada beban dan pembangkit. Penilaian penggunaan platform untuk pemodelan pembangkit listrik virtual oleh responden dapat dikategorikan cukup baik dengan *Mean of Survey* (MoS) berdasarkan responden memahami pembangkit listrik virtual yaitu 4,1 sedangkan untuk responden tidak memahami pembangkit lsitrik virtual yaitu 3,93. *Confidence interval* dengan tingkat kepercayaan 80% untuk setiap parameter memiliki interval yang cukup lebar dengan margin of error yang bervariasi yaitu 0,3066 – 0,8939 untuk responden memahami pembangkit listrik virtual sedangkan 0,2061 – 0,2896 untuk responden tidak memahami pembangkit listrik virtual.

Apabila platform untuk pemodelan pembangkit listrik virtual ingin dikembangkan lebih lanjut dari segi tampilan antarmuka dan alur kerja pada sistem maka dapat dilakukan beberapa pengembangan pada platform yang meliputi sebagai berikut:

1. Tampilan antarmuka dapat diperindah atau dikembangkan seperti dengan menambahkan komponen tertentu agar desain lebih menarik.
2. Pembuatan *node* lain untuk mendukung platform ini seperti penambahan protokol komunikasi selain menggunakan MQTT.
3. Pembentukan pricing model untuk mendukung transaksi sehingga menghasilkan nilai untung dan rugi.
4. Pemanfaatan konsep *Big Data analytics* dan *machine learning* dalam mengolah data yang terdapat pada platform ini untuk dijadikan sebagai parameter dalam penentuan algoritma seperti melihat tren yang ada, menemukan pola tertentu dan mencari korelasi terhadap variabel tertentu.

REFERENSI

[1] E. A. Setiawan, “Concept and Controllability of Virtual Power Plant”, Kassel: Kassel Univ. Press, 2007.

[2] C. Giron and S. Omran, “Virtual Power Plant for a Smart Grid: A Technical Feasibility Case Study”, INTERNATIONAL JOURNAL of RENEWABLE ENERGY RESEARCH, vol. 8, no.2, June, 2018.

[3] L. Dulau, M. Abrudean and D. Bica, “Distributed generation and virtual power plants”, 49th International Universities Power Engineering Conference (UPEC), 2014. Available: 10.1109/upec.2014.6934630.

[4] D. Zubov, “An IoT Concept of the Small Virtual Power Plant Based on Arduino Platform and MQTT Protocol”, Proccedings of The ICAIIT2016, 2016. Available: 10.20544/aiit2016.13

[5] M. Rouse. “What is MQTT (MQ Telemetry Transport)?”. [Online]. Available: https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport [Accessed: 31 May 2020]

[6] Steves. “MQTT Protocol Messages Overview”. [Online]. Available: http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/ [Accessed: 31 May 2020]

[7] Khan Academy. 2020. “Standard Deviation: Calculating Step by Step”. [Online]. Available: https://www.khanacademy.org/math/probability/data-distributions-a1/summarizing-spread-distributions/a/calculating-standard-deviation-step-by-step [Accessed: 23 June 2020]

[8] L. Sullivan, Professor of Biostatistics, Boston University School of Public Health. “Confidence Intervals”. [Online]. Available: https://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_Confidence_Intervals/BS704_Confidence_Intervals_print.html [Accessed: 23 June 2020]