



UNIVERSITAS INDONESIA

**RANCANG BANGUN PLATFORM SEBAGAI SISTEM
PEMANTAUAN UNTUK PEMODELAN PEMBANGKIT
LISTRIK VIRTUAL**

SKRIPSI

Muhammad Djati Pradana

1606829680

**FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KOMPUTER**

**DEPOK
JULI 2020**



UNIVERSITAS INDONESIA

**RANCANG BANGUN PLATFORM SEBAGAI SISTEM
PEMANTAUAN UNTUK PEMODELAN PEMBANGKIT
LISTRIK VIRTUAL**

SKRIPSI

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

**Muhammad Djati Pradana
1606829680**

**FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KOMPUTER**


**DEPOK
JULI 2020**

HALAMAN PERNYATAAN ORISINALITAS

**Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.**

Nama : Muhammad Djati Pradana

NPM : 1606829280

Tanda Tangan : 

Tanggal : 30 Juli 2020

HALAMAN PENGESAHAN

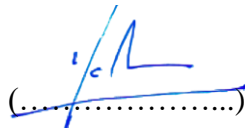
Skripsi ini diajukan oleh:

Nama : Muhammad Djati Pradana
NPM : 1606829680
Program Studi : Teknik Komputer
Judul Skripsi : Rancang Bangun Platform sebagai Sistem Pemantauan
untuk Pemodelan Pembangkit Listrik Virtual

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada program Studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Dr.-Ing. Eko Adhi Setiawan S.T., M.T.



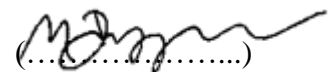
(.....)

Penguji : F. Astha Ekadiyanto, S.T., M.Sc.



(.....)

Penguji : Dr. Muhammad Suryanegara, S.T., M.Sc.



(.....)

Ditetapkan di : Depok

Tanggal : 30 Juli 2020

KATA PENGANTAR

Segala puji ke hadirat Allah SWT karena berkat rahmat dan rida-Nya, penulis dapat menyelesaikan skripsi yang berjudul “Rancang Bangun Platform sebagai Sistem Pemantauan untuk Pemodelan Pembangkit Listrik Virtual”. Skripsi ini merupakan syarat wajib bagi mahasiswa sarjana jurusan Teknik Komputer Universitas Indonesia untuk menyelesaikan studinya.

Penulis sadar bahwa dalam menyelesaikan skripsi ini tak lepas dari dukungan dan bantuan banyak pihak. Oleh karena itu, izinkan penulis mengucapkan terima kasih kepada:

1. Bapak Dr.-Ing. Eko Adhi Setiawan S.T., M.T. dan F. Astha Ekadiyanto S.T., M.Sc. selaku dosen pembimbing yang telah memberikan bantuan dalam bentuk waktu, tenaga dan pikiran perihal pembuatan skripsi ini.
2. Bapak Prof. Dr.-Ing. Ir. Kalamullah Ramli, M.Eng. selaku pembimbing akademis.

Di akhir kata penulis mengharapkan saran yang membangun karena skripsi ini masih membutuhkan pengembangan lebih lanjut. Penulis berharap dengan adanya skripsi ini dapat menjadi ilmu pengetahuan bagi orang lain yang menerapkannya.

Depok, 30 Juli 2020



Muhammad Djati Pradana

**HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI KARYA
ILMIAH UNTUK KEPENTINGAN AKADEMIS**

Sebagai sivitas akademik Universitas Indonesia, saya yang bertandatangan di bawah ini:

Nama : Muhammad Djati Pradana
NPM : 1606829680
Program Studi : Teknik Komputer
Fakultas : Teknik
Jenis Karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty-Free Right)** atas karya ilmiah saya yang berjudul:

**RANCANG BANGUN PLATFORM SEBAGAI SISTEM PEMANTAUAN
UNTUK PEMODELAN PEMBANGKIT LISTRIK VIRTUAL**

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 30 Juli 2020
Yang menyatakan



(Muhammad Djati Pradana)

ABSTRAK

Nama : Muhammad Djati Pradana
Program Studi : Teknik Komputer
Judul : Rancang Bangun Platform sebagai Sistem Pemantauan untuk
Pemodelan Pembangkit Listrik Virtual

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 menunjukkan 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 listrik virtual yaitu 3,93 dalam skala 5.

Kata Kunci: Energi listrik, Pembangkit listrik virtual, Pemantauan, Pemodelan.

ABSTRACT

Name : Muhammad Djati Pradana

Study Program : Computer Engineering

Title : Design a Platform as Monitoring System for Virtual Power Plant Modeling

Technological development change the centralized electricity supply system to decentralized using virtual power plant. The concept of virtual power plant uses power plant with distributed renewable energy that allows client as prosumer can be connected to the grid. This thesis aims to designing platform as monitoring system for virtual power plant modeling. Monitoring system allows client can monitor activity of generator and load include the power is generated from each generator, the capacity of battery, the power of load and the energy total is obtained from accumulation of power per day in the load and generator. This modeling uses 3 clients consisting of 2 clients having a generator and load while 1 client having a battery. System testing is performed on the usage of node editor and usability testing. Results of the testing show that the usage of node editor can support configuration in the platform that has been modeled and the assessment of platform as monitoring system for virtual power plant modeling by respondents can be categorized quite well with the Mean of Survey based on respondents understanding virtual power plant that is 4.1 in a scale of 5 while for respondents not understanding the virtual power plant is 3.93 in a scale of 5.

Keywords: Electrical energy, Virtual power plant, Monitoring, Modeling.

DAFTAR ISI

HALAMAN PERNYATAAN ORISINALITAS.....	ii
HALAMAN PENGESAHAN.....	iii
KATA PENGANTAR	iv
HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS	v
ABSTRAK.....	vi
ABSTRACT.....	vii
DAFTAR ISI.....	viii
DAFTAR GAMBAR	xi
DAFTAR TABEL.....	xiii
DAFTAR LAMPIRAN.....	xiv
DAFTAR SINGKATAN	xv
BAB 1 PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Tujuan Penelitian	2
1.3. Batasan Penelitian.....	3
1.4. Metodologi Penelitian.....	3
1.5. Sistematika Penulisan	4
BAB 2 LANDASAN TEORI.....	6
2.1. Pembangkit Listrik Virtual	6
2.2. Panel Surya	9
2.3. Sel Bahan Bakar.....	11
2.4. Baterai	12
2.5. MQTT	13
2.6. Node.js	15
2.7. AngularJS.....	16
2.8. Rete.js.....	17
2.9. TimescaleDB.....	18
2.10. GraphQL	19
2.11. Hasura	19

BAB 3 PERANCANGAN SISTEM	21
3.1. Pemodelan Konsep Pembangkit Listrik Virtual	21
3.2. Pengembangan Platform untuk Pemodelan Pembangkit Listrik Virtual	23
3.3. Perancangan <i>Node Editor</i> pada Platform untuk Pemodelan Pembangkit Listrik Virtual.....	27
3.3.1. <i>Node</i> MQTT (<i>Subscriber</i> dan <i>Publisher</i>)	28
3.3.2. <i>Node</i> Kalkulasi (<i>Addition</i> dan <i>Multiplication</i>)	29
3.3.3. <i>Node Database</i>	30
3.3.4. <i>Node</i> Akumulator (<i>Load</i> dan <i>Generator</i>)	32
3.3.5. <i>Node</i> Spesifikasi (<i>Battery</i> dan <i>Fuel cell</i>).....	33
3.3.6. <i>Node</i> Kontrol	34
3.4. <i>Use Case</i> Diagram pada Platform untuk Pemodelan Pembangkit Listrik Virtual	35
3.5. Diagram Alir pada Platform untuk Pemodelan Pembangkit Listrik Virtual	36
BAB 4 PEMODELAN, PENGUJIAN DAN ANALISIS SISTEM.....	38
4.1. Pemodelan Sistem.....	38
4.1.1. Pemodelan Klien Pertama (<i>Photovoltaic</i>)	39
4.1.2. Pemodelan Klien Kedua (<i>Fuel Cell</i>)	44
4.1.3. Pemodelan Klien Ketiga (Baterai).....	47
4.2. Pengujian Sistem.....	51
4.2.1. Pengujian <i>Node Editor</i>	51
4.2.2. <i>Usability Testing</i>	52
4.3. Hasil dan Analisis Pengujian Sistem	53
4.3.1. Hasil dan Analisis Pengujian <i>Node Editor</i>	53
4.3.2. Hasil dan Analisis <i>Usability Testing</i>	56
4.3.2.1. Responden Memahami Pembangkit Listrik Virtual	57
4.3.2.2. Responden Tidak Memahami Pembangkit Listrik Virtual	62
4.3.2.3. Hasil dan Analisis Keseluruhan <i>Usability Testing</i>	67
BAB 5 PENUTUP	69
5.1. Kesimpulan	69
5.2. Saran untuk Pengembangan Selanjutnya	70

DAFTAR PUSTAKA	71
LAMPIRAN.....	74

DAFTAR GAMBAR

Gambar 2.1. Skema Umum Sistem Pembangkit Listrik Virtual.....	7
Gambar 2.2. Pembangkit Listrik Virtual Sentralisasi	8
Gambar 2.3. Pembangkit Listrik Virtual Desentralisasi	9
Gambar 2.4. Proses Konversi Energi Matahari ke Listrik pada Panel Surya	10
Gambar 2.5. Grafik Radiasi Terhadap Waktu.....	11
Gambar 2.6. Proses pada <i>Fuel Cell</i>	12
Gambar 2.7. Konsep Dasar MQTT	14
Gambar 2.8. Tampilan <i>Node Editor</i> pada <i>Rete.js</i>	18
Gambar 2.9. Arsitektur TimescaleDB.....	18
Gambar 3.1. Pemodelan Konsep Pembangkit Listrik Virtual.....	21
Gambar 3.2. Perancangan Sistem pada <i>Sofwan House</i> TREC FTUI.....	22
Gambar 3.3. Arsitektur Platform untuk Pemodelan Pembangkit Listrik Virtual..	24
Gambar 3.4. Halaman Login pada Platform	25
Gambar 3.5. Halaman Utama pada Platform	25
Gambar 3.6. Halaman <i>Node Editor</i> pada Platform	27
Gambar 3.7. <i>Node MQTT Client Subscriber dan Publisher</i>	28
Gambar 3.8. <i>Node MQTT Addition dan Multiplication</i>	29
Gambar 3.9. <i>Node MQTT Database</i>	30
Gambar 3.10. <i>Node Generator dan Load Accumulator</i>	32
Gambar 3.11. <i>Node Battery and FuelCell Specification</i>	33
Gambar 3.12. <i>Node Control</i>	34
Gambar 3.13. <i>Use Case Diagram</i> pada Platform	35
Gambar 3.14. Diagram alir pada Platform	36
Gambar 4.1. Tampilan Konfigurasi <i>Node</i> pada Klien Pertama	40
Gambar 4.2. Tampilan Potongan JSON pada Klien Pertama	41
Gambar 4.3. Tampilan <i>Hypertable</i> pada Klien Pertama	42
Gambar 4.4. Daftar Tabel <i>Chunk</i> pada Klien Pertama.....	42
Gambar 4.5. Tampilan Tabel <i>Photovoltaic</i> pada Klien Pertama	43
Gambar 4.6. Tampilan Grafik <i>Power</i> pada <i>Photovoltaic</i> (Klien Pertama)	43
Gambar 4.7. Tampilan Konfigurasi <i>Node</i> pada Klien Kedua	44

Gambar 4.8. Tampilan Potongan JSON pada Klien Kedua	45
Gambar 4.9. Tampilan <i>Hypertable</i> pada Klien Kedua.....	46
Gambar 4.10. Daftar Tabel <i>Chunk</i> pada Klien Kedua	46
Gambar 4.11. Tampilan Tabel <i>Fuel Cell</i> pada Klien Kedua.....	47
Gambar 4.12. Tampilan Grafik <i>Power</i> pada <i>Fuel Cell</i> (Klien Kedua)	47
Gambar 4.13. Tampilan Konfigurasi <i>Node</i> pada Klien Ketiga.....	48
Gambar 4.14. Tampilan Potongan JSON pada Klien Ketiga.....	49
Gambar 4.15. Tampilan <i>Hypertable</i> pada Klien Ketiga	49
Gambar 4.16. Daftar Tabel <i>Chunk</i> pada <i>Battery Generation</i>	50
Gambar 4.17. Tampilan Tabel <i>Battery Generation</i> pada Klien Ketiga	50
Gambar 4.18. Tampilan Grafik <i>Power</i> pada <i>Battery Generation</i> (Klien Ketiga). 51	
Gambar 4.19. Tampilan Menu pada <i>Node Editor</i>	53
Gambar 4.20. Fitur <i>Delete</i> dan <i>Clone</i> pada <i>Node Editor</i>	54
Gambar 4.21. Fitur <i>Search</i> pada <i>Node Editor</i>	54
Gambar 4.22. <i>Node</i> Dibesarkan atau Dikecilkan	55
Gambar 4.23. <i>Node</i> Dihubungkan dengan <i>Node</i> Lain	55
Gambar 4.24. Grafik Data Hasil Kuesioner untuk Penilaian Berdasarkan Responden Memahami Pembangkit Listrik Virtual	58
Gambar 4.25. Grafik Data Hasil Kuesioner untuk Penilaian Berdasarkan Responden Tidak Memahami Pembangkit Listrik Virtual	63

DAFTAR TABEL

Tabel 4.1. Pengujian <i>Node Editor</i>	55
Tabel 4.2. <i>Confidence Interval</i> Berdasarkan Responden Memahami Pembangkit Listrik Virtual.....	61
Tabel 4.3. <i>Confidence Interval</i> Berdasarkan Responden Tidak Memahami Pembangkit Listrik Virtual.....	66

DAFTAR LAMPIRAN

Lampiran 1. Tampilan Kuesioner Penilaian terhadap Platform untuk Pemodelan Pembangkit Listrik Virtual.....	74
Lampiran 2. Tabel <i>T-Distribution</i>	75
Lampiran 3. Data Hasil Kuesioner Berdasarkan Responden Memahami Konsep Pembangkit Listrik Virtual.....	76
Lampiran 4. Data Hasil Kuesioner Berdasarkan Responden Tidak Memahami Konsep Pembangkit Listrik Virtual	76

DAFTAR SINGKATAN

API	: <i>Application Programming Interface</i>
ESS	: <i>Energy Storage System</i>
JSON	: <i>Javascript Object Notation</i>
MoS	: <i>Mean of Survey</i>
MQTT	: <i>Message Queuing Telemetry Transport</i>
MVC	: <i>Model View Controller</i>
REST	: <i>Representational State Transfer</i>
SQL	: <i>Standard Query Language</i>
VPP	: <i>Virtual Power Plant</i>

BAB 1

PENDAHULUAN

Bab ini akan membahas mengenai latar belakang penelitian, tujuan penelitian, batasan penelitian, metodologi penelitian, dan sistematika penulisan.

1.1. Latar Belakang

Dalam memberikan pasokan energi listrik yang sesuai permintaan konsumen (optimal) maka diperlukan waktu yang tidak sedikit untuk membangun suatu pembangkit tenaga listrik. Para perencana sistem juga harus dapat melihat kemungkinan perkembangan sistem tenaga listrik di tahun-tahun yang akan datang. Maka dari itu, diperlukan pengembangan industri listrik yang meliputi perencanaan pembangkit, sistem pemantauan dan kendali, serta sistem transmisi dan distribusi listrik yang akan disalurkan hingga sampai pada konsumen. 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. Dengan kata lain pembangunan bidang kelistrikan harus dapat mengimbangi kebutuhan tenaga listrik yang akan terus meningkat tiap tahunnya. Pembangkit listrik yang dimiliki oleh PLN secara umum menggunakan energi yang termasuk tidak terbarukan seperti batubara dan bahan bakar fosil lainnya. Untuk dapat memenuhi kebutuhan energi yang terus meningkat maka diperlukan pembangkit tenaga listrik dengan memanfaatkan energi yang ramah lingkungan atau terbarukan (*renewable energy*) seperti cahaya matahari, angin, air dan hidrogen.

Namun, sampai saat ini sistem pengelolaan energi listrik hanya dilakukan oleh PLN yang memberikan pasokan listrik secara satu arah dari pembangkit konvensional yang diteruskan ke rumah-rumah sehingga penyediaan listrik tidak optimal untuk memenuhi permintaan konsumen. Perkembangan teknologi menghasilkan sebuah sistem penyediaan listrik desentralisasi dengan menggunakan pembangkit listrik virtual menjadi sebuah konsep terbaru untuk penyediaan listrik. Penggunaan teknologi ini, juga dapat dipadukan dengan pasokan energi dari PLN.

Dengan konsep pembangkit listrik virtual ini, pengelolaan listrik dapat berjalan dua arah yang 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*). Konsep pembangkit listrik virtual dapat meminimalkan penggunaan energi listrik dari PLN serta membantu PLN dalam mengatasi pemadaman bergilir karena terdapat gangguan distribusi. Selain itu, juga dapat mengurangi beban puncak (*peak load*) terhadap kebutuhan energi listrik baik di rumah tangga maupun industri serta mengurangi polusi akibat sisa pembakaran dengan menggunakan energi terbarukan (*renewable energy*). Pengelolaan pembangkit listrik virtual ini sendiri memang lebih diperuntukkan pada suatu area atau kawasan yang mempunyai pembangkit energi listrik sendiri seperti yang biasanya ditemukan di negara maju seperti Jerman. Pembangkit listrik virtual juga dapat mengetahui dan mengendalikan pengeluaran atas energi listrik yang dihasilkan oleh setiap pembangkit.

Konsep pembangkit listrik virtual membutuhkan sistem pemantauan untuk dapat mengetahui aktivitas pembangkit dalam memproduksi energi serta aktivitas beban pada jaringan listrik sesuai dengan algoritma yang telah didefinisikan sehingga diperlukan platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual. Aktivitas yang dapat dipantau oleh klien meliputi daya 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.

1.2. Tujuan Penelitian

Penulis melaksanakan penelitian ini dengan maksud untuk memenuhi syarat kelulusan mata kuliah skripsi dalam jenjang pendidikan yang sedang diambil oleh penulis dan sebagai syarat wajib dalam memperoleh gelar Sarjana Teknik. Tujuan dari penelitian ini adalah sebagai berikut:

1. Membuat *node editor* yang dapat membantu pengguna dalam melakukan konfigurasi yaitu *interfacing* antara platform dengan pembangkit menggunakan protokol MQTT, melakukan kalkulasi, menyimpan data hasil pemantauan serta menghitung total energi pada pembangkit dan beban.

- Membuat sistem pemantauan yang dapat membantu klien dalam mengetahui daya yang dihasilkan dari setiap pembangkit, kapasitas baterai, daya pada beban serta total energi yang dihasilkan dari akumulasi daya per hari pada beban dan pembangkit.

1.3. Batasan Penelitian

Dalam penelitian ini dibatasi dengan perancangan platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual yang digunakan untuk mengoptimasi aktivitas pembangkit dalam memproduksi energi.

Batasan masalah yang akan dibahas dalam skripsi meliputi pembuatan *node editor* untuk mempermudah konfigurasi, protokol yang digunakan sebagai komunikasi dalam pertukaran data, *database* yang digunakan untuk menyimpan rekaman data berdasarkan *timeseries*, parameter yang dapat dicatat dari setiap klien yang terlibat dalam pemodelan yang dibuat.

1.4. Metodologi Penelitian

Dalam penulisan sebuah laporan, penulis menggunakan beberapa metode yang diterapkan dalam penulisan ini. Metode penelitian yang digunakan adalah:

1. Metode Observasi

Metode observasi merupakan metode yang dilakukan untuk menemukan dan mengetahui bagian ataupun komponen yang perlu dikembangkan dalam sistem.

2. Mentoring

Mentoring merupakan sesi bimbingan oleh dosen pembimbing kepada penulis. Penulis diberikan bimbingan dalam bentuk tanya-jawab dan diskusi dengan dosen pembimbing.

3. Metode Studi Literatur

Metode studi literatur dilakukan oleh penulis dengan cara membaca buku manual operasional, karya tulis (*paper*), skripsi dan tesis yang berkaitan dengan topik, buku pendukung yang tersedia pada perpustakaan dan internet. Metode ini digunakan untuk mencari referensi teori yang relevan dengan topik yang dibahas.

4. Perancangan

Pembuatan *node editor* dan sistem pemantauan yang mendukung platform pemodelan untuk pembangkit listrik virtual.

5. Pemodelan

Perancangan yang telah dibuat akan dimodelkan untuk melihat apakah pemodelan dalam bentuk platform dapat berjalan dengan baik sesuai fungsinya.

6. Pengujian

Platform pemodelan yang telah dibuat akan diuji melalui *usability testing* dengan memberikan kuesioner mengenai penilaian pengguna terhadap penggunaan platform pemodelan pembangkit listrik virtual. Selain itu, juga dilakukan pengujian terhadap *node editor*.

7. Analisis

Analisis dilakukan terhadap hasil *usability testing* untuk mengetahui penilaian rata-rata responden terhadap platform pemodelan dan mengukur keakuratan data sampel apakah dapat mewakili populasi data sesungguhnya. Selain itu, juga dilakukan analisis terhadap hasil pengujian *node editor*.

8. Kesimpulan

Kesimpulan akan ditarik dari hasil perancangan, pemodelan, pengujian dan analisis.

1.5. Sistematika Penulisan

Sistematika dari penulisan skripsi ini dibagi menjadi lima bab dengan struktur sebagai berikut.

Bab 1 merupakan pendahuluan yang menjelaskan mengenai latar belakang penelitian, tujuan penelitian, batasan penelitian, metodologi penelitian, dan sistematika penulisan. Dasar teori yang berhubungan dengan pembangkit listrik virtual, karakteristik dari pembangkit *Photovoltaic* (PV) dan *Fuel cell*, perangkat lunak yang digunakan dalam perancangan platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual, *framework* yang digunakan dalam pembuatan *node editor*, protokol komunikasi data yang digunakan dan database yang digunakan kemudian dibahas pada Bab 2.

Bab selanjutnya (Bab 3) merupakan kontribusi utama dalam pembuatan skripsi ini yang berisi tentang pemodelan konsep pembangkit listrik virtual, pengembangan platform sebagai model sistem pemantauan untuk pemodelan pembangkit listrik virtual, perancangan *node editor* untuk mendukung sistem komunikasi pada pembangkit listrik virtual, *use case* diagram dan diagram alir pada platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual. Perancangan sistem yang telah dibuat kemudian akan dimodelkan pada Bab 4. Selain itu, Bab 4 juga merupakan kontribusi utama yang akan menjelaskan mengenai pengujian yang dilakukan pada platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual yang telah dimodelkan serta analisis terhadap hasil pengujian yang telah dilakukan.

Sebagai penutup, seluruh hasil penelitian ini disimpulkan secara ringkas pada Bab 5. Kesimpulan tersebut dapat berisi sesuai dengan apa yang dikerjakan dalam skripsi dimulai dari perancangan, pemodelan, pengujian dan analisis sistem yang telah dilakukan serta memberikan saran untuk pengembangan lebih lanjut yang dapat dilakukan pada platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual yang telah dimodelkan.

BAB 2

LANDASAN TEORI

Bab ini akan membahas tentang dasar teori yang berhubungan dengan pembangkit listrik virtual, karakteristik dari pembangkit *Photovoltaic* (PV) dan *Fuel cell*, perangkat lunak yang digunakan dalam perancangan platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual, *framework* yang digunakan dalam pembuatan *node editor*, protokol komunikasi data yang digunakan dan database yang digunakan.

2.1. 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]. Berdasarkan hal tersebut, dapat dikatakan bahwa pembangkit listrik virtual merupakan sebuah konsep untuk mengoptimasi produksi energi dari beberapa pembangkit dengan berbagai macam sumber energi yang terdistribusi dan hasilnya akan dikumpulkan serta diagregasi sehingga menjadi satu entitas. Sehingga energi dari entitas tersebut dapat diekspor kepada *grid* untuk meminimalkan penggunaan energi listrik dari PLN dan mengurangi beban puncak di *grid* saat penggunaan pada waktu tertentu. Selain itu, energi listrik yang dihasilkan dari pembangkit listrik gabungan tersebut dan konsumsi daya pada setiap unit pada pembangkit listrik virtual dapat diperdagangkan untuk pertukaran energi dengan energi listrik sebagai barang atau komoditas utama yang diperdagangkan. Komponen utama dalam pembangkit listrik virtual yaitu [1]:

1. Pembangkit listrik yang terdistribusi (*Distributed Generation*)

Sumber energi yang digunakan dalam pembangkit listrik yang terdistribusi dapat berasal dari energi terbarukan seperti *photovoltaic*, *fuel cell*,

geothermal, turbin energi, *hydro station*, biogas, *micro gas turbine* dan *wind turbine*.

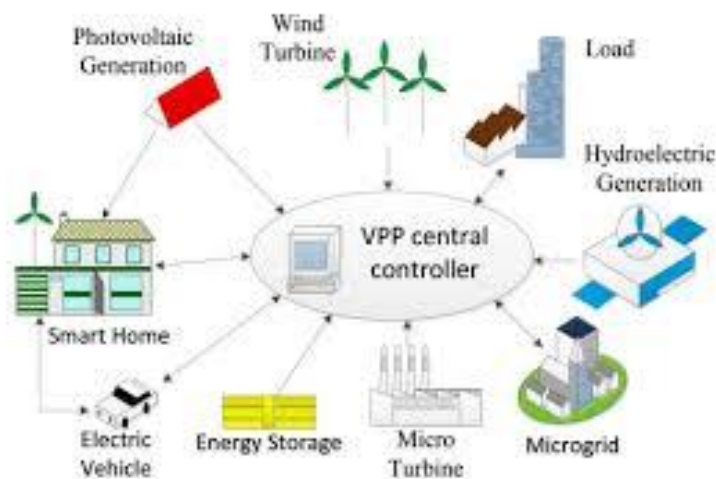
2. Baterai (*Energy Storage Sistem*)

Penggunaan baterai sebagai alat penyimpanan energi listrik yang sewaktu-waktu dapat digunakan untuk melepaskan atau mengeksport energi listrik.

3. Teknologi komunikasi dan informasi

Infrastruktur pada pembangkit listrik virtual berupa teknologi komunikasi dan informasi merupakan hal penting untuk menghubungkan antar unit yang terdapat dalam entitas pembangkit listrik virtual. Komunikasi sendiri dibagi menjadi tiga yaitu *Real-time Operational Communication*, *Administrative Operational Communication* dan *Administrative Communication*. Teknologi media komunikasi yang dipertimbangkan dalam pembangkit listrik virtual yaitu *Energy Management Sistem (EMS)*, *Supervisory Control and Data Acquisition (SCADA)* dan *Distribution Dispatching Centre Project*.

Untuk lebih memperjelas mengenai pembangkit listrik virtual dapat dilihat pada Gambar 2.1.



Gambar 2.1. Skema Umum Sistem Pembangkit Listrik Virtual [2]

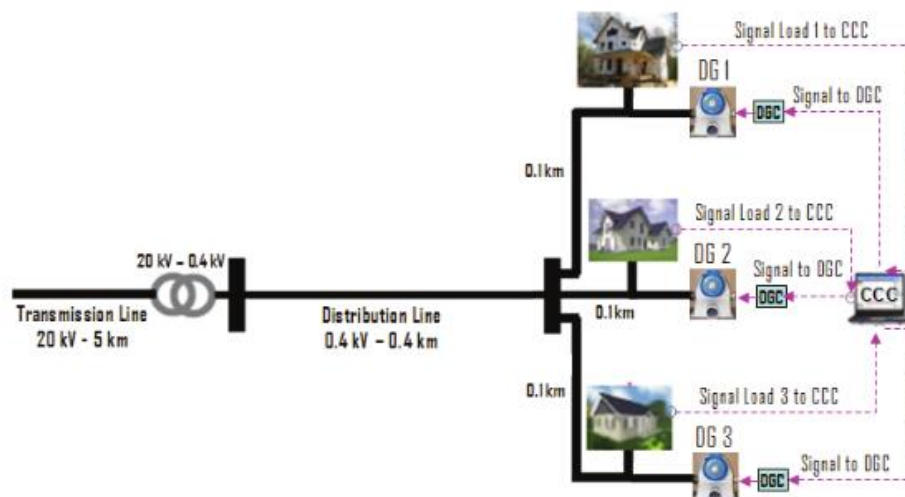
Hal penting dari pembangkit listrik virtual adalah adanya pusat kendali yang akan mengatur pembangkit dengan berbagai macam sumber energi yang terlibat dalam pembangkit listrik virtual [3]. Pusat kendali ini akan mengatur pengoperasian

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 menyalakan barang-barang elektronik berdasarkan penjadwalan otomatis dan lain-lain. Pusat kontrol agregasi dapat berupa sistem sentralisasi dan desentralisasi yang didukung oleh algoritma kontrol dan infrastruktur komunikasi yang kemudian dapat dianggap sebagai pembangkit listrik tunggal yang besar [1].

Pembangkit listrik virtual dibagi menjadi dua kategori berdasarkan topologi kontrol yaitu [1]:

1. Pembangkit Listrik Virtual Sentralisasi

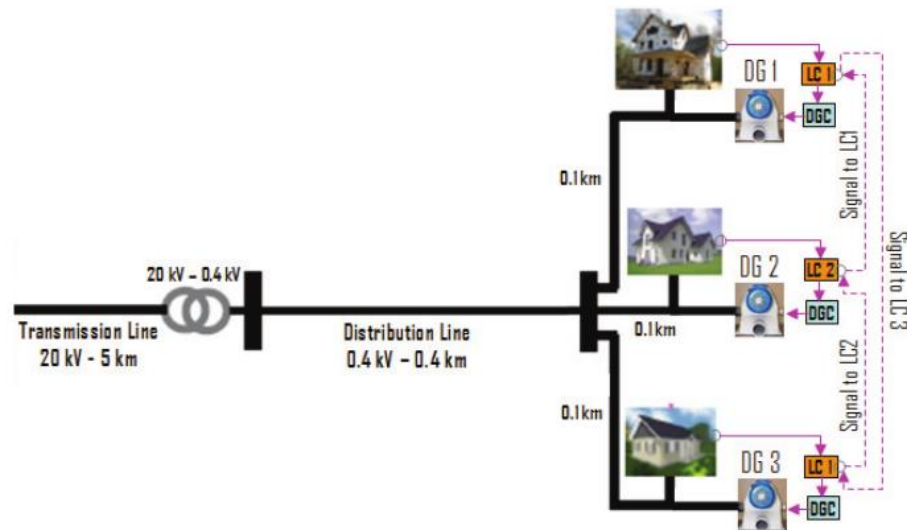
Pada kategori ini, pembangkit listrik yang terdistribusi dikontrol oleh *Control Coordination Centre (CCC)* yang berlokasi tepat pada pusat unit pembangkit listrik terdistribusi. Sinyal beban yang tersedia akan ditransmisikan ke CCC dan diproses oleh algoritma. Setelah itu, sinyal akan dikirim ke masing-masing pengontrol pembangkit terdistribusi (DGC) kemudian keluaran daya aktif yang diproduksi sesuai sinyal yang diberikan oleh CCC terlihat pada Gambar 2.2. Dengan menggunakan CCC maka dapat menjalankan fungsi teknis dan ekonomis untuk mendapatkan manfaat dari agregasi yang terdistribusi.



Gambar 2.2. Pembangkit Listrik Virtual Sentralisasi [1]

2. Pembangkit Listrik Virtual Desentralisasi

Pada topologi desentralisasi, setiap unit pembangkit listrik terdistribusi dikontrol oleh *Local Controller* (LC). Keluaran daya aktif dari pembangkit listrik terdistribusi yang dikontrol oleh pengontrol pembangkit terdistribusi (DGC) dan DGC dikontrol oleh LC yang diproses oleh algoritma. Untuk menunjukkan integrasi sistem tersebut dapat dilihat pada Gambar 2.3.



Gambar 2.3. Pembangkit Listrik Virtual Desentralisasi [1]

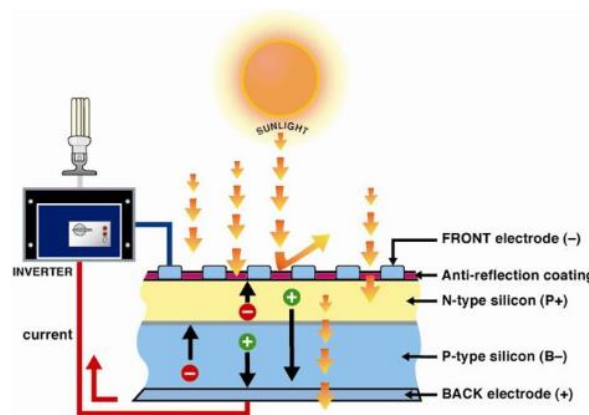
Penggunaan sistem pembangkit listrik virtual ini secara otomatis juga akan merubah proses penyediaan energi listrik dari yang semula hanya satu arah (dari PLN menuju ke konsumen) menjadi dua arah. Pada konsep dua arah, konsumen juga dapat berperan sebagai penghasil energi listrik atau yang disebut sebagai prosumer (*producer-consumer*). Sistem pembangkit listrik virtual juga dapat membantu memenuhi pasokan listrik dari sumber listrik utama sehingga dapat menjaga ketersediaan listrik bagi konsumen [4]. Selain itu, adanya pembangkit listrik virtual ini juga dapat menjadi suatu ide bisnis baru yang mungkin dapat muncul dalam waktu dekat dengan menerapkan ekspor dan impor dengan barang berupa listrik.

2.2. Panel Surya

Panel Surya merupakan alat konversi energi matahari atau radiasi menjadi energi listrik. Pada panel surya menggunakan *photovoltaic* yang terbuat dari bahan

semikonduktor untuk menghasilkan energi listrik. *Photovoltaic* akan tetap menghasilkan energi selama terpapar oleh cahaya matahari sedangkan *photovoltaic* akan berhenti menghasilkan energi karena tidak lagi terpapar cahaya matahari [5].

Prinsip kerja dari panel surya menggunakan dua bahan semikonduktor untuk menghasilkan energi. Bahan semikonduktir yang digunakan adalah bertipe p dan n. Ketika sinar matahari yang merupakan foton-foton menyinari panel surya terdapat tiga kemungkinan yang mungkin terjadi, yaitu akan diserap, dipantulkan ataupun akan dilewatkan. Sebuah foton harus memiliki tingkat energi tertentu untuk dapat melepaskan electron dan menghasilkan listrik. Batas tingkat energi yang dibutuhkan untuk melepaskan suatu electron dari atom disebut dengan *band-gap*. Jika tingkat energi foton kurang dari *band-gap* maka electron tidak dapat lepas, sebaliknya jika tingkat energi nya jauh lebih besar dibandingkan *band-gap* maka kelebihan energi tersebut akan berubah menjadi panas. Gambaran proses konversi energi matahari menjadi listrik pada panel surya dapat dilihat pada Gambar 2.4.

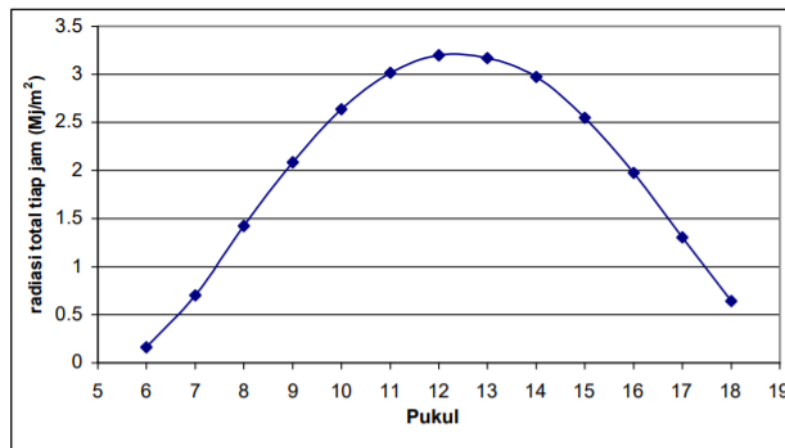


Gambar 2.4. Proses Konversi Energi Matahari ke Listrik pada Panel Surya [6]

Kapasitas daya pada panel surya sendiri dilambangkan dengan menggunakan satuan *Watt peak* (W_p). *Watt peak* adalah satuan yang digunakan untuk mengukur kapasitas daya dari panel surya. *Watt peak* merupakan daya tertinggi yang dapat dihasilkan oleh suatu panel surya dalam waktu tertentu. Daya pada panel surya dipengaruhi oleh beberapa faktor diantaranya yaitu efisiensi, area total dari panel surya, radiasi dan suhu [7].

Panel surya memanfaatkan cahaya matahari namun matahari tidak terus menerangi suatu daerah sepanjang waktu sehingga energi yang dihasilkan

bergantung pada intensitas dari cahaya matahari. Hal ini menunjukkan bahwa panel surya memiliki waktu-waktu aktif dalam memproduksi energi yang dapat dilihat pada Gambar 2.5.



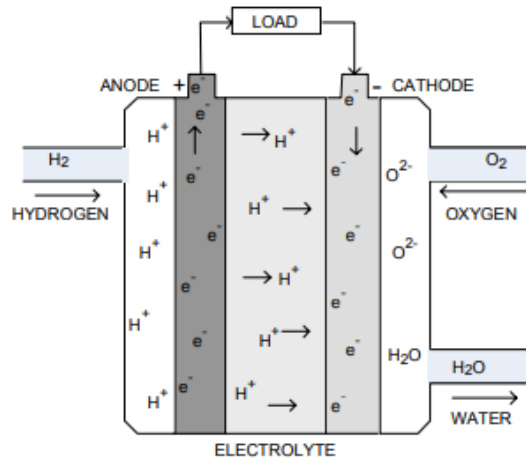
Gambar 2.5. Grafik Radiasi Terhadap Waktu [8]

2.3. Sel Bahan Bakar

Sel bahan bakar (*Fuel cell*) merupakan sebuah alat konversi energi elektrokimia yang berfungsi untuk menghasilkan energi listrik. *Fuel cell* sendiri memiliki fitur hampir sama dengan baterai yang memiliki dua elektroda (anoda dan katoda), elektrolit dan katalis untuk elektrolisis dan reaksi kimia untuk menghasilkan energi listrik. Sehingga perbedaannya dengan baterai yaitu *fuel cell* dirancang untuk dapat terus menghasilkan energi listrik memanfaatkan penyediaan bahan bakar hidrogen dan oksigen dari luar sedangkan baterai hanya menggunakan elektrolit yang terdapat didalamnya untuk menyimpan energi listrik [9].

Fuel cell memiliki tiga komponen utama yaitu *fuel reformer* atau *processor*, *power section* yang terdiri dari tumpukan *fuel cell*, dan *power conditioner* yang terdiri dari *DC converter*. Dalam proses menghasilkan energi listrik, hidrogen akan masuk ke dalam *fuel cell*. Kemudian elektron bebas akan meninggalkan molekul hidrogen pada anoda melalui jalur luar. Ketika elektron bebas telah meninggalkan hidrogen, ion hidrogen akan berpindah ke katoda. Di katoda, terjadi proses kimia yang mana oksigen dari udara, ion hidrogen dan elektron bebas akan bergabung membentuk air dan panas (energi listrik). Pada proses kimia tersebut, memiliki

persamaan pada anoda ($\text{H}_2 = 2\text{H}^+ + 2\text{e}^-$) dan pada katoda ($\frac{1}{2}\text{O}_2 + 2\text{H}^+ + 2\text{e}^- = \text{H}_2\text{O}$). Untuk lebih memperjelas, proses kimia pada *fuel cell* terlihat pada Gambar 2.6.



Gambar 2.6. Proses pada *Fuel Cell* [9]

Karena energi yang diproduksi *fuel cell* merupakan reaksi kimia pembentukan air maka alat ini tidak akan menghasilkan efek samping yang berbahaya bagi lingkungan. Penentuan daya yang dihasilkan oleh *fuel cell* dapat dipengaruhi oleh beberapa parameter yaitu banyaknya tumpukan sel, tegangan dari setiap tumpukan sel dan arus pada setiap sel [10].

2.4. Baterai

Pada pembangkit listrik virtual, baterai biasanya disebut dengan ESS (*Energy Storage System*) yang digunakan sebagai alat penyimpanan energi yang bersifat pasif (hanya dapat diisi kembali atau tidak dapat menghasilkan energi dengan sendirinya). Setiap klien yang terlibat biasanya memiliki baterai dengan total kapasitas daya yang bervariasi. Baterai akan menyimpan energi ketika daya tersisa atau tersedia pada *grid*, maka dengan kata lain pembangkit menghasilkan daya lebih banyak dari total beban yang ada pada *grid* sedangkan baterai akan melakukan *discharge* ketika daya tidak mencukupi pada *grid* atau pembangkit menghasilkan daya lebih sedikit dari total beban yang ada pada *grid* [11].

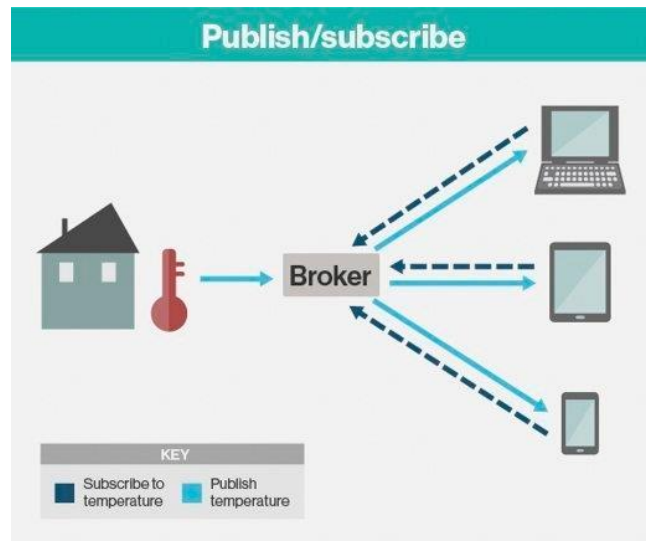
Baterai memiliki komponen utama yaitu elektroda dengan kutub positif dan negatif, elektrolit dan bahan isolasi (pemisah). Pada baterai, anoda dan katoda terbuat dari bahan yang dapat bereaksi dengan bahan elektrolitnya. Dalam proses

mengalirkan energi listrik, anoda dan elektrolit akan bereaksi dengan membentuk satu senyawa baru yang menyisakan satu elektron. Sebaliknya, reaksi antara katoda dan elektrolit membutuhkan satu elektron. Sehingga sisa elektron dari reaksi anoda dan elektrolit akan dikirimkan ke katoda agar katoda dapat bereaksi dengan elektrolit. Perpindahan elektron inilah yang dapat menimbulkan aliran listrik dari sebuah baterai. Dikarenakan baterai tidak dapat menghasilkan energi, namun hanya dapat mengisi kembali energi dengan sumber energi yang berasal dari pembangkit lain sehingga pada baterai sendiri memiliki siklus *charge* dan *discharge* energi.

2.5. MQTT

Salah satu pendukung perkembangan teknologi IoT adalah munculnya protokol MQTT (*Message Queuing Telemetry Transport*), yang dirilis dengan lisensi Royalty Free pada 2010, yang pada awalnya bersifat proprietary. MQTT ditemukan oleh Andy Stanford – Clark (IBM) dan Arlen Nipper di tahun 1999, yang semula dibuat untuk menghubungkan sistem telemetri jalur pipa minyak melalui satelit [12]. MQTT merupakan protokol komunikasi *publish* dan *subscribe* berdasarkan topik yang sangat sederhana dan ringan, yang didesain untuk alat yang memiliki kemampuan terbatas dengan *bandwidth* yang rendah dan latensi yang tinggi atau jaringan yang kurang dapat diandalkan. MQTT berjalan pada protokol TCP yang berada pada *application layer*. Protokol MQTT hanya mengirimkan paket dengan ukuran yang kecil sehingga tidak membebani *resource* jaringan internet, karena sifat dari protokol MQTT yang sederhana dan efisien [13].

Prinsip dari desain ini adalah untuk meminimalkan penggunaan *bandwidth* jaringan dan kebutuhan sumber daya pada perangkat serta pada waktu yang sama juga berusaha untuk memastikan keandalan dan kepastian dari pengiriman data. Prinsip yang ada ini juga memunculkan beberapa ide protokol mengenai “*machine-to-machine*” (M2M) atau IoT yang menginginkan perangkat di dunia untuk saling terhubung dan bertukar data [14]. Contoh broker MQTT adalah HiveMQ dan Mosquitto. Dalam mengetahui konsep dasar dari MQTT maka dapat dilihat pada Gambar 2.7.



Gambar 2.7. Konsep Dasar MQTT [14]

Berdasarkan Gambar 2.7, 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* [15]. *Publish* adalah konsep yang menjadikan *client* akan mengirimkan informasi. Informasi tersebut akan dikirimkan dengan melabeli informasi tertentu dengan suatu *topic*. *Subscribe* adalah konsep yang menjadikan *client* akan menerima informasi sesuai dengan *topic* yang diminta oleh *client* tersebut. Dalam MQTT juga terdapat *Quality of Service (QoS)* yang digunakan untuk memastikan kualitas pelayanan pada pengiriman pesan antara broker dan klien. QoS sendiri dibagi menjadi tiga yaitu level 0 (*At most once delivery*) yang memiliki makna bahwa pesan akan dikirimkan sebanyak satu kali dan tidak memiliki jaminan bahwa pesan akan sampai pada penerima, level 1 (*At least once delivery*) berarti pesan akan dikirimkan minimal

satu kali (dapat terjadi duplikasi) dan memiliki jaminan bahwa pesan akan diterima oleh penerima serta level 2 (*Exactly once delivery*) yang berarti pesan akan dikirimkan hanya satu kali dan memiliki jaminan bahwa pesan akan sampai pada penerima.

2.6. Node.js

Node.js adalah suatu *run-time environment* untuk menjalankan aplikasi web berbasis bahasa pemrograman Javascript yang berjalan pada sisi server. Node.js juga bersifat *open source* dan *cross-platform* (Windows, Linux, Mac OS dan sebagainya). Dalam mendukung performanya, Node.js dibangun dengan engine Javascript V8 milik Google. Node.js berjalan pada proses tunggal, tanpa membuat *thread* baru untuk setiap request. Node.js menyediakan kumpulan I/O yang bersifat asinkron pada *library* yang mencegah kode Javascript dari *blocking* sehingga Node.js dibentuk dengan paradigma *non-blocking*. Ketika Node.js melakukan operasi I/O seperti mengakses sistem file atau database dan menangani *request* dari suatu file maka dapat melanjutkan operasi selanjutnya ketika respons didapatkan tanpa memblokir *thread* dan menunggu operasi yang sedang berjalan. Dengan model *event-driven* dan *non-blocking I/O*, Node.js lebih mampu menangani banyak proses secara bersamaan daripada platform bersifat *thread-based networking* [16].

Pada Node.js sendiri terdapat beberapa framework yang biasanya digunakan seperti Express, AdonisJS, Fastify, Loopback.io, Socket.io, NextJS dan sebagainya. Pada pembuatan skripsi ini, framework yang digunakan adalah Express. Express sendiri merupakan framework aplikasi web Node.js yang bersifat fleksibel [17]. Selain itu, Express juga menyediakan serangkaian fitur yang dapat mempermudah pembuatan aplikasi web daripada menggunakan modul http bawaan Node.js. Framework ini menawarkan beberapa fitur seperti *routing*, *rendering view* dan mendukung *middleware* yang dapat menghemat waktu dalam pengembangan aplikasi Node.js. Framework ini juga memungkinkan untuk membuat web server HTML, server file statik, aplikasi chat, *search engine*, sosial media, layanan web dengan akses melalui REST API.

2.7. AngularJS

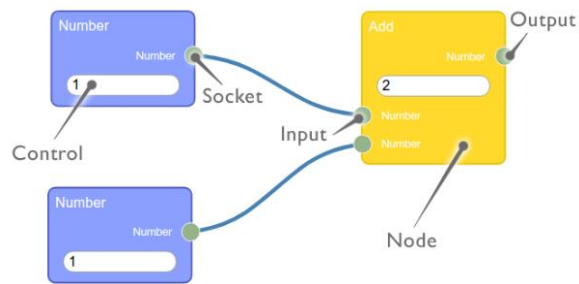
AngularJS adalah *frontend framework* Javascript untuk aplikasi web dinamis yang dikembangkan oleh Google dan bersifat *open source*. AngularJS sendiri berjalan pada sisi klien yang dapat diakses melalui browser [18]. Pada AngularJS menggunakan arsitektur *Model View Controller* (MVC). *Model* berperan untuk mengatur, menyiapkan, memanipulasi dan mengorganisasikan data dari database sesuai dengan instruksi dari *controller*. *View* berperan untuk menyajikan informasi atau data kepada pengguna sesuai dengan instruksi dari *controller*. *Controller* berperan sebagai jembatan antara *view* dan *model* dan *controller* juga akan mengatur apa yang harus dilakukan *model* serta mengatur apa yang harus ditampilkan oleh *view* berdasarkan permintaan dari pengguna. AngularJS juga dapat dibagi menjadi tiga bagian utama yaitu ng-app sebagai *directive* yang digunakan untuk mendefinisikan aplikasi AngularJS ke HTML, ng-model sebagai *directive* yang digunakan untuk memberikan nilai dari data aplikasi AngularJS ke input HTML dan ng-bind sebagai *directive* yang digunakan untuk memberikan data aplikasi AngularJS ke tag HTML. AngularJS juga memiliki beberapa fitur utama seperti sebagai berikut [19].

1. *Data-binding* yang digunakan untuk melakukan sinkronisasi data secara otomatis antara komponen *model* dan *view*.
2. *Scope* merupakan sebuah objek yang mengacu pada *model* dan bertugas sebagai jembatan antara *controller* dan *view*.
3. *Controller* merupakan fungsi javascript yang terikat pada lingkup tertentu.
4. *Directives* sebagai penanda pada elemen DOM seperti elemen, atribut, css dan lain-lain. Ini dapat digunakan untuk membuat kustom tag HTML yang berfungsi sebagai widget baru. AngularJS memiliki *built-in directives* seperti ngBind, ngModel dan sebagainya.
5. *Routing* merupakan konsep *switching* atau *routing* dari suatu halaman (*view*).
6. *Dependency Injection* yang berarti AngularJS memiliki *built-in dependency* yang dapat membantu developer dalam memahami, mengembangkan dan menguji aplikasi dengan mudah.

7. *Template* sebagai tampilan yang diberikan dengan informasi dari *controller* dan *model*. *Template* ini dapat membuat beberapa *view* dalam satu halaman menggunakan file tunggal seperti *index.html*.
8. *Services* yaitu AngularJS dapat membuat layanan seperti *\$http* untuk membuat *XMLHttpRequests*.
9. *Deep linking* memungkinkan untuk aplikasi dengan URL yang dapat di-*bookmark*.

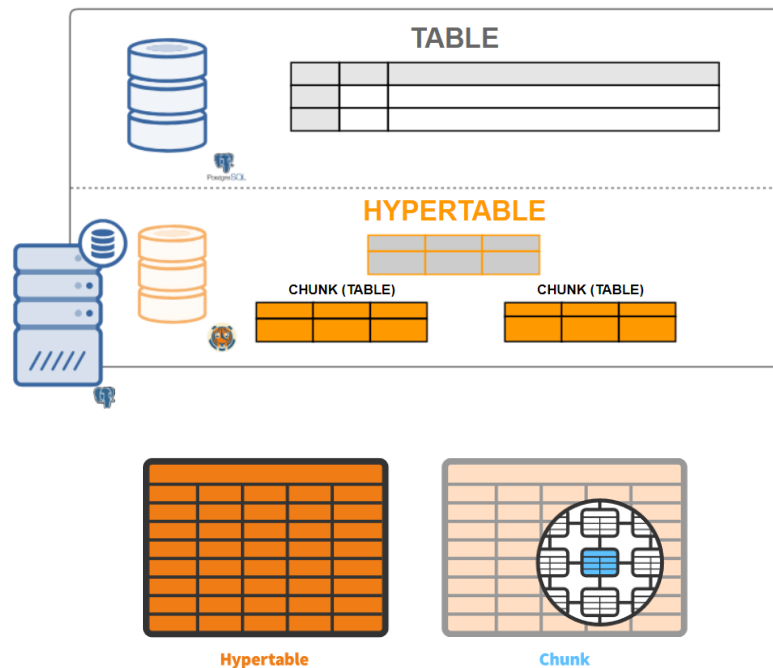
2.8. Rete.js

Rete.js merupakan *framework* Javascript yang digunakan untuk *visual programming* dan bersifat modular. Rete.js memungkinkan untuk membuat atau mengembangkan *node-based editor* secara langsung pada browser. Dengan adanya Rete.js juga dapat mendefinisikan *node* dan *worker* yang mengizinkan pengguna dapat memberikan instruksi untuk memproses data pada *node editor* yang dibuat. Dalam membuat *node editor* terdapat beberapa *built-in components* yang dibutuhkan seperti *socket* digunakan untuk mendefinisikan atau menentukan koneksi antara *input* dan *output*, *input* untuk merepresentasikan sebuah masukan dari koneksi yang dibuat, *output* untuk merepresentasikan sebuah keluaran dari koneksi yang dibuat dan *control* sebagai kolom editor dengan tipe data yang telah didefinisikan dan digunakan untuk menerima input sesuai yang diberikan oleh pengguna [20]. Inisialisasi editor dengan menghubungkan *plugin* yang penting untuk menampilkan *nodes* dan *connections* kemudian mendaftarkan *node* yang dibuat ke fungsi *register* dan memasukan ke editor event pada Rete.js. Untuk memproses skema yang dibuat, dibutuhkan inisialisasi *engine*. Komponen pada Rete.js memungkinkan untuk memproses data secara langsung pada *node* dan data tersebut dapat dikirimkan ke *node* lain melalui koneksi *input-output*. Hasil dari *node editor* dapat disimpan dalam bentuk JSON agar dapat digunakan pada sistem lain. Untuk lebih memperjelas mengenai bentuk *node editor* pada Rete.js dapat dilihat pada Gambar 2.8.

Gambar 2.8. Tampilan *Node Editor* pada Rete.js [20]

2.9. TimescaleDB

TimescaleDB yang merupakan database mendukung SQL yang didesain untuk menerima rekaman data yang dikumpulkan berdasarkan *timeseries* dan juga bersifat *scalable* dan analitik. TimescaleDB sebagai ekstensi dari PostgreSQL yang juga berjalan pada layanan PostgreSQL. Sehingga untuk melakukan query tidak jauh berbeda. Pada timescaleDB sendiri menggunakan *hypertable* yang merupakan abstraksi dari banyak potongan tabel tunggal yang menyimpan rekaman data. Dalam hal ini, potongan tabel tunggal tersebut disebut *chunks* [21]. Untuk memperjelas mengenai *hypertable* dan tabel *chunk* maka dapat dilihat pada arsitektur TimescaleDB pada Gambar 2.9.



Gambar 2.9. Arsitektur TimescaleDB [21]

Dalam praktiknya, semua instruksi (membuat tabel, memasukan data, menampilkan data, mengubah data, menghapus data, mengganti atribut tabel dan melakukan pengindeksan (*indexing*) yang berinteraksi dengan timescaleDB terdapat pada *hypertable*. *Indexing* yang dibentuk bertujuan untuk mempercepat proses pencarian data pada tabel berdasarkan *timeseries*. Ketika membuat *hypertable* maka secara otomatis tabel *chunk* juga terbentuk (*automatic chunking*) dengan default interval tabel *chunk* yaitu tujuh hari. Interval pada tabel *chunk* juga dapat diatur sesuai keinginan pengguna. Dalam menghapus data lama, timescaleDB menyediakan fungsi untuk menghapus tabel *chunk* dari pada hanya menghapus baris pada tabel. TimescaleDB ini dapat mengefisiensikan kemampuan *data retention* dan agregasi data secara *realtime*.

2.10. GraphQL

GraphQL merupakan konsep baru dalam membangun API dengan menggunakan bahasa query. GraphQL ini dikembangkan oleh Facebook dan diimplementasikan pada sisi server atau klien yang berhubungan dalam mengakses suatu API. Dalam praktiknya graphql akan mengeksekusi query sesuai dengan yang diminta oleh pengguna dan akan mengembalikan nilai tersebut dalam bentuk JSON atau dapat dikatakan GraphQL sebagai penerjemah bahasa query. GraphQL tidak terbatas untuk *database* tertentu, sehingga pengguna dapat menggunakan *database* yang sudah ada sebelumnya. Salah satu tujuan pengembangan bahasa query ini adalah untuk mempermudah komunikasi data antara *backend* dan *frontend* [22]. GraphQL dapat diimplementasikan di berbagai bahasa pada sisi klien seperti react, vue, meteor, angular, dan apapun jenis *framework*-nya selama dapat mengakses data dengan API.

2.11. Hasura

Hasura merupakan server GraphQL yang dapat menyediakan GraphQL API secara instan dan *realtime*. Hasura dikembangkan untuk dapat berjalan pada *database* PostgreSQL. Dengan memanfaatkan Hasura, pengguna sudah dapat membentuk API menggunakan GraphQL dengan pengguna tidak perlu membuat GraphQL server secara mandiri. Sehingga untuk membuat API tidak perlu

membutuhkan waktu yang lama dalam melakukan *setup* dengan menggunakan banyak baris kode. Hasura memiliki beberapa fitur antara lain [23]:

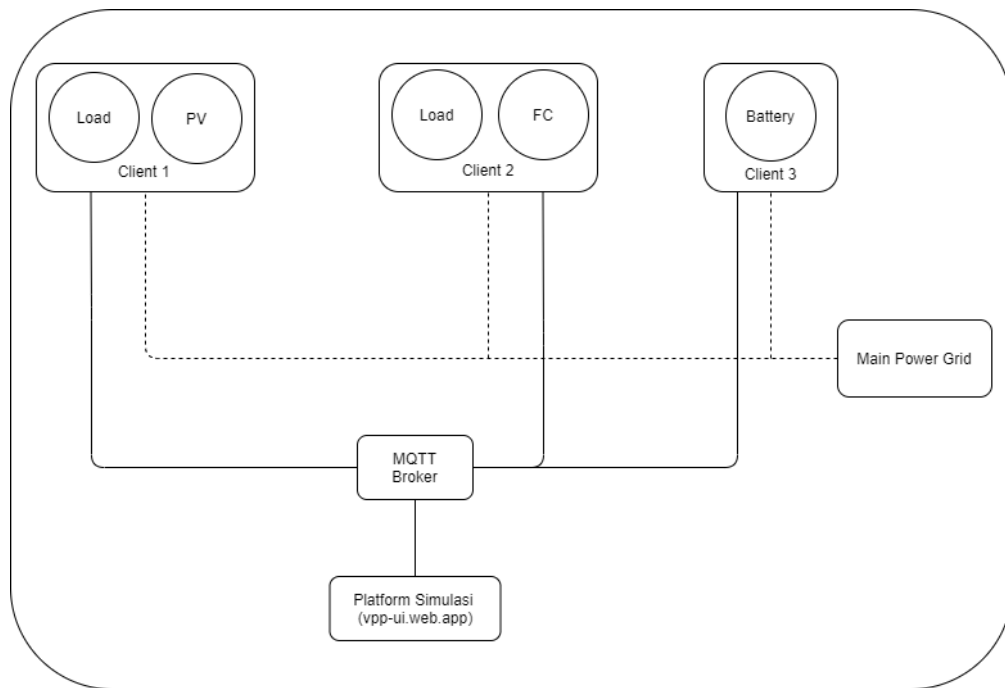
1. Memiliki query bawaan seperti *built-in filtering, pagination, pattern search, bulk insert, update, delete mutations*.
2. Hasura dapat mengubah query GraphQL menjadi *live query* dengan menggunakan fungsi *subscription*.
3. Dapat menggunakan *database* yang telah ada sebelumnya untuk mendapatkan data dengan GraphQL API yang dapat langsung digunakan.
4. Memiliki kemampuan untuk menghubungkan *database* yang sudah ada sebelumnya sehingga dapat diakses hanya dengan menggunakan satu alamat melalui fitur *remote schema*.
5. Memiliki *dashboard* yang dapat mempermudah konfigurasi.

BAB 3

PERANCANGAN SISTEM

Bab ini akan menjelaskan tentang pemodelan konsep pembangkit listrik virtual, pengembangan platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual, perancangan *node editor* untuk mendukung sistem komunikasi pada pembangkit listrik virtual, *use case diagram* dan diagram alir pada platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual.

3.1. Pemodelan Konsep Pembangkit Listrik Virtual



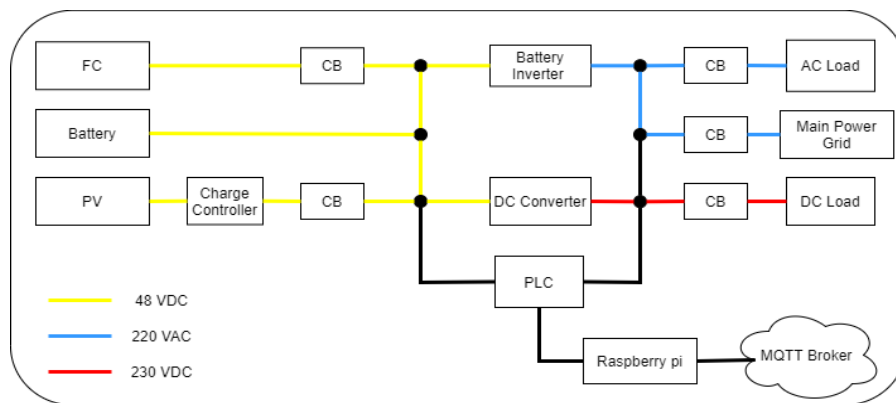
Gambar 3.1. Pemodelan Konsep Pembangkit Listrik Virtual

Pada Gambar 3.1, 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 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* tersebut merupakan sistem yang sudah berjalan dengan data-data yang sudah ditentukan. Untuk melihat perancangan sistem tersebut dapat dilihat pada Gambar 3.2



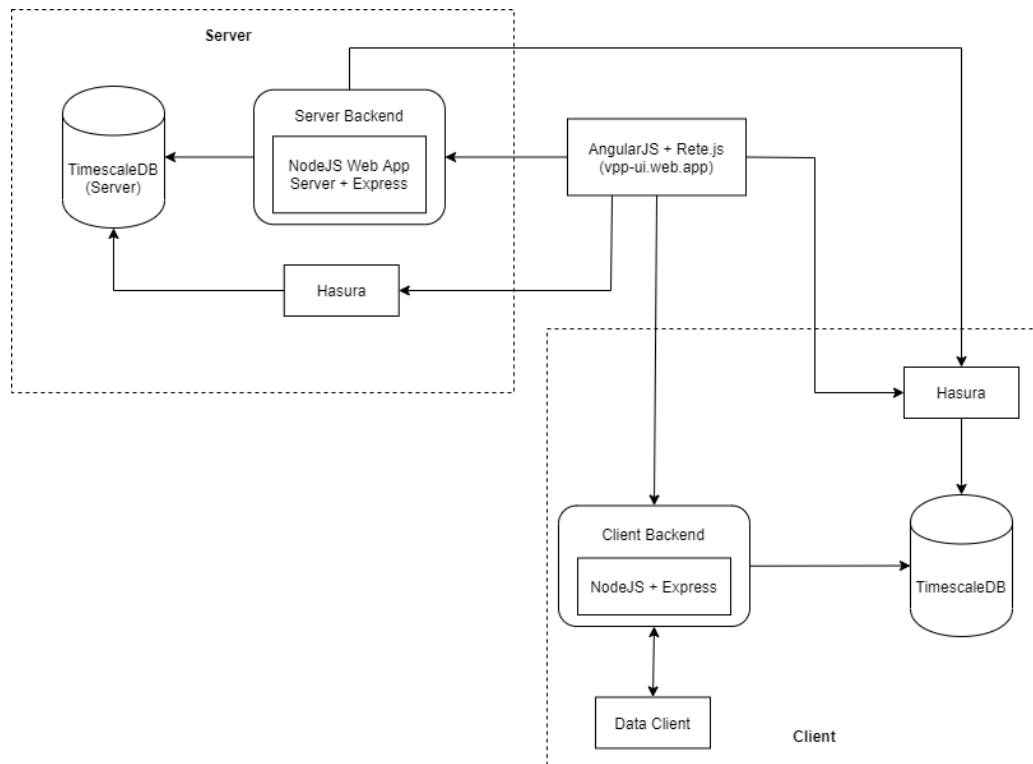
Gambar 3.2. Perancangan Sistem pada *Sofwan House TREC FTUI*

Berdasarkan Gambar 3.2, 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. Pada sistem di *container* juga terdapat *battery inverter* yang digunakan untuk mengkonversi arus DC ke AC (48 VDC menjadi 220 VAC) serta *DC converter* yang digunakan untuk meningkatkan tegangan pada arus DC (48 VDC menjadi 230 VDC). Selain itu juga, terdapat *Circuit Breaker* (CB) sebagai

pemutus aliran yang digunakan untuk mengganti sumber energi apakah dari pembangkit atau dari *main power grid* serta pemutus aliran pada beban baik AC (mesin cuci, pompa air, *air conditioner* dan lain-lain) maupun DC (lampu, kompor listrik, blender, printer, dan sebagainya). 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 diambil atau 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.

3.2. Pengembangan Platform untuk Pemodelan Pembangkit Listrik Virtual

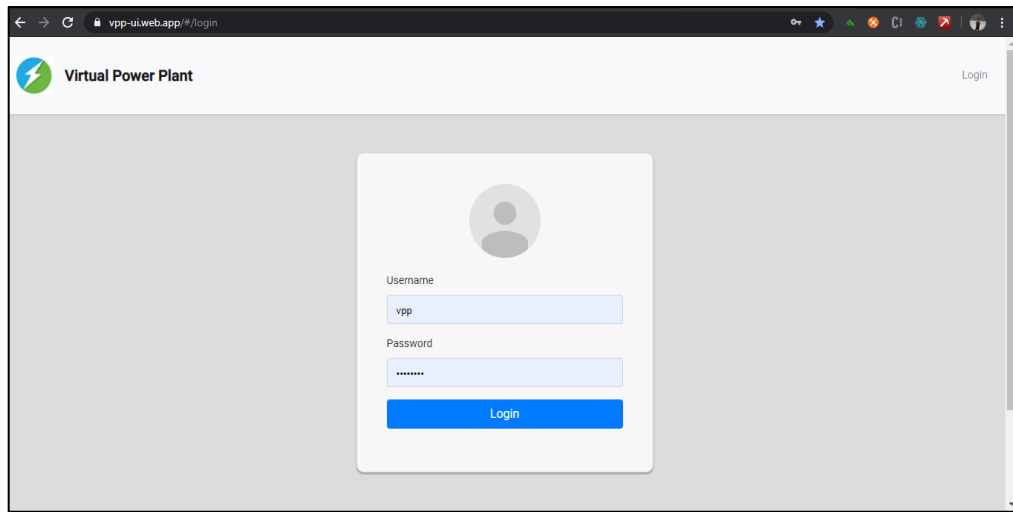
Platform ini digunakan untuk memodelkan konsep pembangkit listrik virtual seperti melakukan pemantauan, kendali dan juga manajemen energi berdasarkan logika yang sudah dibentuk yaitu mengagregasi daya yang dihasilkan oleh seluruh jenis pembangkit serta daya yang digunakan oleh beban sehingga daya dari pembangkit dan beban yang telah diagregasi akan dibandingkan untuk mengetahui keputusan yang dibuat dalam menentukan pembangkit mana yang akan aktif untuk mengatasi beban pada *grid*. Klien pada pemodelan ini memiliki kemampuan untuk menghasilkan energi dari pembangkit yang dimiliki, mengonsumsi energi listrik, menyimpan energi menggunakan baterai. Platform ini memungkinkan klien dapat memantau aktivitas pembangkit dan beban. Untuk memperjelas mengenai arsitektur platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual dapat dilihat pada Gambar 3.3.



Gambar 3.3. Arsitektur Platform untuk Pemodelan Pembangkit Listrik Virtual

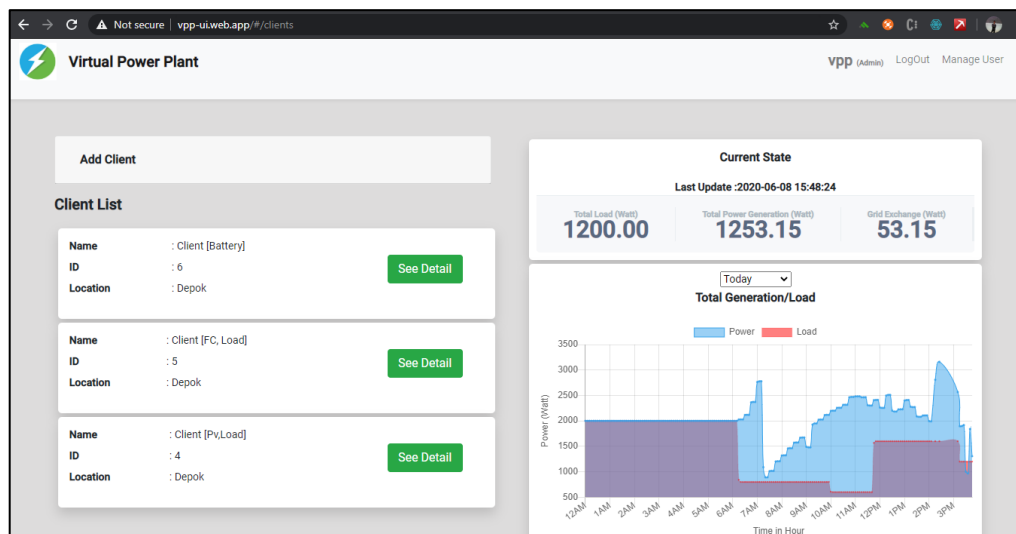
Berdasarkan Gambar 3.3, terdapat tiga bagian utama dalam platform ini yaitu server, website dan klien. Pada server memiliki *backend* yang menyediakan layanan *web service* menggunakan Node.js dan Express sehingga dapat berkomunikasi atau terhubung menggunakan RESTFUL API. Server dalam platform ini bertugas 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 mana 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. Website dibentuk dengan menggunakan AngularJS yang berjalan pada sisi klien sebagai

frontend yang di-hosting menggunakan layanan pada Firebase. Pada website, memiliki halaman login pada platform yang dapat dilihat pada Gambar 3.4.



Gambar 3.4. Halaman Login pada Platform

Pada platform ini, website bertindak sebagai 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. Berikut ini merupakan halaman utama pada platform yang dapat dilihat pada Gambar 3.5.



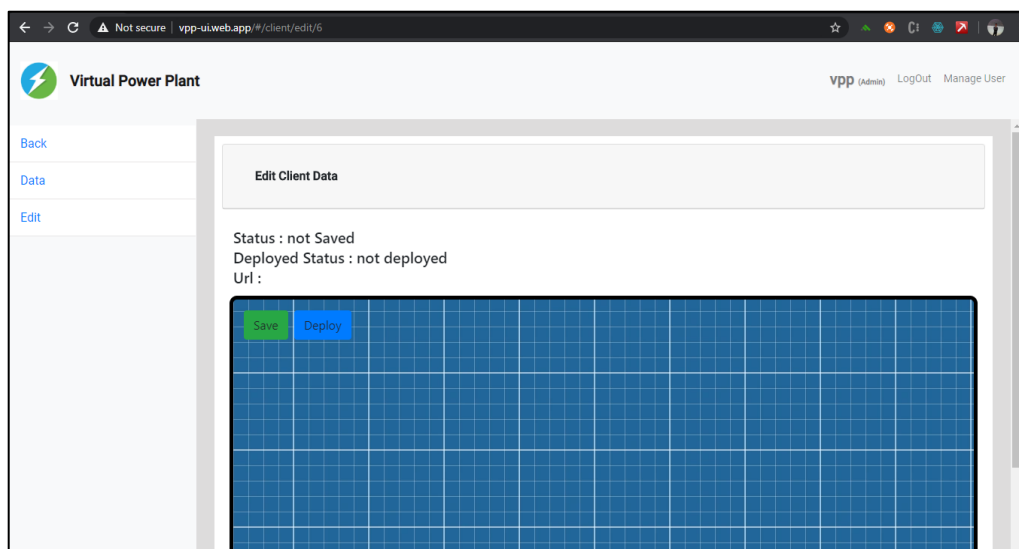
Gambar 3.5. Halaman Utama pada Platform

Klien dalam platform 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 konverter. 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.

3.3. Perancangan *Node Editor* pada Platform untuk 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. Pembuatan *node editor* pada Rete.js perlu dilakukan pendefinisian *node* dan *worker* yang dapat memberikan instruksi untuk memproses data pada *node editor* yang dibuat. Dalam membuat *node editor*, diperlukan penentuan *built-in components* yang akan digunakan seperti *socket*, *input*, *output* dan *control*. Inisialisasi *node editor* dengan menghubungkan *plugin* yang penting untuk menampilkan *nodes* dan *connections* kemudian mendaftarkan *node* yang dibuat ke fungsi register dan memasukan ke editor event pada Rete.js. Untuk memproses skema yang dibuat, dibutuhkan inisialisasi *engine* terlebih dahulu. Untuk mengetahui halaman *node editor* pada platform dapat dilihat pada Gambar 3.6.



Gambar 3.6. Halaman *Node Editor* pada Platform

Node yang dibuat untuk mendukung platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual antara lain *node* MQTT (*Subscriber* dan *Publisher*), kalkulasi (*Addition* dan *Multiplication*), *database*, akumulator (*Load* dan *Generator*), Spesifikasi (*Battery* dan *Fuel Cell*) dan kontrol.

3.3.1. Node MQTT (*Subscriber* dan *Publisher*)

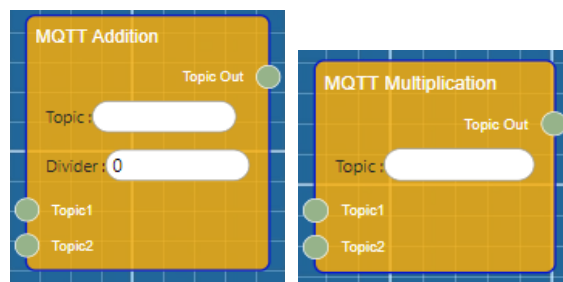
The image shows two side-by-side screenshots of MQTT client interfaces. The left interface is titled 'MQTT Client Subscriber' and features a 'Topic' label with a green circular indicator. Below it are input fields for 'Host', 'Port' (pre-filled with '0'), 'Username', 'Password', and 'Topic'. The right interface is titled 'MQTT Client Publisher' and features input fields for 'Host', 'Port' (pre-filled with '0'), 'Username', 'Password', 'Topic', and 'Message'.

Gambar 3.7. *Node* MQTT Client Subscriber dan Publisher

Berdasarkan Gambar 3.7, 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 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 dengan tipe data string, *port* dengan tipe data *number*, *username* dengan tipe data string, *password* dengan tipe data string, dan *topic* dengan tipe data string untuk menerima input sesuai yang diberikan oleh pengguna. Selain itu, terdapat juga komponen *output* yang akan mengirimkan nilai yang telah didapatkan berdasarkan topik kepada *node* lain sesuai dengan koneksi yang dibuat (*Node* Kalkulasi atau *Node* MQTT Database). *Node* MQTT Client Publisher digunakan untuk mengirimkan nilai 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*.

3.3.2. Node Kalkulasi (*Addition* dan *Multiplication*)



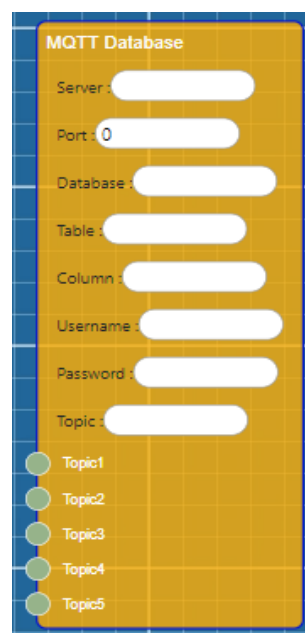
Gambar 3.8. *Node MQTT Addition* dan *Multiplication*

Pada Gambar 3.8, menunjukan bahwa kedua *node* tersebut digunakan untuk kalkulasi. *Node MQTT Addition* dibentuk untuk melakukan kalkulasi berupa penambahan dari nilai yang didapatkan dari *node MQTT Client Subscriber* serta pembagian dari nilai 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 melalui komponen *input*. Terdapat juga komponen *input* yang akan menerima nilai dari *node MQTT Client Subscriber* serta komponen *output* yang akan mengirimkan hasil kalkulasi dari nilai yang telah didapatkan kepada *node* lain sesuai dengan koneksi yang dibuat

(*Node Database*). *Node MQTT Multiplication* dibentuk untuk untuk melakukan kalkulasi berupa perkalian dari nilai 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*. Pembuatan *node* kalkulasi ini berdasarkan dengan dokumentasi yang terdapat pada sistem di *container* yang sudah berjalan.

Dalam pembuatan *backend* dari *Node MQTT Addition* dan *Node MQTT Multiplication* 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.

3.3.3. *Node Database*



Gambar 3.9. *Node MQTT Database*

Berdasarkan Gambar 3.9, terlihat bahwa *node MQTT Database* dibentuk untuk menyimpan 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 *database* dengan tipe data string, *port* dengan tipe data *number*, *database* dengan tipe data string, *table* dengan tipe data string, *column* dengan tipe data string, *username* dengan tipe data string, *password* dengan tipe data string untuk menerima input sesuai yang diberikan oleh pengguna serta *topic* dengan tipe data string untuk menerima pesan yang diberikan oleh *node* lain melalui komponen *input*. Terdapat juga komponen *input* yang akan menerima nilai dari *Node MQTT Client Subscriber* atau *Node Kalkulasi*.

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.

3.3.4. Node Akumulator (*Load* dan *Generator*)

Gambar 3.10. *Node Generator* dan *Load Accumulator*

Pada Gambar 3.10, menunjukan bahwa *node Generator* dan *Load Accumulator* dibentuk untuk menghitung total energi yang digunakan selama sehari pada pembangkit dan beban. Total energi yang dihitung berdasarkan akumulasi daya per hari yang didapatkan dari tabel pada *database* kemudian diakumulasikan lagi dengan total energi pada hari sebelumnya sehingga menyebabkan total energi 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* dengan tipe data string, *port* dengan tipe data *number*, *database* dengan tipe data string, *table* dengan tipe data string, *column* dengan tipe data string, *username* dengan tipe data string dan *password* dengan tipe data string 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 energi 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 mengelompokkan 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 energi dalam kWh akan bertambah terus menerus setiap harinya. Proses perhitungan total energi dilakukan dengan penjadwalan menggunakan fungsi *Cron Job* setiap pukul 23:59.

3.3.5. Node Spesifikasi (*Battery* dan *Fuel cell*)

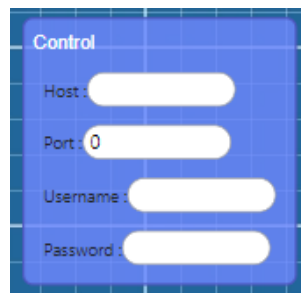


Gambar 3.11. *Node Battery* and *FuelCell Specification*

Pada Gambar 3.11, 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 dengan baik sesuai fungsinya. Pada kedua *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.

3.3.6. Node Kontrol


 The image shows a web-based control interface titled "Control". It contains four input fields: "Host:" with a white text input box, "Port:" with a white text input box containing the number "0", "Username:" with a white text input box, and "Password:" with a white text input box. The entire interface is set against a blue background with a subtle grid pattern.

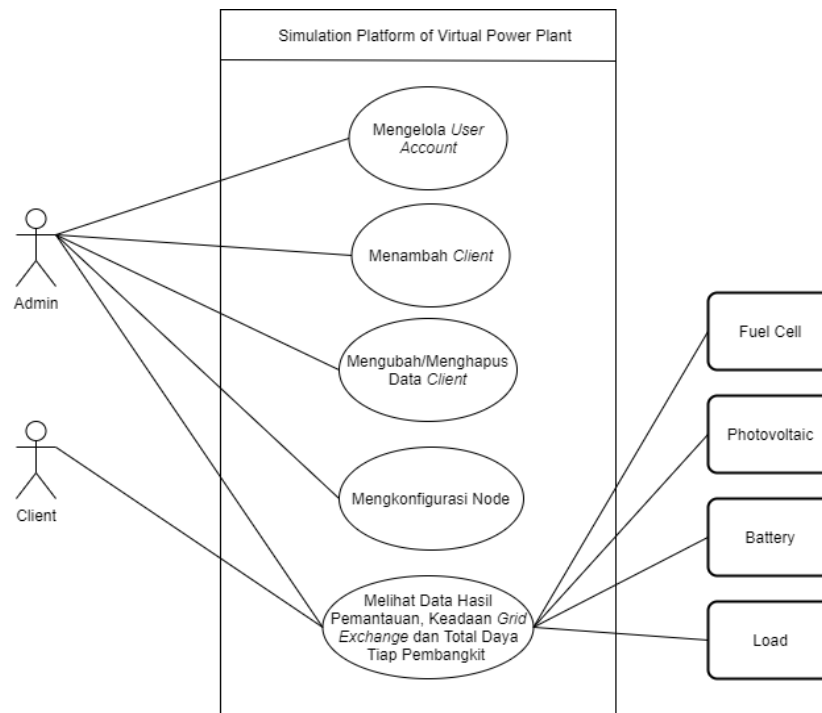
Gambar 3.12. *Node Control*

Berdasarkan Gambar 3.12, menunjukkan bahwa *node control* digunakan untuk mengontrol pembangkit melalui pendefinisian *credentials* broker yang diberikan oleh pengguna. *Node* ini akan mengirimkan nilai berupa status atau keadaan dari pembangkit (aktif atau tidak aktif) ke broker dengan topik dan nilai atau pesan yang sudah didefinisikan pada algoritma yang telah dibuat pada server. Pada *node* ini terdapat komponen *control* dengan tipe data yang telah didefinisikan yaitu alamat broker dengan tipe data string, *port* dengan tipe data *number*, *username* dengan tipe data string, *password* dengan tipe data string 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*).

3.4. 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 serta dapat dilihat pada Gambar 3.13.

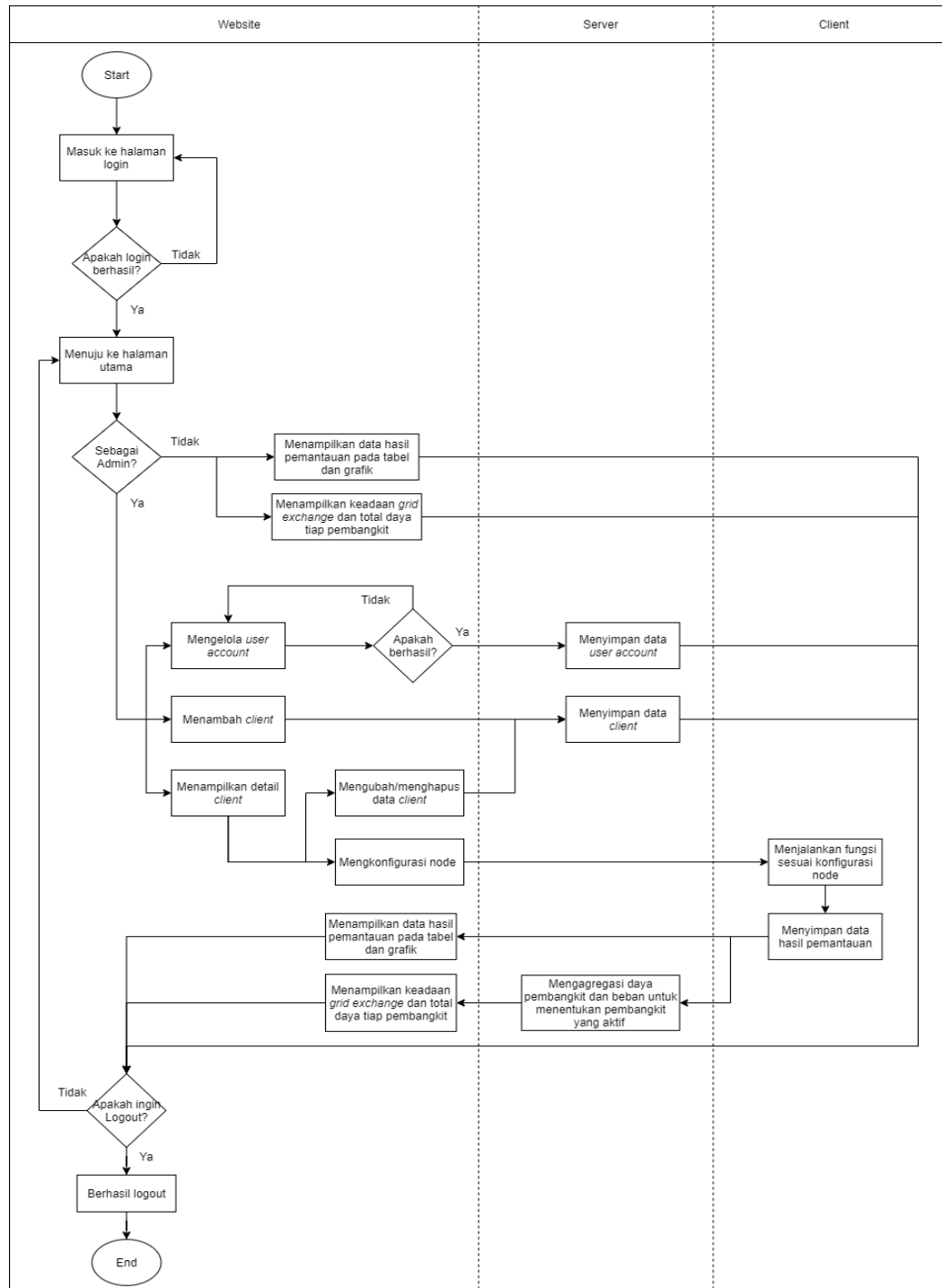


Gambar 3.13. Use Case Diagram pada Platform

Pada Gambar 3.13, terlihat bahwa platform sebagai sistem pemantauan 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 dilakukan oleh admin meliputi mengelola (menambah atau menghapus) 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*. Data klien yang dapat dilihat oleh admin dan klien meliputi data pemantauan terhadap *photovoltaic*, *fuel cell*, baterai dan beban.

3.5. Diagram Alir pada Platform untuk Pemodelan Pembangkit Listrik Virtual

Dalam memberikan gambaran terhadap keseluruhan kerja sistem pada platform ini, maka disertakan diagram alir yang dapat dilihat pada Gambar 3.14.



Gambar 3.14. Diagram alir pada Platform

Berdasarkan diagram alir pada Gambar 3.14, proses kerja sistem dibagi menjadi dua bagian berdasarkan *role* yaitu admin dan klien. Ketika berhasil *login* sebagai admin, maka akan diarahkan ke halaman utama oleh sistem pada platform. Di halaman utama, admin dapat melakukan aktivitas seperti mengelola (menambah atau menghapus) akun pengguna, menambah klien, melihat detail klien. Ketika admin sudah melakukan pengelolaan akun pengguna dan penambahan klien maka data tersebut akan disimpan pada *database* di server. Pada aktivitas menampilkan detail klien di platform ini, admin dapat mengubah atau menghapus data klien dan mengkonfigurasi *node* pada klien. Kemudian konfigurasi *node* tersebut dapat di-*deploy* untuk dikirimkan dalam bentuk JSON ke klien untuk dijalankan fungsi sesuai konfigurasi *node* dan ketika sudah didapatkan hasilnya berupa data pemantauan maka akan disimpan pada *database* TimescaleDB di klien. Ketika data pada setiap klien sudah didapatkan kemudian mengagregasi daya pada pembangkit dan beban untuk menentukan pembangkit mana yang akan aktif. Data-data pada setiap klien akan ditampilkan pada website menggunakan tabel dan grafik. Selain itu, total daya tiap pembangkit serta keadaan dari *grid exchange* juga akan ditampilkan di halaman utama. Nilai *grid exchange* didapatkan dari selisih dari agregasi daya pada seluruh jenis pembangkit dan beban. Ketika admin sudah selesai dalam melakukan aktivitas yang berkaitan dengan platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual maka admin dapat *logout* untuk mengakhirinya.

Sedangkan ketika berhasil *login* sebagai klien maka akan diarahkan ke halaman utama. Pada halaman ini, klien hanya dapat menampilkan detail klien. Ketika diarahkan ke halaman untuk melihat detail klien, klien hanya dapat melihat data hasil pemantauan pada tabel dan grafik sesuai dengan data klien yang bersangkutan dan tidak bisa mengubah atau menghapus data klien yang berkaitan dengan klien tersebut. Selain data hasil pemantauan, klien juga dapat melihat total daya tiap pembangkit dan keadaan dari *grid exchange* di halaman utama dengan nilai dari *grid exchange* didapatkan dari selisih dari agregasi daya pada seluruh jenis pembangkit dan beban. Untuk mengakhiri aktivitas yang berkaitan dengan platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual maka klien dapat *logout* dari website.

BAB 4

PEMODELAN, PENGUJIAN DAN ANALISIS SISTEM

Bab ini akan menjelaskan mengenai pemodelan dari perancangan sistem yang telah dilakukan. Selain itu, juga akan menjelaskan mengenai pengujian yang dilakukan pada platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual serta analisis terhadap hasil pengujian yang telah dilakukan.

4.1. Pemodelan Sistem

Dalam pemodelan yang dilakukan pada platform sebagai sistem pemantauan untuk pemodelan 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 sebagai sistem pemantauan 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 sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual. Spesifikasi *instance* yang digunakan pada server dari platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual 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 sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual. Selain itu, pada setiap klien juga memiliki *database* yang akan

menyimpan data-data sesuai dengan klien tersebut. Spesifikasi *instance* pada setiap klien sebagai berikut.

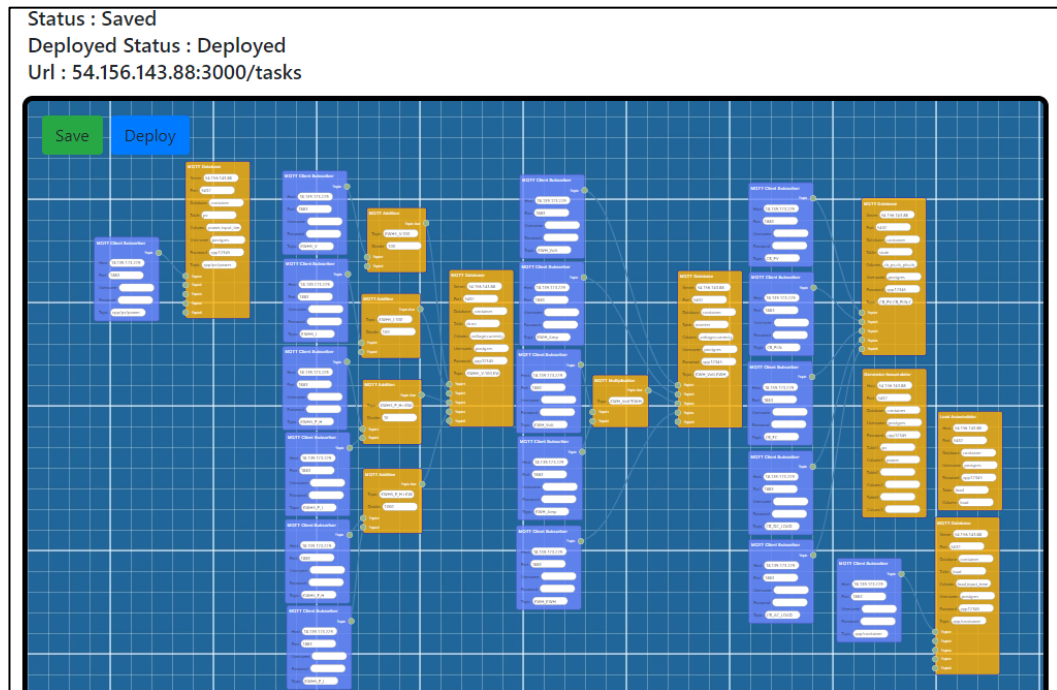
- Jumlah virtual CPU: 1
- RAM: 2 GB
- *Network Bandwidth*: Up to 10 Gbps
- Kapasitas Penyimpanan: 8 Gb

Pada setiap klien perlu dilakukan instalasi *backend*, pengaturan database dan layanan Hasura. Kemudian alamat klien dijadikan sebagai URL untuk akses API klien yang didefinisikan pada konfigurasi data klien di website dan juga URL layanan Hasura sebagai layanan *stream* data yang akan mengembalikan nilai dari *database* dalam bentuk JSON untuk ditampilkan di website.

4.1.1. 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. Dalam memantau data-data tersebut, dilakukan konfigurasi *node editor* untuk mengatur *interfacing* antara platform dengan perangkat seperti pembangkit (*photovoltaic*), *battery inverter* dan *DC converter*. 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 4.1. Tampilan Konfigurasi *Node* pada Klien Pertama

Pada Gambar 4.1, menunjukan bahwa *node* yang digunakan antara lain *node* MQTT *Client Subscriber* untuk mendapatkan data dari broker dengan *credential* yang sudah didefinisikan pada *node* meliputi alamat broker, port dan topik yang disesuaikan, *node* kalkulasi baik *node* *Addition* maupun *Multiplication* untuk melakukan kalkulasi terhadap nilai yang didapatkan dari *node* MQTT *Client Subscriber*, *node* MQTT *database* untuk menyimpan nilai ke dalam *database* dengan nilai tersebut didapatkan dari *node* kalkulasi atau *node* MQTT *Client Subscriber* 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.

```
{
  "id": "demo@0.2.0",
  "nodes": {
    "84": {
      "id": 84,
      "data": {
        "host": "18.139.173.229",
        "port": 1883,
        "username": "",
        "password": "",
        "topic": "KWH8_V"
      },
      "inputs": {},
      "outputs": {
        "topic": {
          "connections": [
            {
              "node": 189,
              "input": "topic1",
              "data": {}
            }
          ]
        }
      },
      "position": [
        -5.086517543771045,
        73.21700667079558
      ],
      "name": "MQTT Client Subscriber"
    },
    "85": {
      "id": 85,
      "data": {
        "host": "18.139.173.229",
        "port": 1883,
        "username": "",
        "password": "",
        "topic": "KWH8_I"
      },
      "inputs": {},
      "outputs": {
        "topic": {
          "connections": [
            {

```

Gambar 4.2. Tampilan Potongan JSON pada Klien Pertama

Ketika JSON seperti yang dapat dilihat pada Gambar 4.2 sudah diterima oleh *client backend* maka klien akan menjalankan fungsi sesuai dengan konfigurasi *node* yang telah dibuat. Kemudian data hasil pemantauan 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 4.3.

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 4.3. Tampilan *Hypertable* pada Klien Pertama

Data *realtime* tersebut kemudian akan dipartisi sesuai interval yang sudah didefinisikan pada tabel *chunk* sehingga tabel *chunk* akan terus bertambah 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 4.4.

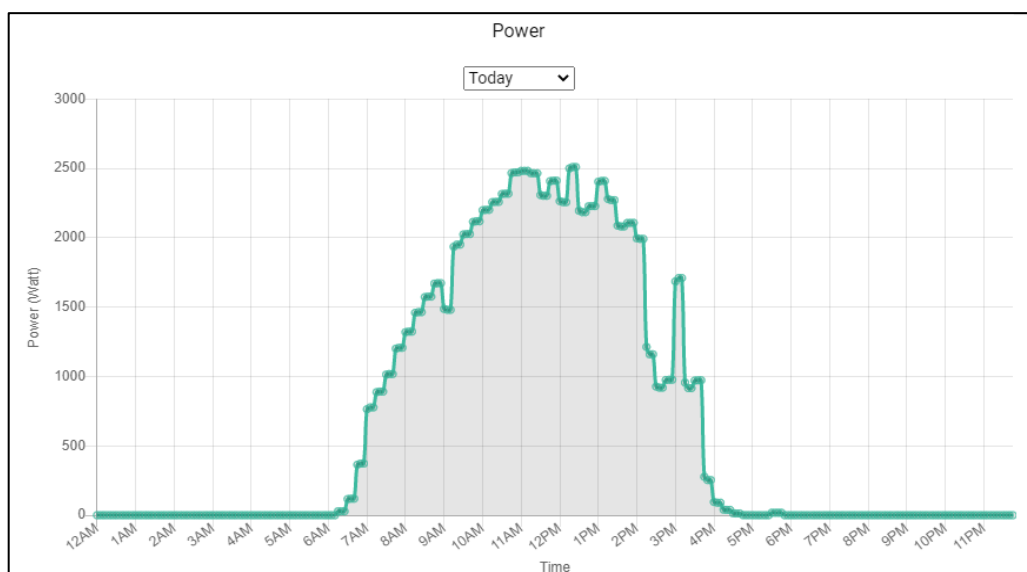
chunk_id integer	chunk_table text	ranges text[]	table_size text
41	_timescaledb_internal._hyper_12_41_chunk	{'["2020-06-04 07:00:00+07";2020-06-11 07:00:00+07]"}	4208 kB
28	_timescaledb_internal._hyper_2_28_chunk	{'["2020-05-28 07:00:00+07";2020-06-04 07:00:00+07]"}	4960 kB
37	_timescaledb_internal._hyper_2_37_chunk	{'["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 4.4. 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 serta 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. Dalam menampilkan data *realtime*, platform ini menggunakan layanan Hasura yang berfungsi untuk mengembalikan data dari *database* dalam bentuk JSON melalui fungsi *subscription*. Salah satu tabel dan grafik yang ditampilkan pada klien pertama yaitu data berupa daya yang dihasilkan oleh *photovoltaic* yang dapat dilihat pada Gambar 4.5 dan Gambar 4.6.

PV	DCON	Inverter	State	Load	Total Load	Total Power Generation
Power (W)					Input_time	
PV		Inverter	State	Load	Time	Total Power Generation
	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	

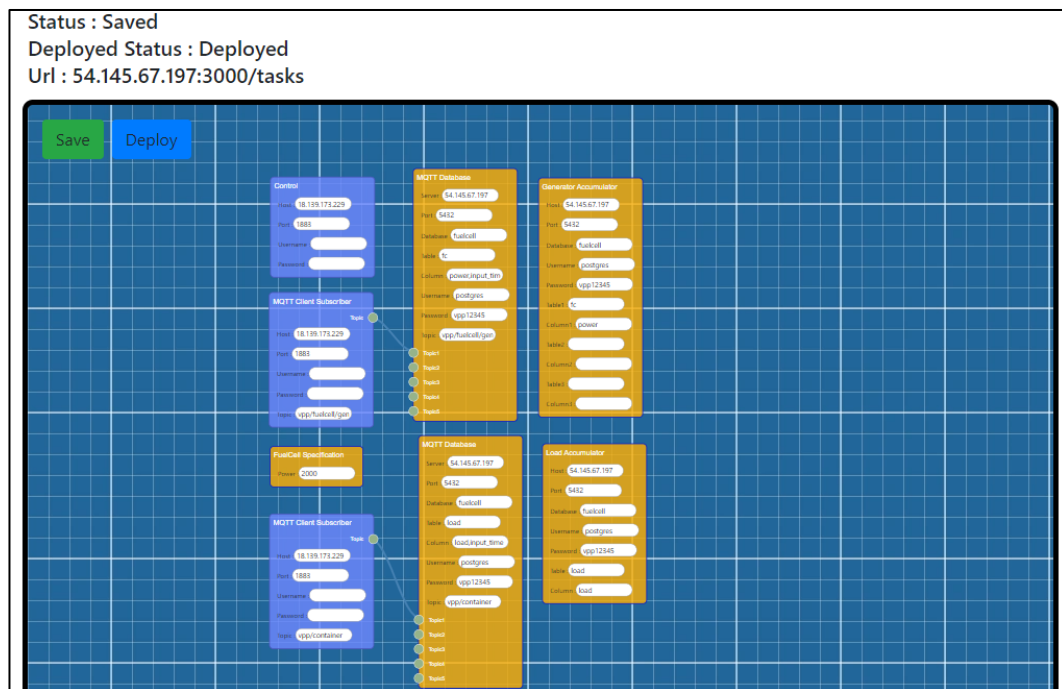
Gambar 4.5. Tampilan Tabel *Photovoltaic* pada Klien Pertama



Gambar 4.6. Tampilan Grafik *Power* pada *Photovoltaic* (Klien Pertama)

4.1.2. 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 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 4.7. Tampilan Konfigurasi *Node* pada Klien Kedua

Berdasarkan Gambar 4.7, terlihat bahwa *node* yang digunakan antara lain *node* MQTT *Client Subscriber* untuk mendapatkan data dari broker dengan credential yang sudah didefinisikan pada *node* meliputi alamat broker, port dan topik yang disesuaikan, *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 sehingga konfigurasi *node* tersebut tidak akan hilang ketika pengguna kembali lagi untuk mengubah konfigurasi *node* tersebut.

```
{
  "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": "topicMerge1",
              "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": {
        "topicMerge1": {
```

Gambar 4.8. Tampilan Potongan JSON pada Klien Kedua

Client backend akan menerima JSON seperti yang terlihat pada Gambar 4.8 dan menjalankan fungsi sesuai dengan konfigurasi *node* yang telah dibuat. Kemudian data hasil pemantauan 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 4.9.

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 4.9. Tampilan *Hypertable* pada Klien Kedua

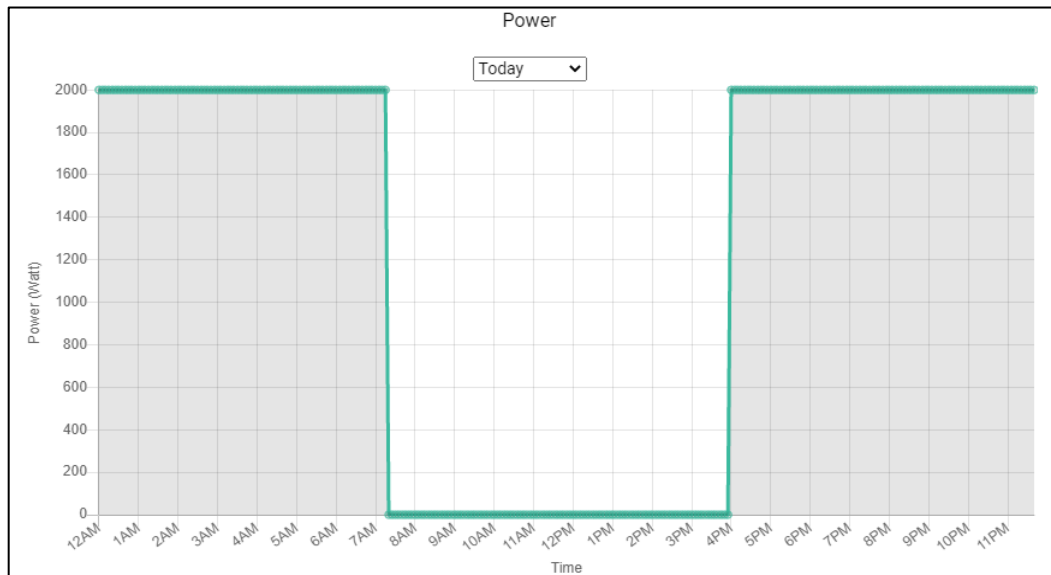
Data tersebut kemudian akan dipartisi sesuai interval yang sudah didefinisikan pada tabel *chunk* sehingga tabel *chunk* akan terus bertambah secara otomatis jika *timeseries* pada data sudah melewati *range* waktu yang sudah ditentukan untuk setiap tabel *chunk*. Daftar tabel *chunk* yang telah terbentuk dapat dilihat pada Gambar 4.10.

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 4.10. Daftar Tabel *Chunk* pada Klien Kedua

Data *realtime* yang sudah disimpan pada *database* kemudian data tersebut akan dikembalikan dalam bentuk JSON menggunakan fungsi *subscription* pada layanan Hasura. Data yang dikembalikan tersebut 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 4.11 dan Gambar 4.12.

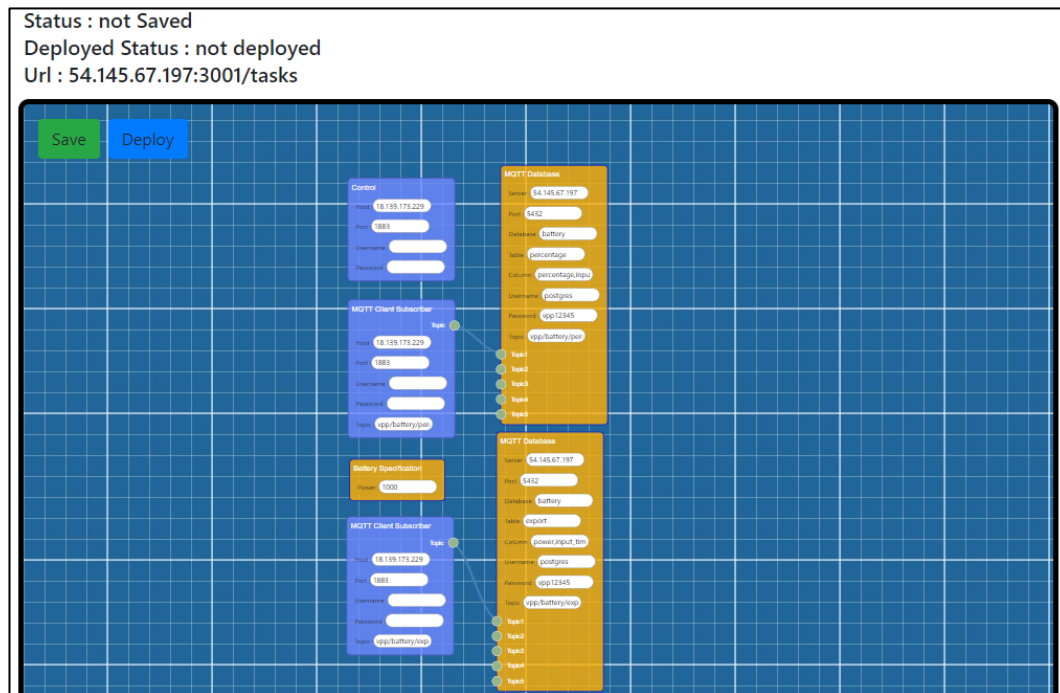
Fuelcell	Load	Total Load	Total Power Generation
Power (W)		Input_time	
2000		2020-6-18 14:36:47	
2000		2020-6-18 14:36:42	
2000		2020-6-18 14:36:37	
2000		2020-6-18 14:36:32	
2000		2020-6-18 14:36:27	
2000		2020-6-18 14:36:22	
2000		2020-6-18 14:36:17	
2000		2020-6-18 14:36:12	

Gambar 4.11. Tampilan Tabel *Fuel Cell* pada Klien KeduaGambar 4.12. Tampilan Grafik *Power* pada *Fuel Cell* (Klien Kedua)

4.1.3. 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 mencukupi 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 4.13. Tampilan Konfigurasi *Node* pada Klien Ketiga

Berdasarkan Gambar 4.13, menunjukkan bahwa *node* yang digunakan sama seperti pada klien kedua yaitu *node MQTT Client Subscriber* untuk mendapatkan data dari broker dengan credential yang sudah didefinisikan pada *node* meliputi alamat broker, port dan topik yang disesuaikan, *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.


```

{
  "id": "demo@0.2.0",
  "nodes": {
    "1": {
      "id": 1,
      "data": {
        "host": "18.139.173.229",
        "port": 1883,
        "username": "",
        "password": "",
        "topic": "vpp/battery/percentage"
      },
      "inputs": {},
      "outputs": {
        "topic": {
          "connections": [
            {
              "node": 2,
              "input": "topicMerge1",
              "data": {}
            }
          ]
        }
      },
      "position": [
        157.85341842048445,
        21.602119739824392
      ],
      "name": "MQTT Client Subscriber"
    },
    "2": {
      "id": 2,
      "data": {
        "server": "54.145.67.197",
        "portDB": 5432,
        "database": "battery",
        "table": "percentage",
        "column": "percentage",
        "usernameDB": "postgres",
        "passwordDB": "vpp12345",
        "topicStr": "vpp/battery/percentage"
      },
      "inputs": {
        "topicMerge1": {

```

Gambar 4.14. Tampilan Potongan JSON pada Klien Ketiga

Client backend akan menerima konfigurasi *node* berupa JSON seperti yang terlihat pada Gambar 4.14 kemudian menjalankan fungsi sesuai dengan konfigurasi *node* yang telah dibuat. Data hasil pemantauan 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 4.15.

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 4.15. Tampilan *Hypertable* pada Klien Ketiga

Data tersebut kemudian akan dipartisi sesuai interval yang sudah didefinisikan pada tabel *chunk* sehingga tabel *chunk* akan terus bertambah secara otomatis jika *timeseries* pada data sudah melewati *range* waktu yang sudah

ditentukan untuk setiap tabel *chunk*. Daftar tabel *chunk* dapat dilihat pada Gambar 4.16.

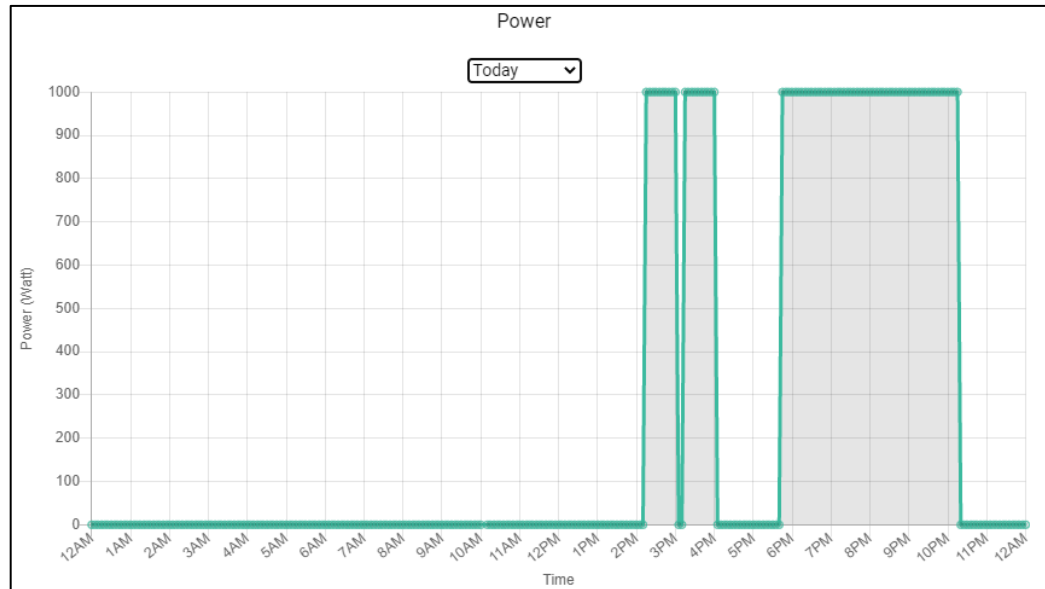
chunk_id integer	chunk_table text	ranges text[]	table_size text
3	_timescaledb_internal_hyper_1_3_chunk	{["2020-05-28 07:00:00+07";"2020-06-04 07:00:00+07"]}	4680 kB
5	_timescaledb_internal_hyper_1_5_chunk	{["2020-06-04 07:00:00+07";"2020-06-11 07:00:00+07"]}	4576 kB
chunk_id integer	chunk_table text	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 4.16. Daftar Tabel *Chunk* pada *Battery Generation*

Data *realtime* yang sudah disimpan pada *database* di klien akan ditampilkan pada website menggunakan layanan Hasura yang berfungsi untuk mengembalikan data dari *database* dalam bentuk JSON melalui fungsi *subscription*. Data ditampilkan 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 ketika melakukan *discharge* ke *grid* yang dapat dilihat pada Gambar 4.17 dan Gambar 4.18.

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 4.17. Tampilan Tabel *Battery Generation* pada Klien Ketiga



Gambar 4.18. Tampilan Grafik *Power* pada *Battery Generation* (Klien Ketiga)

4.2. Pengujian Sistem

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

4.2.1. Pengujian *Node Editor*

Pengujian yang dilakukan pada *node editor* bertujuan untuk mengetahui kemampuan adaptasi dan fleksibilitas pada *node editor* dalam mendukung platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual. 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

4.2.2. *Usability Testing*

Pada *Usability Testing*, pengujian dilakukan dengan memberikan kuesioner mengenai penilaian pengguna terhadap penggunaan platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual. Sebelum mengisi kuesioner, para responden telah diminta untuk mencoba terlebih dahulu dengan dipandu dalam menggunakan platform ini. Para responden dibagi menjadi dua jenis pengelompokan 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)

Responden pada kuesioner ini berjumlah 20 orang yang terdiri dari 15 responden merupakan seorang yang memahami konsep pembangkit listrik virtual serta 5 responden merupakan seorang yang tidak memahami konsep pembangkit listrik virtual (orang awam). Tampilan kuesioner yang diberikan kepada responden dapat dilihat pada Lampiran 1.

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:

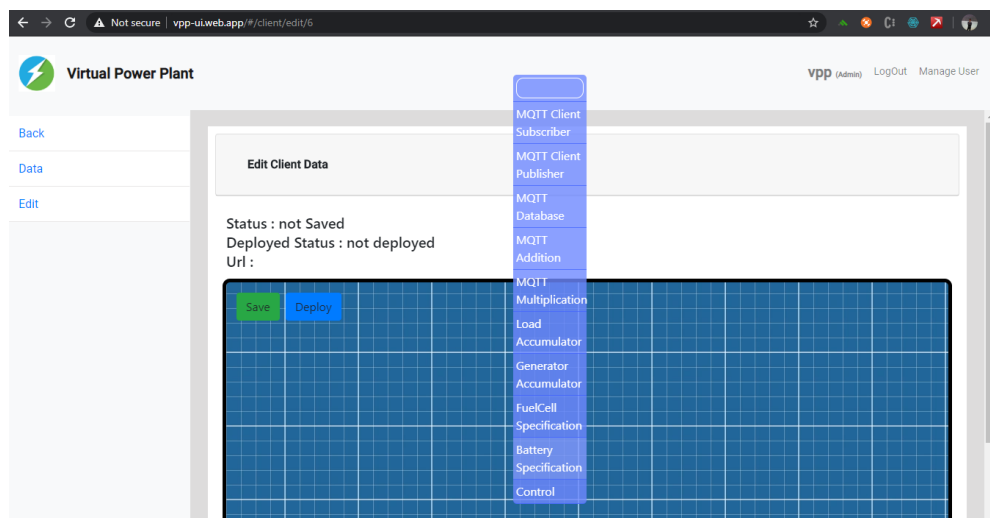
1. Tampilan antarmuka pada setiap klien secara keseluruhan menarik.
2. Tampilan *node editor* secara keseluruhan mudah dipahami dan menarik.
3. Kejelasan detail informasi yang ditampilkan pada tabel dan grafik.

4.3. 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*.

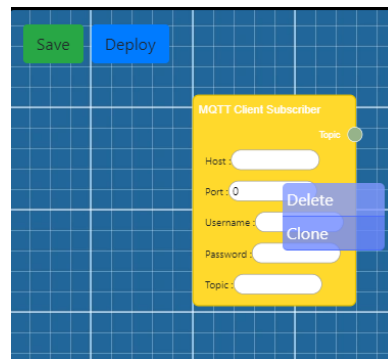
4.3.1. Hasil dan Analisis Pengujian *Node Editor*

Untuk mendukung platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual, terdapat menu untuk menambahkan *node* ketika melakukan konfigurasi *node editor*. Proses menambahkan *node* tersebut dapat dilakukan dengan mengklik kanan pada halaman editor sehingga akan menampilkan menu yang berisi daftar *node* yang tersedia. Kemudian *node* tersebut dapat dipilih sesuai kebutuhan dengan cara mengkliknya. Daftar *node* yang dapat dipilih terlihat pada Gambar 4.19.



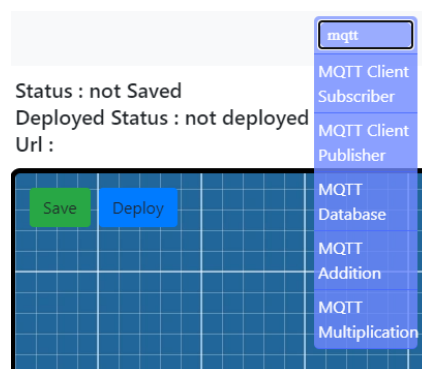
Gambar 4.19. Tampilan Menu pada *Node Editor*

Pada perancangan *node editor*, juga terdapat fitur untuk menghapus (*delete*) dan menduplikasi (*clone*) *node* yang telah dipilih untuk mendukung platform ini. Untuk menghapus dan menduplikasi *node* dapat dilakukan dengan cara mengklik kanan pada *node* yang diinginkan. Untuk lebih memperjelas data dapat dilihat pada Gambar 4.20.



Gambar 4.20. Fitur *Delete* dan *Clone* pada *Node Editor*

Fitur pencarian *node* (*search*) juga terdapat dalam platform ini. Fitur ini dapat mempermudah pengguna dalam mencari *node editor* yang diinginkan sehingga tidak perlu mencari secara satu per satu. Pencarian *node* dilakukan dengan menuliskan nama *node editor* yang diinginkan pada kolom yang tersedia di menu. Fitur *search* ini dapat dilihat pada Gambar 4.21.

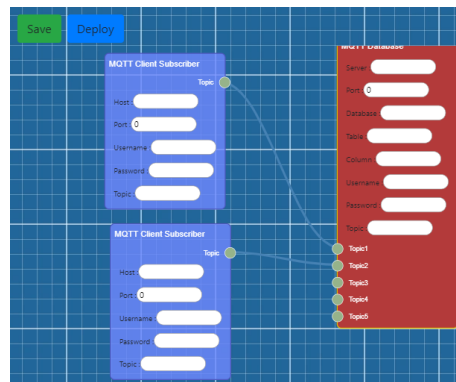


Gambar 4.21. Fitur *Search* pada *Node Editor*

Node editor dapat dibesarkan dan dikecilkan ukurannya sesuai keinginan pengguna. Hal tersebut bertujuan untuk dapat membantu mengurangi ruang pada halaman *node editor* jika terdapat banyak *node* yang ingin ditambahkan. Proses tersebut dapat dilakukan dengan melakukan *scroll up & down* yang dibuktikan pada Gambar 4.22.

Gambar 4.22. *Node* Dibesarkan atau Dikecilkan

Node editor memiliki komponen *socket* untuk membentuk koneksi antara *input* dari *node* dengan *output* dari *node* lain. Komponen ini dapat mendukung pembuatan konfigurasi pada *node editor* yang dapat dilihat pada Gambar 4.23.

Gambar 4.23. *Node* Dihubungkan dengan *Node* Lain

Berdasarkan pengujian *node editor*, menunjukkan bahwa perancangan *node* dengan menggunakan *framework* Rete.js sebagai *visual programming* dapat memberikan fitur - fitur dan komponen yang mendukung platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual sehingga dapat berjalan sesuai rancangan. Untuk merangkum pengujian *node editor* yang telah dilakukan, maka dapat dilihat pada Tabel 4.1.

Tabel 4.1. Pengujian *Node Editor*

Parameter	Berhasil	Gagal
<i>Node</i> dapat ditambahkan melalui menu yang tersedia	✓	
<i>Node</i> dapat dihapus (<i>delete</i>) dan diduplikasi (<i>clone</i>)	✓	
<i>Node</i> dapat dicari melalui fitur <i>search</i>	✓	

<i>Node</i> dapat dibesarkan atau dikecilkan	✓	
<i>Node</i> dapat dihubungkan dengan <i>node</i> lain	✓	

4.3.2. 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. Kuesioner diberikan kepada 20 responden mengenai penilaian mereka terhadap 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. *Mean of Survey* digunakan untuk mengetahui nilai yang mewakili sekumpulan data sampel pada suatu parameter sedangkan standar deviasi digunakan untuk mencari dan mengukur persebaran nilai terhadap nilai rata-rata dari sampel (*Mean of Survey*). Untuk mencari standar deviasi dapat dilihat pada Persamaan 4.1 [24].

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

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. *Mean of Survey* dan standar deviasi yang sudah didapatkan kemudian akan diolah lagi sehingga menghasilkan *confidence interval* untuk setiap parameter penilaian. Dalam menghitung *confidence interval* dapat dilihat pada Persamaan 4.2 [25].

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

Keterangan:

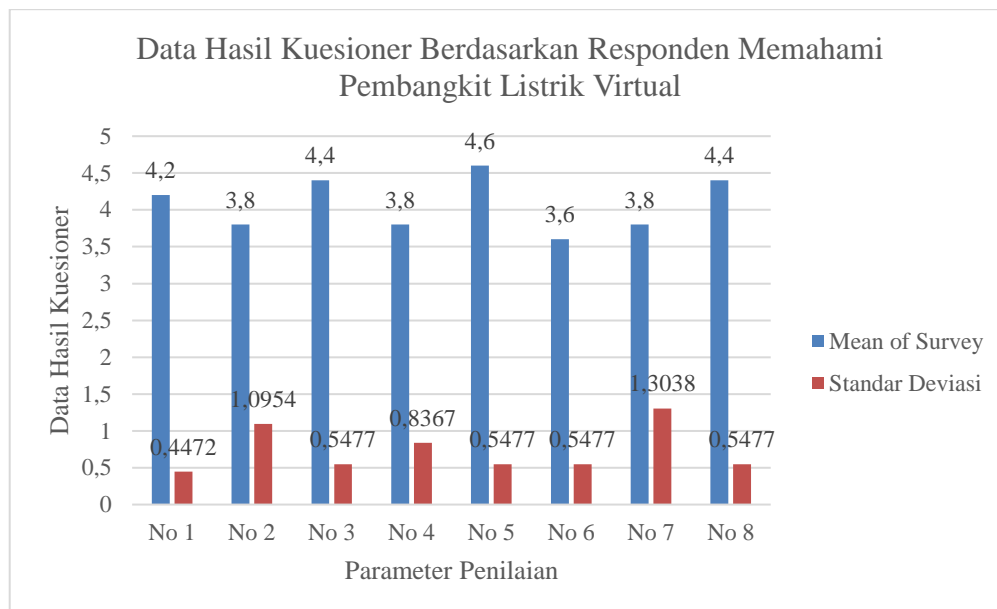
- $CI = \text{confidence interval}$
- $T = \text{nilai T}$
- $\alpha = \text{alpha}$
- $\sigma = \text{standar deviasi}$
- $\bar{x} = \text{rata-rata}$
- $n = \text{jumlah sampel}$

Dalam analisis ini, nilai T pada *confidence interval* dapat dicari pada tabel *T-Distribution* dengan menggunakan *degrees of freedom* (df) = $n - 1$ dan *confidence level* untuk mencari nilai $\alpha = 1 - \text{confidence level}$. Untuk lebih memperjelas, tabel *T-Distribution* dapat dilihat pada Lampiran 2 [26]. 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*.

4.3.2.1. Responden Memahami Pembangkit Listrik Virtual

Berdasarkan hasil kuesioner dengan 20 responden terdapat 5 responden yang memahami konsep pembangkit listrik virtual. Dari 5 responden tersebut, terdapat dua bagian penilaian pada kuesioner yaitu tentang penggunaan *node editor* dan tampilan antarmuka pada setiap klien. Untuk lebih memperjelas, data hasil kuesioner berdasarkan responden memahami konsep pembangkit listrik virtual dapat dilihat pada Lampiran 3.

Dengan jumlah data sampel yaitu 5 responden, data hasil kuesioner mengenai penilaian terhadap penggunaan *node editor* dan tampilan antarmuka pada setiap klien dapat dilihat pada Gambar 4.24.



Gambar 4.24. Grafik Data Hasil Kuesioner untuk Penilaian Berdasarkan Responden Memahami Pembangkit Listrik Virtual

Pada Gambar 4.24, menunjukkan bahwa data tersebut merupakan *Mean of Survey* (MoS) dan standar deviasi dari tiap parameter yang dihasilkan dari penilaian terhadap penggunaan *node editor* dan tampilan antarmuka pada setiap klien berdasarkan kuesioner yang diberikan kepada 5 responden. Kuesioner diisi oleh responden yang memahami konsep pembangkit listrik virtual serta responden sudah mencoba untuk mengoperasikan platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual dengan dipandu oleh penulis.

Parameter yang dinilai akan menghasilkan *Mean of Survey* dan standar deviasi yang digunakan untuk menghitung *confidence interval* pada setiap parameter penilaian. Dalam menghitung *confidence interval* maka digunakan *confidence level* 80% dengan *df* bernilai 4 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. Berdasarkan *Mean of Survey* dan standar deviasi yang sudah diolah maka menghasilkan *confidence interval* $4,2 \pm 0,3066$ dengan rentang 3,8934 - 4,5066. Berdasarkan *Mean of Survey* untuk parameter ini yaitu 4,2 maka dapat dikategorikan “Setuju”. Hal ini menunjukkan bahwa alur kerja pada platform ini mudah dipahami dan dipelajari bagi para responden yang sudah memahami konsep pembangkit listrik virtual 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. *Mean of Survey* dan standar deviasi yang sudah didapatkan kemudian digunakan untuk menghitung *confidence interval*. *Confidence interval* yang dihasilkan yaitu $3,8 \pm 0,751$ dengan rentang 3,049 – 4,551. Berdasarkan *Mean of Survey* 3,8 maka dapat dikategorikan “Cukup”. Hal ini menunjukkan bahwa fitur yang terdapat pada *node editor* seperti *search*, *delete* dan *clone* cukup dapat dipahami dan dipelajari namun bagi beberapa responden kurang dapat dipahami karena diperlukan pedoman atau 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. Dengan *Mean of Survey* dan standar deviasi maka menghasilkan *confidence interval* $4,4 \pm 0,3755$ dengan rentang 4,0245 – 4,7755. Berdasarkan *Mean of Survey* 4,4 maka dapat dikategorikan “Setuju”. Hal ini menunjukkan bahwa responden merasa terbantu dengan adanya *node editor* yang dapat mendukung konfigurasi pada platform 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. *Mean of Survey* dan standar deviasi yang sudah diolah kemudian akan dihitung *confidence interval* yang menghasilkan nilai $3,8 \pm 0,5736$ dengan rentang 3,2264 – 4,3736. Berdasarkan *Mean of Survey* 3,8 maka dapat dikategorikan “Cukup”. Hal ini menunjukkan bahwa penggunaan *node editor* kurang dapat dipahami atau dipelajari dengan singkat bagi beberapa responden yang sudah memahami konsep pembangkit listrik virtual karena diperlukan panduan atau 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. Dengan *Mean of Survey* dan standar deviasi maka menghasilkan *confidence interval* $4,6 \pm 0,3755$ dengan rentang 4,2245 – 4,9755. Berdasarkan *Mean of Survey* 4,6 maka dapat dikategorikan “Setuju”. Hal ini menunjukkan bahwa responden memerlukan *training* dalam mengoperasikan *node editor* pada platform 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. Berdasarkan *Mean of Survey* dan standar deviasi maka menghasilkan *confidence interval* $3,6 \pm 0,3755$ dengan rentang 3,2245 – 3,9755. Berdasarkan *Mean of Survey* 3,6 maka dapat dikategorikan “Cukup”. Hal ini menunjukkan 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. Dengan *Mean of Survey* dan standar deviasi maka menghasilkan *confidence interval* $3,8 \pm 0,8939$ dengan rentang 2,9061 – 4,6939. Berdasarkan *Mean of Survey* 3,8 maka dapat dikategorikan “Cukup”. Hal ini menunjukkan bahwa tampilan *node editor* secara keseluruhan cukup dapat dipahami dan menarik namun beberapa responden yang sudah memahami konsep pembangkit listrik virtual 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. Dengan *Mean of Survey* dan standar deviasi maka menghasilkan *confidence interval* $4,4 \pm 0,3755$ dengan rentang 4,0245 – 4,7755. Berdasarkan *Mean of Survey* 4,4 maka dapat dikategorikan “Setuju”. Hal ini menunjukkan bahwa responden dapat memahami dengan jelas detail informasi yang ditampilkan pada tabel dan grafik seperti daya yang dihasilkan oleh pembangkit, daya pada beban, kapasitas baterai, serta total energi yang didapatkan dari akumulasi daya per hari pada beban dan pembangkit.

Untuk lebih memperjelas, *confidence interval* pada setiap parameter dapat dilihat pada Tabel 4.2.

Tabel 4.2. *Confidence Interval* Berdasarkan Responden Memahami Pembangkit Listrik Virtual

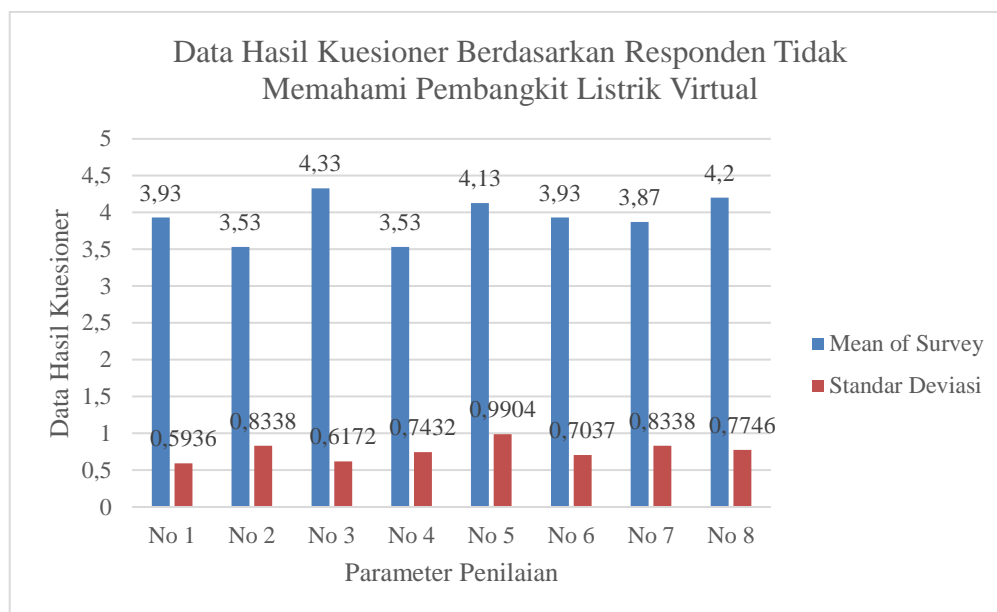
No	Parameter Penilaian	<i>Mean of Survey</i>	<i>Margin of Error</i>	<i>Confidence Interval</i>
1.	Mampu mempelajari alur kerja dari platform dengan singkat	4,2	0,3066	3,8934 - 4,5066
2.	Mampu memahami fitur yang terdapat pada <i>node editor</i>	3,8	0,751	3,049 – 4,551
3.	Penggunaan <i>node editor</i> dapat mendukung konfigurasi pada platform	4,4	0,3755	4,0245 – 4,7755

4.	Mampu mempelajari penggunaan <i>node editor</i> dengan singkat	3,8	0,5736	3,2264 – 4,3736
5.	Dibutuhkan training dalam mengoperasikan <i>node editor</i>	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 <i>node editor</i> 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

4.3.2.2. Responden Tidak Memahami Pembangkit Listrik Virtual

Berdasarkan hasil kuesioner dengan 20 responden terdapat 15 responden yang tidak memahami konsep pembangkit listrik virtual. Dari 15 responden tersebut, terdapat dua bagian penilaian pada kuesioner yaitu tentang penggunaan *node editor* dan tampilan antarmuka pada setiap klien. Untuk lebih memperjelas, data hasil kuesioner berdasarkan responden tidak memahami konsep pembangkit listrik virtual (orang awam) dapat dilihat pada Lampiran 4.

Dengan jumlah data sampel yaitu 15 responden, data hasil kuesioner mengenai penilaian terhadap penggunaan *node editor* dan tampilan antarmuka pada setiap klien dapat dilihat pada Gambar 4.25.



Gambar 4.25. Grafik Data Hasil Kuesioner untuk Penilaian Berdasarkan Responden Tidak Memahami Pembangkit Listrik Virtual

Pada Gambar 4.25, menunjukkan bahwa data tersebut merupakan *Mean of Survey* dan standar deviasi dari tiap parameter yang dihasilkan dari penilaian terhadap penggunaan *node editor* dan tampilan antarmuka pada setiap klien berdasarkan kuesioner yang diberikan kepada 15 responden. Kuesioner diisi oleh responden yang tidak memahami konsep pembangkit listrik virtual (orang awam) serta responden sudah mencoba untuk mengoperasikan platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual dengan dipandu oleh penulis.

Mean of Survey dan standar deviasi yang telah diolah akan digunakan untuk menghitung *confidence interval* pada setiap parameter. Dalam mencari *confidence interval* maka digunakan *confidence level* 80% dengan *df* bernilai 14 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. Berdasarkan *Mean of Survey* dan standar deviasi yang sudah diolah maka menghasilkan *confidence interval* $3,93 \pm 0,2061$ dengan rentang 3,7272 - 4,1395. Berdasarkan *Mean of Survey* untuk parameter ini yaitu 3,93 maka dapat dikategorikan “Cukup”. Hal ini menunjukkan 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. *Mean of Survey* dan standar deviasi yang sudah didapatkan kemudian digunakan untuk menghitung *confidence interval*. *Confidence interval* yang dihasilkan yaitu $3,53 \pm 0,2896$ dengan rentang 3,2438 – 3,8229. Berdasarkan *Mean of Survey* 3,53 maka dapat dikategorikan “Cukup”. Hal ini menunjukkan bahwa fitur yang terdapat pada *node editor* seperti *search*, *delete* dan *clone* kurang dapat dipahami dan dipelajari bagi beberapa responden karena diperlukan pedoman atau 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. Dengan *Mean of Survey* dan standar deviasi maka menghasilkan *confidence interval* $4,33 \pm 0,2143$ dengan rentang 4,119 – 4,5477. Berdasarkan *Mean of Survey* 4,33 maka dapat dikategorikan “Setuju”. Hal ini menunjukkan 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. *Mean of Survey* dan standar deviasi yang sudah diolah kemudian akan dihitung *confidence interval* yang menghasilkan nilai $3,53 \pm 0,2581$ dengan rentang 3,2752 – 3,7914. Berdasarkan *Mean of Survey* 3,53 maka dapat dikategorikan “Cukup”. Hal ini menunjukkan bahwa penggunaan *node editor* kurang dapat dipahami dengan singkat bagi responden sebagai orang awam karena diperlukan panduan atau 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. Dengan *Mean of Survey* dan standar deviasi maka menghasilkan *confidence interval* $4,13 \pm 0,344$ dengan rentang 3,7894 – 4,4773. Berdasarkan *Mean of Survey* 4,13 maka dapat dikategorikan “Setuju”. Hal ini menunjukkan 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. Berdasarkan *Mean of Survey* dan standar deviasi maka menghasilkan *confidence interval* $3,93 \pm 0,2444$ dengan rentang 3,6889 – 4,1777. Berdasarkan *Mean of Survey* 3,93 maka dapat dikategorikan “Cukup”. Hal ini menunjukkan 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. Dengan *Mean of Survey* dan standar

deviasi maka menghasilkan *confidence interval* $3,87 \pm 0,2896$ dengan rentang 3,5771 – 4,1562. Berdasarkan *Mean of Survey* 3,87 maka dapat dikategorikan “Cukup”. Hal ini menunjukkan 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. Dengan *Mean of Survey* dan standar deviasi maka menghasilkan *confidence interval* $4,2 \pm 0,269$ dengan rentang 3,931 – 4,469. Berdasarkan *Mean of Survey* 4,2 maka dapat dikategorikan “Setuju”. Hal ini menunjukkan bahwa responden dapat memahami dengan jelas detail informasi yang ditampilkan pada tabel dan grafik seperti daya yang dihasilkan oleh pembangkit, daya pada beban, kapasitas baterai, serta total energi yang didapatkan dari akumulasi daya per hari pada beban dan pembangkit.

Untuk lebih memperjelas, *confidence interval* pada setiap parameter dapat dilihat pada Tabel 4.3.

Tabel 4.3. *Confidence Interval* Berdasarkan Responden Tidak Memahami Pembangkit Listrik Virtual

No	Parameter Penilaian	<i>Mean of Survey</i>	<i>Margin of Error</i>	<i>Confidence Interval</i>
1.	Mampu mempelajari alur kerja dari platform dengan singkat	3,93	0,2061	3,7272 - 4,1395
2.	Mampu memahami fitur yang terdapat pada <i>node editor</i>	3,53	0,2896	3,2438 – 3,8229
3.	Penggunaan <i>node editor</i> dapat mendukung konfigurasi pada platform	4,33	0,2143	4,119 – 4,5477

4.	Mampu mempelajari penggunaan <i>node editor</i> dengan singkat	3,53	0,2581	3,2752 – 3,7914
5.	Dibutuhkan training dalam mengoperasikan <i>node editor</i>	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 <i>node editor</i> 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

4.3.2.3. Hasil dan Analisis Keseluruhan *Usability Testing*

Menurut analisis yang sudah dilakukan, maka dapat disimpulkan bahwa keseluruhan *Mean of Survey* (MoS) berdasarkan responden memahami pembangkit listrik virtual yaitu 4,1 dalam skala 5 sedangkan untuk responden tidak memahami pembangkit listrik virtual yaitu 3,93 dalam skala 5. Hal ini menunjukkan bahwa keseluruhan platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual dalam segi penggunaan *node editor* dan tampilan antarmuka pada setiap klien dapat dikatakan cukup baik oleh para responden. Dengan kata lain responden merasa cukup puas dengan pengalaman dalam mengoperasikan platform ini. Namun responden baik yang memahami maupun yang tidak memahami pembangkit listrik virtual memerlukan *training* atau panduan dalam mengoperasikan platform ini.

Berdasarkan *confidence interval* yang didapatkan maka menunjukkan bahwa *Mean of Survey* yang diperoleh dari perhitungan akan berada dalam interval tersebut dengan tingkat kepercayaan sebesar 80%. *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 – 0,344.

BAB 5

PENUTUP

Bab ini akan membahas mengenai kesimpulan sesuai dengan apa yang dikerjakan dalam skripsi berdasarkan perancangan, pemodelan, pengujian dan analisis sistem yang telah dilakukan. Selain itu, juga akan menjelaskan tentang saran untuk pengembangan lebih lanjut yang dapat dilakukan pada platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual.

5.1. Kesimpulan

Berdasarkan hasil perancangan, pemodelan, pengujian dan analisis yang telah dilakukan maka didapatkan beberapa kesimpulan yaitu:

1. 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.
2. Penulis telah berhasil membangun sistem pemantauan untuk membantu klien dalam melihat seluruh aktivitas pembangkit dan beban meliputi daya 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.
3. Penilaian penggunaan 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 sedangkan untuk responden tidak memahami pembangkit listrik virtual yaitu 3,93.
4. *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 – 0,344 untuk responden tidak memahami pembangkit listrik virtual.

5.2. Saran untuk Pengembangan Selanjutnya

Apabila platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual ingin dikembangkan lebih lanjut maka dapat dilakukan dengan menciptakan komponen atau fitur baru dalam sistem agar platform ini menjadi lebih baik dari segi tampilan antarmuka dan alur kerja pada sistem. Pengembangan lebih lanjut yang dapat dilakukan pada platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual 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.

Untuk mendukung pengembangan tersebut dapat dilakukan dengan mengekstensi program. Salah satu cara untuk mendukung pengembangan lebih lanjut pada platform sebagai sistem pemantauan untuk pemodelan pembangkit listrik virtual yaitu dengan membuat *node* baru. Setiap pembuatan *node* baru terdapat dua file yang perlu dimodifikasi yaitu file *component rete* yang mendefinisikan tampilan antarmuka pada setiap *node* dengan menentukan *built-in components* yang akan digunakan seperti *socket*, *input*, *output* dan *control* serta file *task* yang mendefinisikan tentang apa yang akan dilakukan oleh suatu *node* dengan membuat fungsi khusus untuk setiap *node*.

DAFTAR PUSTAKA

- [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] S. Ghavidel, L. Li, J. Aghaei, T. Yu and J. Zhu, “A Review on The Virtual Power Plant: Components and Operation Systems”, IEEE International Conference on Power System Technology (POWERCON), 2016. Available: 10.1109/powercon.2016.7754037.
- [5] Ahmad, Adnan, A. Khan, N. Javaid, H. M. Hussain, W. Abdul, A. Almogren, A. Alamri and I. A. Niaz, “An Optimized Home Energy Management System with Integrated Renewable Energy and Storage Resources”, Energies 10, no. 4: 549, 2017.
- [6] Eco2Solar. “How Does Solar PV Work”. [Online]. Available: <https://www.eco2solar.co.uk/solar-electricity/how-does-solar-pv-work/#> [Accessed: 31 May 2020]
- [7] S. Aslam, “An Optimal Home Energy Management Scheme Considering Grid Connected Microgrids with Day-ahead Weather Forecasting using Artificial Neural Network”, MS Thesis in Computer Science, COMSATS University Islamabad, 2018.
- [8] Jansen T.J., “Teknologi Rekayasa Sel Surya”, PT Pradnya Paramita, Jakarta, 1995.

- [9] J. Balakrishnan, "Fuel cell technology", Third International Conference On Information And Automation For Sustainability, 2007. Available: 10.1109/iciafs.2007.4544796
- [10] J. Larminie and A. Dicks, "Fuel Cell Systems Explained", 2003. Available: 10.1002/9781118878330.
- [11] S. Aslam, R. Bukhsh, A. Khalid, N. Javaid, I. Ullah, I. Fatima and Q. U. Hasan, "An Efficient Home Energy Management Scheme Using Cuckoo Search", In International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, pp. 167-178. Springer, Cham, 2017.
- [12] M. W. Habibi, A. Bhawiyuga and A. Basuki, "Rancang Bangun IoT Cloud Platform Berbasis Protokol Komunikasi MQTT", Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer. 2018; vol. 2: p. 7, 2018.
- [13] D. Zubov, "An IoT Concept of The Small Virtual Power Plant Based on Arduino Platform and MQTT Protocol", Proceedings of The ICAIIT2016, 2016. Available: 10.20544/aiit2016.13
- [14] M. Rouse. "What is MQTT (MQ Telemetry Transport)?". [Online]. Available: <https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport> [Accessed: 31 May 2020]
- [15] Steves. "MQTT Protocol Messages Overview". [Online]. Available: <http://www.steves-internet-guide.com/mqtt-protocol-messages-overview/> [Accessed: 31 May 2020]
- [16] OpenJS Foundation. "Introduction to Node.js". [Online]. Available: <https://nodejs.org/en/docs/> [Accessed: 31 May 2020]
- [17] OpenJS Foundation. "Fast, Unopinionated, Minimalist Web Framework for Node.js". [Online]. Available: <https://expressjs.com/> [Accessed: 31 May 2020]

- [18] Google Foundation. “Introduction of AngularJS”. [Online]. Available: <https://angularjs.org/> [Accessed: 31 May 2020]
- [19] Tutorialspoint. “AngularJS Overview”. Available: https://www.tutorialspoint.com/angularjs/angularjs_overview.htm [Accessed: 31 May 2020]
- [20] Vitaliy Stoliarov. “Introduction of Rete.js”. Available: <https://rete.js.org/#/docs> [Accessed: 31 May 2020]
- [21] Timescale Inc. “Introduction of TimescaleDB”. [Online]. Available: <https://docs.timescale.com/latest/introduction> [Accessed: 31 May 2020]
- [22] The GraphQL Foundation. “Introduction to GraphQL”. [Online]. Available: <https://graphql.org/learn/> [Accessed: 1 June 2020]
- [23] Hasura Inc. “What is Hasura?”. [Online]. Available: <https://hasura.io/> [Accessed: 1 June 2020]
- [24] 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]
- [25] 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]
- [26] San Jose State University. “T-table”. [Online]. Available: <https://www.sjsu.edu/faculty/gerstman/StatPrimer/t-table.pdf> [Accessed: 23 June 2020]

LAMPIRAN

Lampiran 1. Tampilan Kuesioner Penilaian terhadap Platform untuk Pemodelan Pembangkit Listrik Virtual

Kuesioner Platform untuk Pemodelan Pembangkit Listrik Virtual

Data Responden

Nama :

Pekerjaan atau Latar Belakang :

Apakah Anda telah memahami konsep pembangkit listrik virtual? (Ya/Tidak)

No	Parameter Penilaian	Skala Penilaian				
		1 (STS)	2 (KS)	3 (C)	4 (S)	5 (SS)
Penggunaan <i>node editor</i>						
1.	Mampu mempelajari alur kerja dari platform dengan singkat					
2.	Mampu memahami fitur yang terdapat pada <i>node editor</i>					
3.	Penggunaan <i>node editor</i> dapat mendukung konfigurasi pada platform					
4.	Mampu mempelajari penggunaan <i>node editor</i> dengan singkat					
5.	Dibutuhkan <i>training</i> dalam mengoperasikan <i>node editor</i>					
Tampilan antarmuka pada setiap klien						
6.	Tampilan antarmuka pada setiap klien secara keseluruhan menarik					
7.	Tampilan <i>node editor</i> secara keseluruhan mudah dipahami dan menarik					

Lampiran 3. Data Hasil Kuesioner Berdasarkan Responden Memahami Konsep Pembangkit Listrik Virtual

Responden	Penggunaan <i>node editor</i>					Tanmpilan antarmuka pada setiap klien		
	No.1	No.2	No.3	No.4	No.5	No.6	No.7	No.8
1	5	5	5	5	4	4	5	5
2	4	4	4	4	5	3	5	4
3	4	2	5	3	5	3	2	4
4	4	4	4	3	5	4	3	5
5	4	4	4	4	4	4	4	4
Rata-rata	4,2	3,8	4,4	3,8	4,6	3,6	3,8	4,4
Standar Deviasi	0,4472	1,0954	0,5477	0,8367	0,5477	0,5477	1,3038	0,5477

Lampiran 4. Data Hasil Kuesioner Berdasarkan Responden Tidak Memahami Konsep Pembangkit Listrik Virtual

Responden	Penggunaan <i>node editor</i>					Tanmpilan antarmuka pada setiap klien		
	No.1	No.2	No.3	No.4	No.5	No.6	No.7	No.8
6	4	3	4	4	5	4	4	5
7	4	2	4	2	5	5	3	5
8	4	3	5	3	5	4	3	3
9	5	3	5	4	3	4	4	4
10	5	5	5	4	3	5	5	4
11	4	3	4	3	5	3	3	4
12	4	4	4	4	3	4	5	5
13	4	4	4	4	5	4	4	4
14	4	5	5	4	5	5	5	5
15	4	4	5	3	3	3	4	5
16	3	4	5	5	3	4	3	3

17	3	3	4	3	4	3	5	4
18	4	3	3	3	5	4	3	3
19	3	3	4	4	5	4	3	4
20	4	4	4	3	3	3	4	5
Rata-rata	3,93	3,53	4,33	3,53	4,13	3,93	3,87	4,2
Standar Deviasi	0,5936	0,8338	0,6172	0,7432	0,9904	0,7037	0,8338	0,7746