



UNIVERSITAS INDONESIA

**PERANCANGAN SISTEM PEMANTAUAN PADA PLATFORM
SIMULASI PEMBANGKIT LISTRIK VIRTUAL**

SEMINAR

Muhammad Djati Pradana

1606829680

**FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KOMPUTER**

**DEPOK
JULI 2020**



UNIVERSITAS INDONESIA

**PERANCANGAN SISTEM PEMANTAUAN PADA PLATFORM
SIMULASI PEMBANGKIT LISTRIK VIRTUAL**

SEMINAR

**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

**Seminar 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 : 8 Juli 2020

HALAMAN PENGESAHAN

Seminar dengan judul:

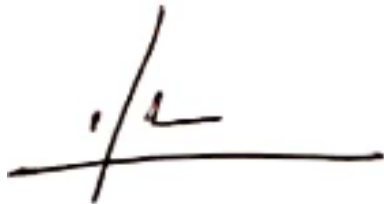
PERANCANGAN SISTEM PEMANTAUAN PADA PLATFORM SIMULASI PEMBANGKIT LISTRIK VIRTUAL

dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada Program Studi Teknik Komputer, Departemen Teknik elektro, Fakultas Teknik Universitas Indonesia dan disetujui untuk diajukan dalam presentasi seminar.

Depok, 8 Juli 2020

Dosen Pembimbing

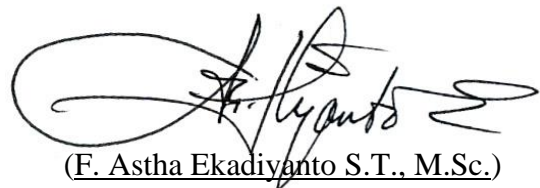
Pembimbing I,



(Dr.-Ing. Eko Adhi Setiawan S.T., M.T.)

NIP. 040803032

Pembimbing II,



(F. Astha Ekadiyanto S.T., M.Sc.)

NIP. 197210041997021002

KATA PENGANTAR

Segala puji ke hadirat Allah SWT karena berkat rahmat dan rida-Nya, penulis dapat menyelesaikan seminar yang berjudul “Perancangan Sistem Pemantauan pada Platform Simulasi Pembangkit Listrik Virtual”. Seminar ini merupakan syarat wajib bagi mahasiswa sarjana jurusan Teknik Komputer Universitas Indonesia untuk menyelesaikan studinya.

Penulis sadar bahwa dalam menyelesaikan seminar 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 seminar ini.
2. Bapak Prof. Dr.-Ing. Ir. Kalamullah Ramli, M.Eng. selaku pembimbing akademis.

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

Depok, 8 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 : Seminar

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:

**PERANCANGAN SISTEM PEMANTAUAN PADA PLATFORM
SIMULASI 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 : 8 Juli 2020
Yang menyatakan



(Muhammad Djati Pradana)

ABSTRAK

Nama : Muhammad Djati Pradana
Program Studi : Teknik Komputer
Judul : Perancangan Sistem Pemantauan pada Platform Simulasi
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. Seminar ini bertujuan untuk membuat sistem pemantauan pada platform simulasi 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. Perancangan simulasi ini menggunakan 3 klien yang terdiri dari 2 klien memiliki pembangkit dan beban serta 1 klien memiliki baterai. Selain itu, perancangan *node editor* dapat mendukung konfigurasi pada platform simulasi pembangkit listrik virtual.

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

ABSTRACT

Name : Muhammad Djati Pradana

Study Program : Computer Engineering

Title : Design of Monitoring System in Virtual Power Plant Simulation Platform

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 seminar aims to create a monitoring system in simulation platform of virtual power plant. 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 design of simulation uses 3 clients consisting of 2 clients having a generator and load while 1 client having a battery. Besides, the design of node editor can support configuration in the virtual power plant simulation platform.

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

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	x
DAFTAR SINGKATAN	xi
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.....	5
2.1. Pembangkit Listrik Virtual	5
2.2. Panel Surya	6
2.3. Sel Bahan Bakar.....	8
2.4. Baterai	9
2.5. MQTT	10
2.6. Node.js	12
2.7. AngularJS.....	13
2.8. Rete.js.....	14
2.9. TimescaleDB.....	15
2.10. GraphQL	16
2.11. Hasura	16
BAB 3 PERANCANGAN SISTEM	18
3.1. Pemodelan Konsep Pembangkit Listrik Virtual	18

3.2. Pengembangan Platform Simulasi Pembangkit Listrik Virtual	20
3.3. Perancangan <i>Node Editor</i> pada Platform Simulasi	24
3.3.1. <i>Node MQTT (Subscriber dan Publisher)</i>	25
3.3.2. <i>Node Kalkulasi (Addition dan Multiplication)</i>	26
3.3.3. <i>Node Database</i>	27
3.3.4. <i>Node Akumulator (Load dan Generator)</i>	28
3.3.5. <i>Node Spesifikasi (Battery dan Fuel cell)</i>	29
3.3.6. <i>Node Kontrol</i>	29
3.4. <i>Use Case Diagram</i> pada Platform Simulasi Pembangkit Listrik Virtual	30
3.5. Diagram Alir pada Platform Simulasi Pembangkit Listrik Virtual	31
BAB 4 PENUTUP	33
4.1. Kesimpulan	33
DAFTAR PUSTAKA	34

DAFTAR GAMBAR

Gambar 2.1. Skema Umum Sistem Pembangkit Listrik Virtual.....	5
Gambar 2.2. Proses Konversi Energi Matahari ke Listrik pada Panel Surya	7
Gambar 2.3. Grafik Radiasi Terhadap Waktu.....	8
Gambar 2.4. Proses pada <i>Fuel Cell</i>	9
Gambar 2.5. Konsep IoT dengan protokol MQTT	11
Gambar 2.6. Tampilan <i>Node Editor</i> pada Rete.js	15
Gambar 2.7. Arsitektur TimescaleDB.....	15
Gambar 3.1. Pemodelan Konsep Pembangkit Listrik Virtual.....	18
Gambar 3.2. Perancangan Sistem pada <i>Sofwan House</i> TREC FTUI.....	19
Gambar 3.3. Arsitektur Platform Simulasi Pembangkit Listrik Virtual	21
Gambar 3.4. Halaman Login pada Platform Simulasi	22
Gambar 3.5. Halaman Utama pada Platform Simulasi	22
Gambar 3.6. Halaman <i>Node Editor</i> pada Platform Simulasi	24
Gambar 3.7. <i>Node MQTT Client Subscriber dan Publisher</i>	25
Gambar 3.8. <i>Node MQTT Addition dan Multiplication</i>	26
Gambar 3.9. <i>Node MQTT Database</i>	27
Gambar 3.10. <i>Node Generator dan Load Accumulator</i>	28
Gambar 3.11. <i>Node Battery and FuelCell Specification</i>	29
Gambar 3.12. <i>Node Control</i>	29
Gambar 3.13. <i>Use Case Diagram</i> pada Platform Simulasi	30
Gambar 3.14. Diagram alir pada Platform Simulasi	31

DAFTAR SINGKATAN

API	: <i>Application Programming Interface</i>
ESS	: <i>Energy Storage System</i>
JSON	: <i>Javascript Object Notation</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 produksi energi serta aktivitas beban pada jaringan listrik sesuai dengan algoritma yang telah didefinisikan sehingga diperlukan perancangan platform simulasi pembangkit listrik virtual yang dapat mendukung hal tersebut. 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 seminar dalam jenjang pendidikan yang sedang diambil oleh penulis dan sebagai syarat wajib dalam memperoleh gelar Sarjana Teknik. Tujuan dari penelitian ini adalah sebagai berikut:

- Merancang *node editor* yang dapat membantu pengguna dalam melakukan konfigurasi pada platform simulasi pembangkit listrik virtual
- Merancang 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 sistem pemantauan pada platform simulasi pembangkit listrik virtual yang digunakan untuk mengoptimasi aktivitas pembangkit dalam memproduksi energi.

Batasan masalah yang akan dibahas dalam seminar meliputi pembuatan *node editor* untuk mempermudah konfigurasi, protokol yang digunakan sebagai komunikasi dalam pertukaran data, *database* yang digunakan untuk menyimpan rekaman data berdasarkan *timeseries*.

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

Perancangan *node editor* dan sistem pemantauan yang mendukung platform simulasi pembangkit listrik virtual.

5. Kesimpulan

Kesimpulan akan ditarik dari hasil perancangan yang telah dilakukan.

1.5. Sistematika Penulisan

Sistematika dari penulisan seminar ini dibagi menjadi empat 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 pembuatan platform simulasi, *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 seminar ini yang berisi tentang pemodelan konsep pembangkit listrik virtual, pengembangan platform simulasi, perancangan *node editor* untuk mendukung sistem komunikasi pada pembangkit listrik virtual, *use case* diagram dan diagram alir pada platform simulasi pembangkit listrik virtual.

Sebagai penutup, seluruh hasil penelitian ini disimpulkan secara ringkas pada Bab 4. Kesimpulan tersebut dapat berisi sesuai dengan apa yang dikerjakan dalam seminar berupa hasil perancangan sistem yang telah dilakukan pada platform simulasi pembangkit listrik virtual.

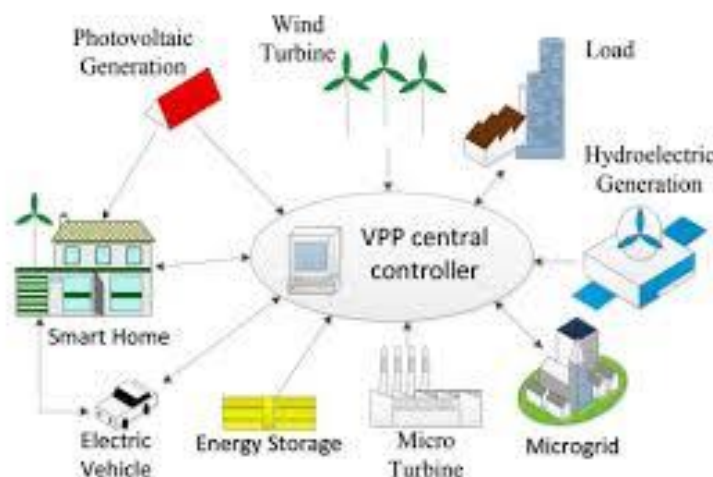
BAB 2

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 pembuatan platform simulasi, *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 logika atau algoritma yang sudah didefinisikan seperti mengagregasi energi yang dihasilkan dari sumber pembangkit energi yang terdistribusi [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 diintegrasikan sehingga menjadi satu entitas. Sehingga energi dari entitas tersebut dapat diekspor kepada *grid* atau mengurangi beban puncak di *grid* saat penggunaan pada waktu tertentu. Untuk lebih memperjelas mengenai pembangkit listrik virtual dapat dilihat pada Gambar 2.1.



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

Sumber energi yang digunakan dalam pembangkit listrik virtual sendiri dapat berasal dari energi terbarukan seperti *photovoltaic*, *fuel cell*, *geothermal*, turbin energi, *hydro station*, biogas dan lain-lain. Penggunaan baterai sebagai alat penyimpanan energi juga diperlukan dalam sistem pembangkit listrik virtual [3].

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 [4]. 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.

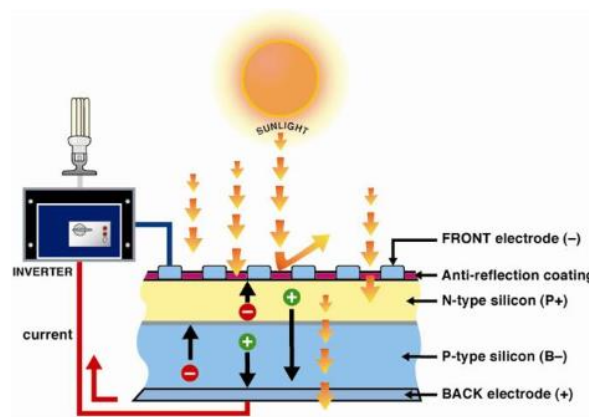
Penggunaan sistem pembangkit listrik virtual ini secara otomatis juga akan merubah proses penyediaan energi listrik dari yang semula hanya satu arah (dari PLN menuju ke konsumen) menjadi dua arah. Pada konsep dua arah, konsumen juga dapat berperan sebagai penghasil energi listrik atau yang disebut sebagai prosumer (*producer-consumer*). Sistem pembangkit listrik virtual juga dapat membantu memenuhi pasokan listrik dari sumber listrik utama sehingga dapat menjaga ketersediaan listrik bagi konsumen. 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

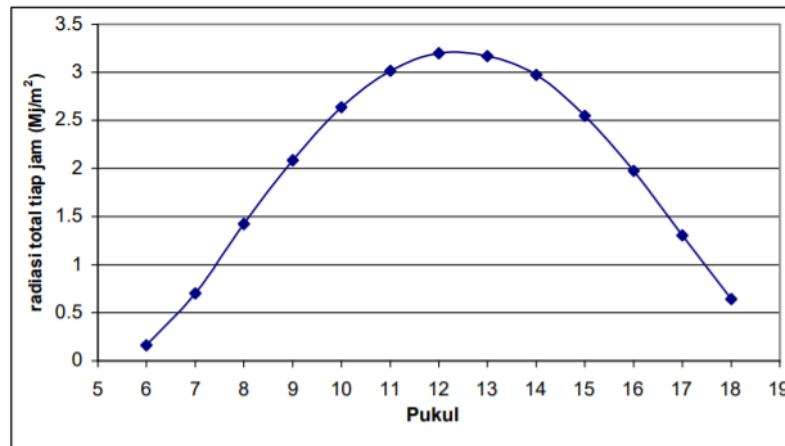
dan n. Ketika sinar matahari yang merupakan foton-foton menyinari panel surya terdapat tiga kemungkinan yang mungkin terjadi, yaitu akan diserap, dipantulkan ataupun akan dilewatkan. Sebuah foton harus memiliki tingkat energi tertentu untuk dapat melepaskan electron dan menghasilkan listrik. Batas tingkat 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 2.2.



Gambar 2.2. 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 2.3.

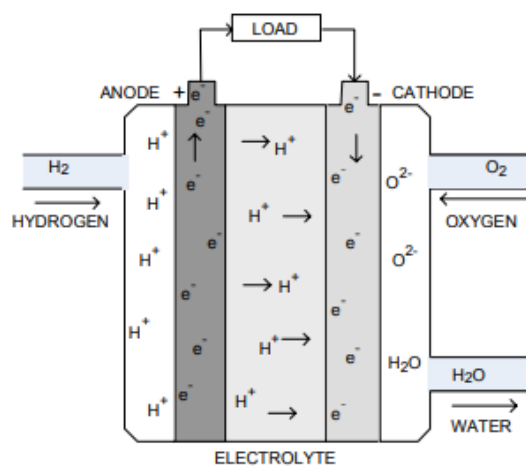


Gambar 2.3. Grafik Radiasi Terhadap Waktu [8]

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 2.4.



Gambar 2.4. 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].

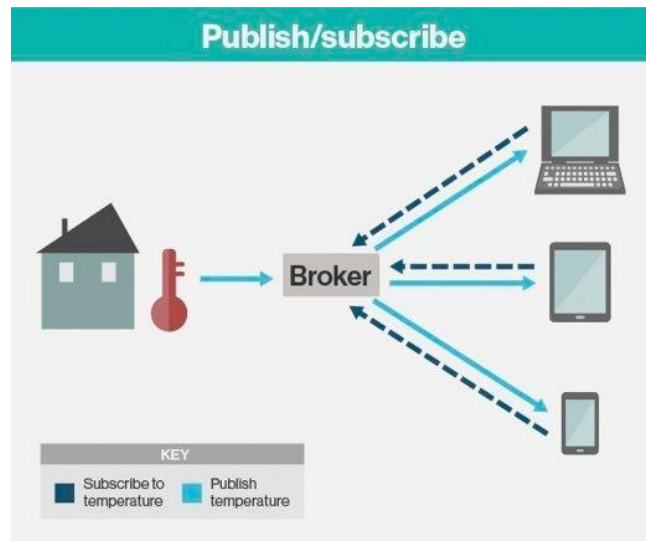
Baterai memiliki komponen utama yaitu elektroda dengan kutub positif dan negatif, elektrolit dan bahan isolasi (pemisah). Pada baterai, anoda dan katoda terbuat dari bahan yang dapat bereaksi dengan bahan elektrolitnya. Dalam proses mengalirkan energi listrik, anoda dan elektrolit akan bereaksi dengan 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. 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 2.5.



Gambar 2.5. Konsep Dasar MQTT [14]

Berdasarkan Gambar 2.5, 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 protocol 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 seminar 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

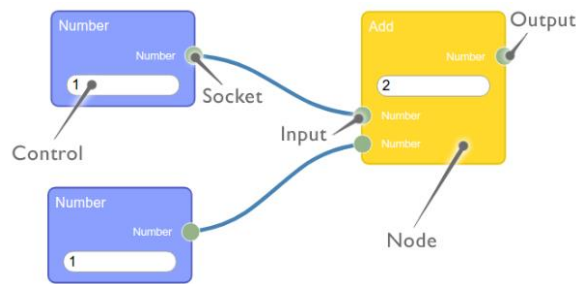
AngularJS adalah *frontend framework* Javascript untuk aplikasi web dinamis yang dikembangkan oleh Google dan bersifat *open source*. AngularJS sendiri berjalan pada sisi klien yang dapat diakses melalui browser [18]. Pada AngularJS menggunakan arsitektur *Model View Controller* (MVC). *Model* berperan 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*.

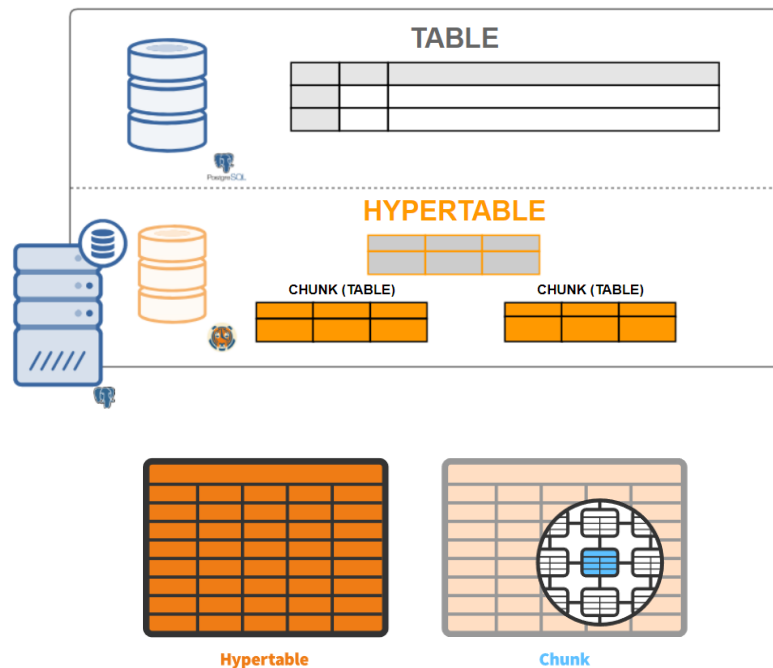
2.8. Rete.js

Rete.js merupakan *framework* Javascript yang digunakan untuk *visual programming* dan bersifat modular. Rete 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 2.6.

Gambar 2.6. 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 postgresSQL. 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 2.7.



Gambar 2.7. 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. Dengan demikian, Hasura memiliki fungsi khusus yang sudah didefinisikan untuk dapat memanfaatkan GraphQL. Hasura memiliki beberapa fitur antara lain [23]:

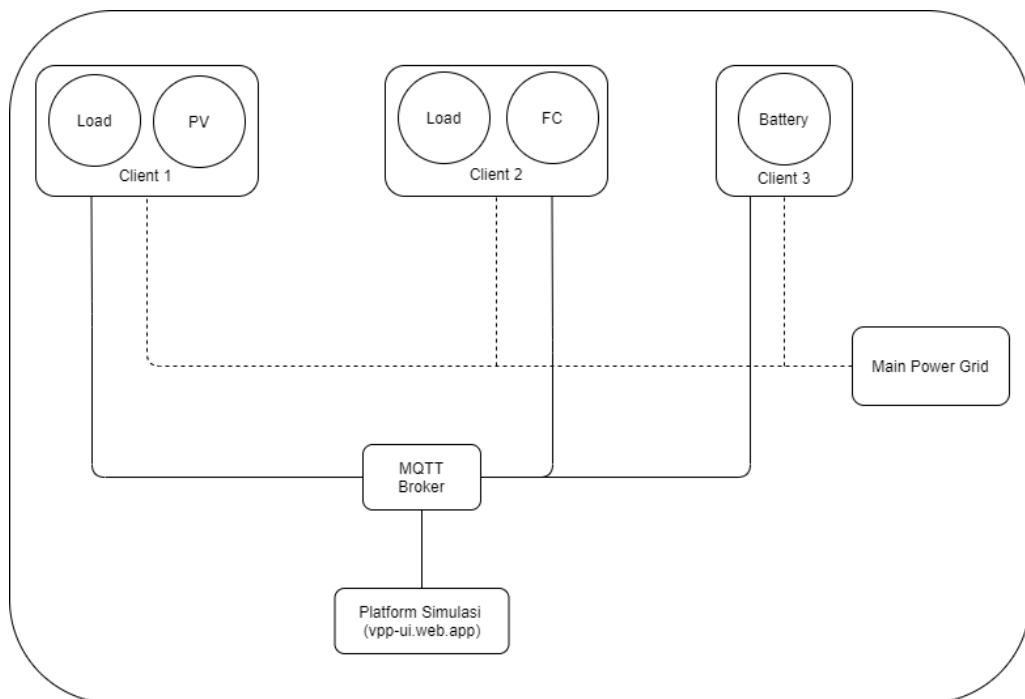
1. Memiliki query bawaan seperti *built-in filtering, pagination, pattern search, bulk insert, update, delete mutations*.
2. Hasura dapat mengubah query GraphQL menjadi *live query* dengan menggunakan fungsi *subscription*.
3. Dapat menggunakan *database* yang telah ada sebelumnya untuk mendapatkan data 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 simulasi, perancangan *node editor* untuk mendukung sistem komunikasi pada pembangkit listrik virtual, *use case diagram* dan diagram alir pada platform simulasi pembangkit listrik virtual.

3.1. Pemodelan Konsep Pembangkit Listrik Virtual



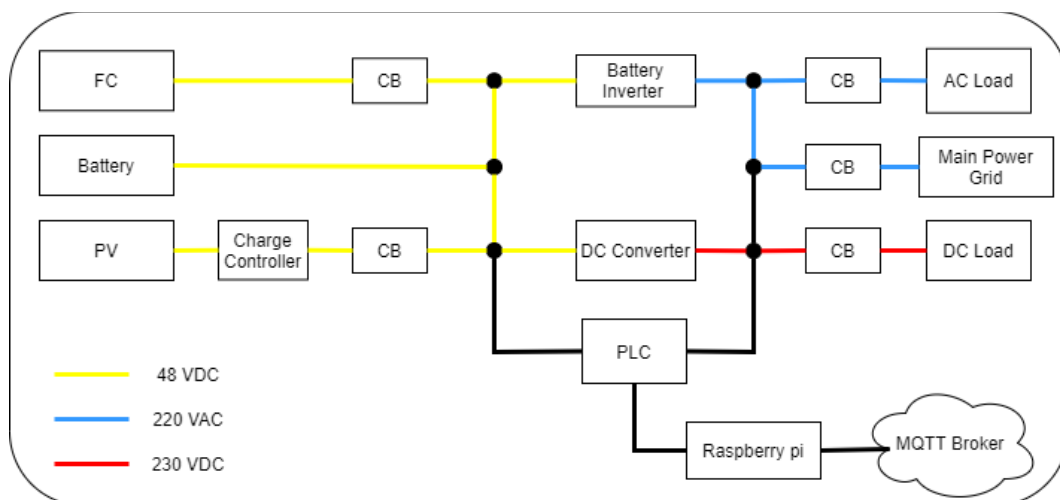
Gambar 3.1. Pemodelan Konsep Pembangkit Listrik Virtual

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 yang digunakan untuk simulasi 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 pembuatan simulasi 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 di platform simulasi yang dibuat (*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



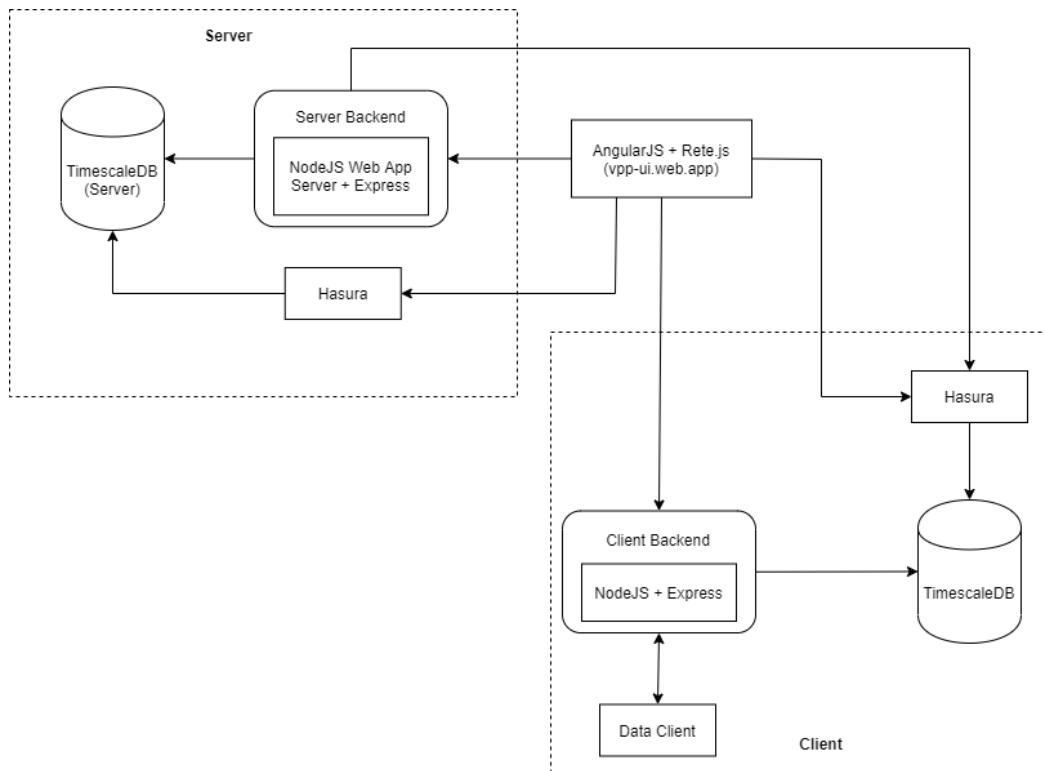
Gambar 3.2. Perancangan Sistem pada *Sofwan House TREC FTUI*

Berdasarkan Gambar 3.2, memperlihatkan perancangan sistem pada *container*. Dalam sistem di *container* terapat 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 Simulasi Pembangkit Listrik Virtual

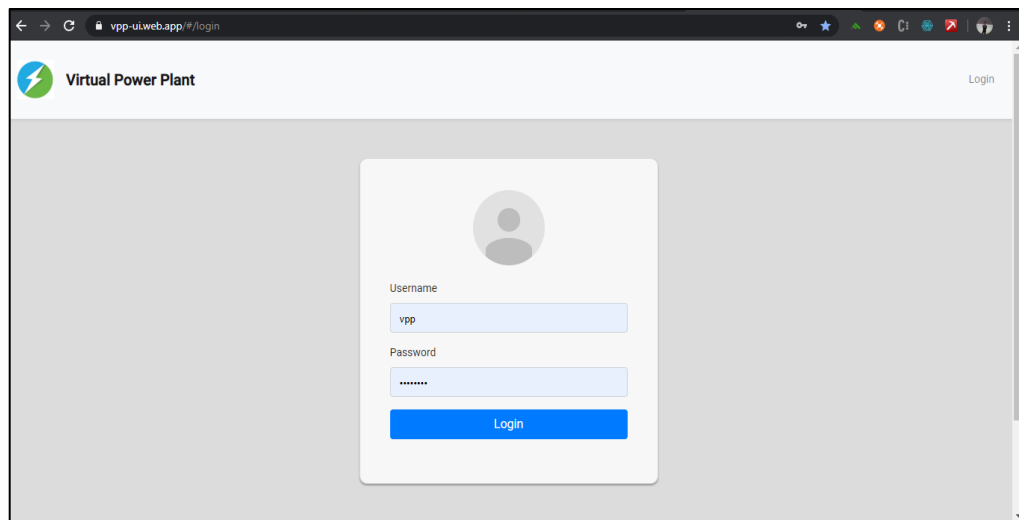
Platform simulasi ini digunakan untuk mensimulasikan konsep pembangkit listrik virtual seperti melakukan pemantauan, kendali dan juga manajemen energi berdasarkan logika yang sudah dibentuk yaitu mengagregasi daya yang dihasilkan oleh seluruh jenis pembangkit serta daya yang digunakan oleh beban sehingga 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 simulasi 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 simulasi pembangkit listrik virtual dapat dilihat pada Gambar 3.3.



Gambar 3.3. Arsitektur Platform Simulasi Pembangkit Listrik Virtual

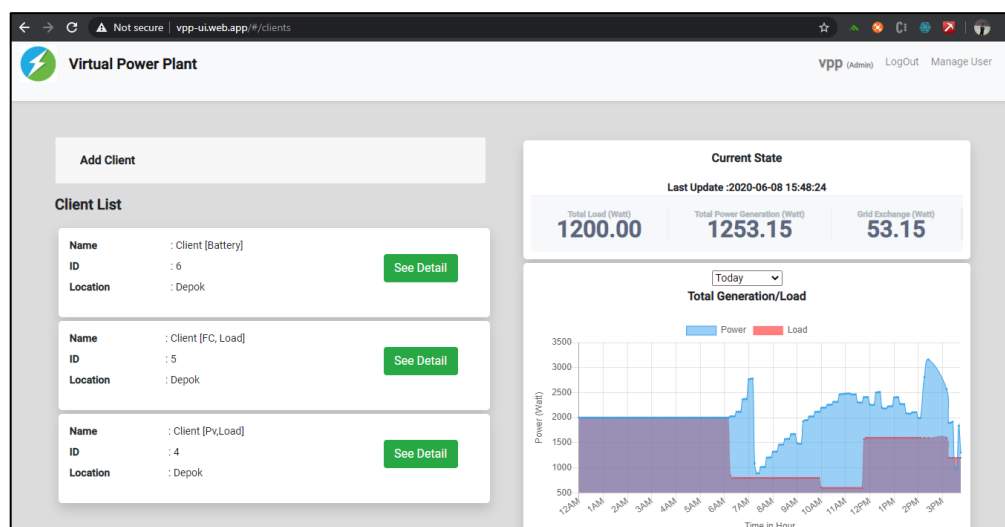
Berdasarkan Gambar 3.3, terdapat tiga bagian utama dalam platform simulasi ini yaitu server, website dan klien. Pada server memiliki *backend* yang menyediakan layanan *web service* menggunakan Node.js dan Express sehingga dapat berkomunikasi atau terhubung menggunakan RESTFUL API. Server dalam platform simulasi 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 simulasi yang dapat dilihat pada Gambar 3.4.



Gambar 3.4. Halaman Login pada Platform Simulasi

Pada platform simulasi 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 simulasi ini. Berikut ini merupakan halaman utama pada platform simulasi yang dapat dilihat pada Gambar 3.5.



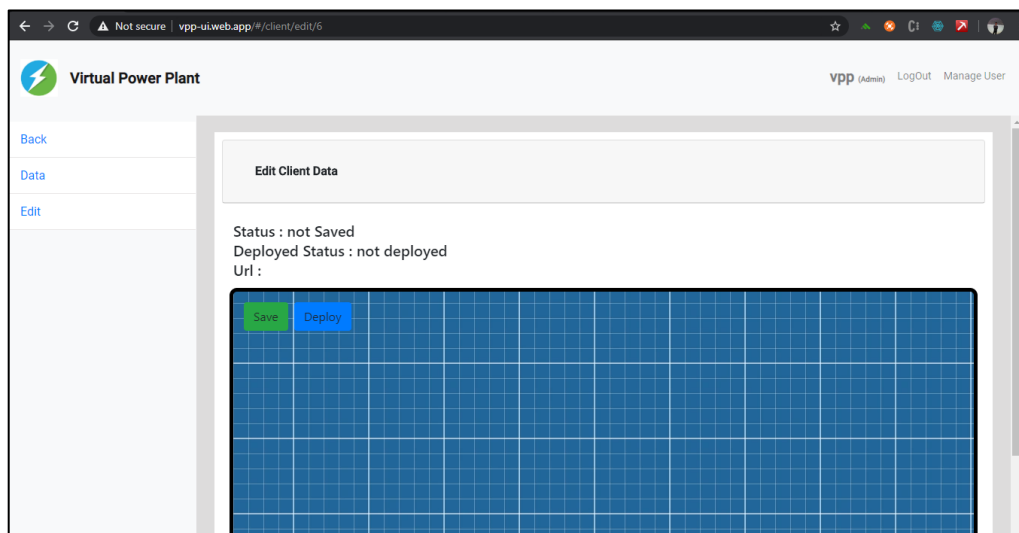
Gambar 3.5. Halaman Utama pada Platform Simulasi

Klien dalam platform simulasi pembangkit listrik virtual 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 Simulasi

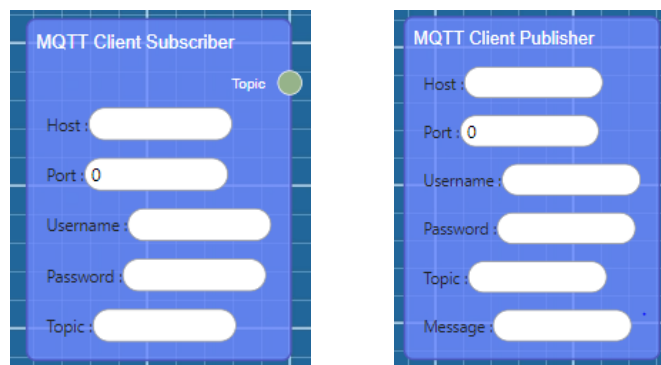
Dalam merancang *node editor*, penulis menggunakan Rete.js sebagai *framework* Javascript yang mendukung pembuatan *node-based* editor. Rete.js pada platform simulasi ini digabungkan dengan *front-end* dari website yang dibuat. Pembuatan *node editor* pada Rete.js perlu dilakukan pendefinisian *node* 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 simulasi dapat dilihat pada Gambar 3.6.



Gambar 3.6. Halaman *Node Editor* pada Platform Simulasi

Node yang dibuat untuk mendukung platform simulasi antara lain *node* MQTT (*Subscriber* dan *Publisher*), kalkulasi (*Addition* dan *Multiplication*), *database*, akumulator (*Load* dan *Generator*), Spesifikasi (*Battery* dan *Fuel Cell*) dan kontrol.

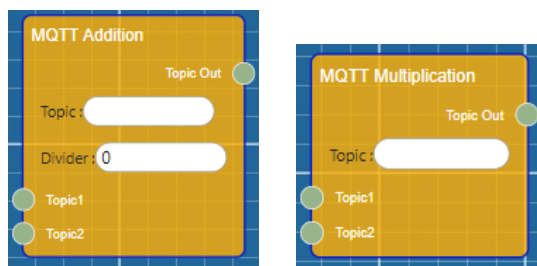
3.3.1. *Node* MQTT (*Subscriber* dan *Publisher*)



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 pesan atau nilai dari broker berdasarkan topik melalui *credentials* broker yang telah didefinisikan. Pada *node* ini terdapat komponen *control* dengan tipe data yang telah didefinisikan meliputi alamat broker 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 pesan 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 pesan kepada broker sesuai topik melalui *credentials* broker yang telah didefinisikan. Pada *node* ini terdapat komponen *control* yang hampir sama dengan *Node* MQTT Client Subscriber namun terdapat perbedaan jumlah komponen *control* yaitu adanya kolom editor *message* dengan tipe data string untuk menerima input sesuai yang diberikan oleh pengguna.

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

3.3.3. Node Database

Gambar 3.9. Node MQTT Database

Berdasarkan Gambar 3.9, terlihat bahwa *node MQTT Database* dibentuk untuk menyimpan pesan atau nilai yang didapat dari *node* lain (*Node MQTT Client Subscriber* atau *Node Kalkulasi*) melalui komponen *input*. Pada *node* ini terdapat komponen *control* dengan tipe data yang telah didefinisikan meliputi alamat server *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 input yang diberikan oleh *node* lain melalui komponen *input*. Terdapat juga komponen *input* yang akan menerima pesan dari *Node MQTT Client Subscriber* atau *Node Kalkulasi*.

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

The image shows two side-by-side screenshots of software interfaces for data accumulation nodes. The left interface is titled 'Generator Accumulator' and contains input fields for Host, Port (set to 0), Database, Username, Password, Table1, Column1, Table2, Column2, Table3, and Column3. The right interface is titled 'Load Accumulator' and contains input fields for Host, Port (set to 0), Database, Username, Password, Table, and Column. Both interfaces have a yellow background with a blue border.

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 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.

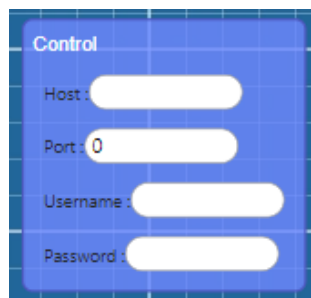
3.3.5. Node Spesifikasi (*Battery* dan *Fuel cell*)



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.

3.3.6. Node Kontrol

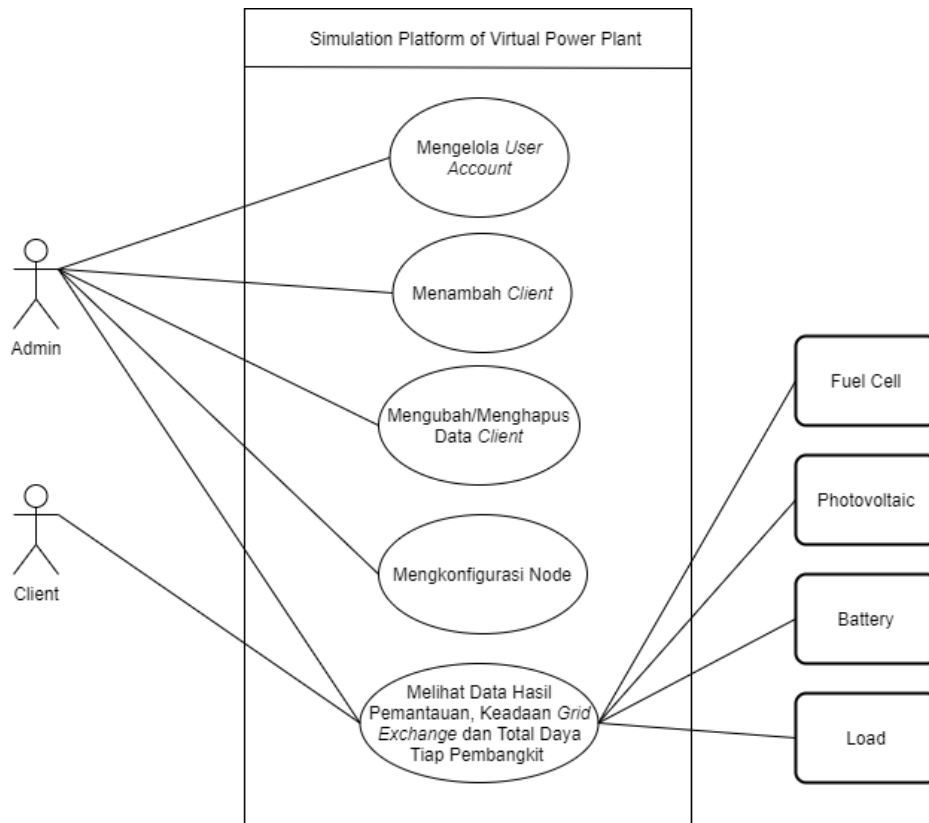


Gambar 3.12. *Node Control*

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

3.4. Use Case Diagram pada Platform Simulasi Pembangkit Listrik Virtual

Untuk mendukung perancangan platform simulasi pembangkit listrik virtual, maka penulis menyertakan *use case* diagram untuk menggambarkan interaksi komponen sistem secara keseluruhan serta dapat dilihat pada Gambar 3.13.

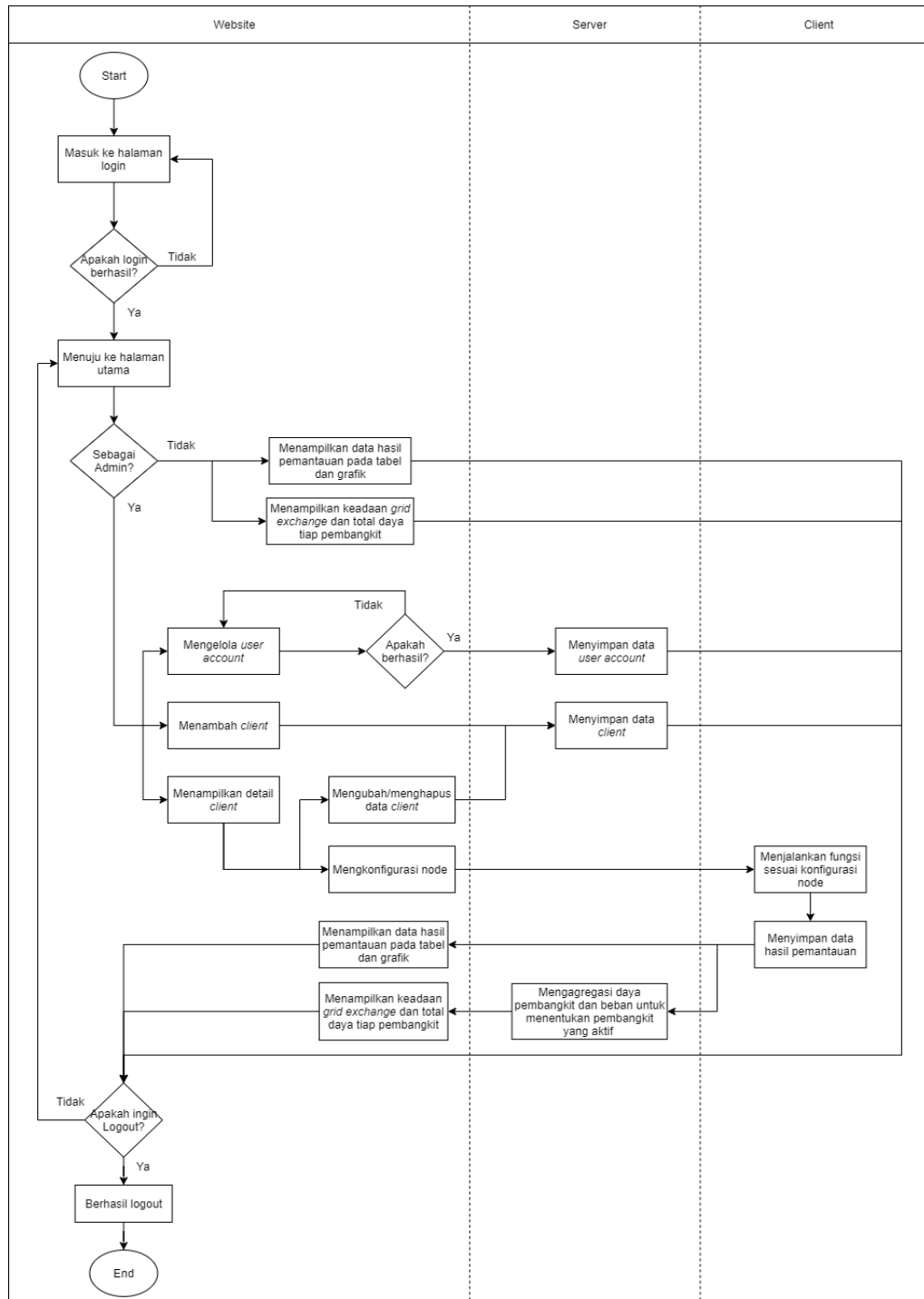


Gambar 3.13. Use Case Diagram pada Platform Simulasi

Pada Gambar 3.13, terlihat bahwa platform simulasi pembangkit listrik virtual memiliki dua *role* yaitu sebagai admin dan klien. Dalam platform simulasi ini, admin bertindak sebagai *superuser* yang dapat melakukan apapun terhadap platform simulasi 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 Simulasi Pembangkit Listrik Virtual

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



Gambar 3.14. Diagram alir pada Platform Simulasi

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 simulasi. 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 simulasi, 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 simulasi 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 simulasi maka klien dapat *logout* dari website.

BAB 4

PENUTUP

Bab ini akan membahas mengenai kesimpulan sesuai dengan apa yang dikerjakan dalam seminar berdasarkan perancangan sistem yang telah dilakukan.

4.1. Kesimpulan

Berdasarkan hasil perancangan sistem yang telah dilakukan maka didapatkan beberapa kesimpulan yaitu:

1. Perancangan *node editor* dapat membantu pengguna dalam melakukan seluruh konfigurasi yang mendukung platform simulasi pembangkit listrik virtual.
2. Perancangan sistem pemantauan dapat 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.

DAFTAR PUSTAKA

- [1] L. Yavuz, A. Önen, S. Mueeen and I. Kamwa, “Transformation of microgrid to virtual power plant – a comprehensive review”, IET Generation, Transmission & Distribution, vol. 13, no. 11, pp. 1994-2005, 2019. Available: 10.1049/iet-gtd.2018.5649.
- [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", Proccedings 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]