Journal of Applied Engineering and Social Science (JAESS)
Vol. 3, Issue 2, pp. 78–83 (2025)
doi: http://doi.org/10.25124/jaess.v3i2.6467

RESEARCH ARTICLE

Pengembangan Aplikasi Web Pengisian Daya Kendaraan Listrik Berbasis *Open Charge Point Protocol* (OCPP)

Hermanus Hasta Wicaksana, Amir Hasanudin Fauzi, S.T., M.T.* and Rizza Indah Mega Mandasari

Fakultas Ilmu Terapan, Universitas Telkom, Bandung, 40257, Jawa Barat, Indonesia
*Corresponding author: amirhf@telkomuniversity.ac.id

Abstrak

Laporan ini menyajikan pengembangan aplikasi web untuk pengisian daya kendaraan listrik berbasis *Open Charge Point Protocol* (OCPP). Modul ini meggunakan protokol OCPP 1.6. Aplikasi ini bertujuan untuk memudahkan dalam mengontrol, memonitoring, dan menganalisa data pengisian daya kendaraan listrik di Universitas Telkom. Pengembangan server lokal diusulkan untuk mengurangi ketergantungan terhadap server eksternal dan meningkatkan keamanan dan stabilitas. Aplikasi ini dirancang untuk memberikan kontrol penuh, pemantauan, dan analisis data pengisian daya kendaraan listrik, dan untuk meminimalkan risiko keamanan yang terkait dengan penggunaan server eksternal. Hasil penelitian ini menunjukkan bahwa aplikasi yang dikembangkan dapat secara efektif memfasilitasi pengelolaan data pengisian daya kendaraan listrik dan meningkatkan keamanan dan stabilitas sistem.

Key words: *back-end, charging point management, electric vehicle, OCPP, SPKLU, server.*

Pendahuluan

Penggunaan bahan bakar fosil telah menyebabkan ketergantungan energi dan krisis energi global. Di Indonesia, pemerintah mendorong pembangunan Stasiun Pengisian Kendaraan Listrik Umum (SPKLU) serta menetapkan Peraturan Presiden Nomor 5 Tahun 2019 tentang ”Percepatan Program Kendaraan Bermotor Listrik Berbasis Baterai (KBLBB)” untuk mendukung mobilitas ramah lingkungan dan mengurangi emisi gas rumah kaca [1]. Pada akhir tahun 2023, jumlah kendaraan listrik di Indonesia mencapai lebih dari 108.000 unit, dan PLN telah membangun 1.124 unit SPKLU di seluruh Indonesia [2]. Pertumbuhan ini diperkirakan akan terus meningkat di tahun-tahun berikutnya. Institusi, instansi, dan universitas turut berupaya membangun SPKLU, termasuk Telkom University. Namun, SPKLU di Telkom University masih bergantung pada server eksternal untuk kontrol dan monitoring, yang menimbulkan beberapa masalah terkait kontrol yang terbatas dan kurang optimal. Penelitian ini bertujuan untuk merancang dan mengembangkan *server back-end* berbasis *Open Charge Point Protocol* (OCPP) sebagai solusi untuk mengatasi masalah ini. OCPP adalah protokol terbuka standar untuk komunikasi antara *Charge Point* dan sistem pusat yang dapat mengakomodasi berbagai teknik pengisian daya [3]. Implementasi server lokal berbasis OCPP diharapkan dapat memberikan kontrol penuh atas stasiun pengisian daya, termasuk monitoring, manajemen pengguna, dan analisis data penggunaan. Keamanan dan privasi data pengguna juga akan ditingkatkan melalui langkah-langkah keamanan standar industri. Penelitian ini juga dapat menjadi panduan bagi institusi lain dalam mengembangkan server kendaraan listrik yang efisien dan berkelanjutan.

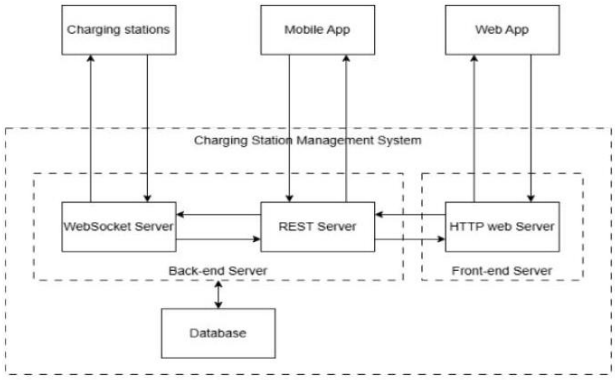
Tinjauan Pustaka

Electric Vehicle (EV)

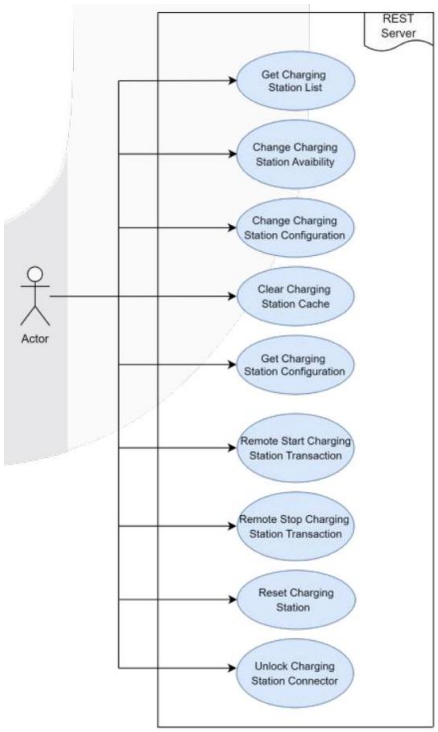
Electric Vehicle (EV) adalah kendaraan yang menggunakan motor listrik sebagai sumber tenaga utama untuk beroperasi, menggantikan mesin pembakaran dalam kendaraan konvensional yang menggunakan bahan bakar fosil seperti bensin atau diesel. Terdapat empat jenis EV, yaitu Plug-in Hybrid *Electric Vehicle* (PHEV), *Hybrid Electric Vehicle* (HEV), *Fuel Cell Electric Vehicle* (FCEV), dan KBLBB atau *Battery Electric Vehicle* (BEV) [8]. Berikut penjelasan masing-masing jenis *Electric Vehicle*.

Hypertext Transfer Protocol (HTPP)

Hypertext Transfer Protocol (HTTP) adalah protokol antara *client* dan server bisa saling berkomunikasi dengan melakukan sistem *request-response*. Biasanya HTTP digunakan untuk mengirimkan dokumen yang berbentuk *hypermedia* seperti video, gambar, audio, HTML. Protokol ini dirancang untuk melakukan komunikasi antara web browser dengan web server, selain itu juga dapat digunakan untuk tujuan lain



Gambar 1. Gambaran Umum Sistem Manajemen SPKLU



Gambar 2. *Use Case Rest Api Server*

[9]. HTTP mengikuti model *clien-server* klasik, dengan klien membuka koneksi untuk membuat permintaan, kemudian menunggu sampai menerima respons. HTTP adalah protokol tanpa status, yang berarti bahwa server tidak menyimpan data (status) apapun di antara dua permintaan [9].

Websocket

Socket memungkinkan komunikasi antar komputer, berguna dalam pemrograman berbasis *client-server*. API *WebSocket* memungkinkan komunikasi dua arah antara klien dan server melalui web. Antarmuka *WebSocket* menentukan metode dan interaksi klien dengan jaringan, dimulai dengan pemanggilan konstruktor *WebSocket* yang mengembalikan instance objek *WebSocket*, mengatur kapan koneksi dibuka, pesan tiba, koneksi ditutup, dan pemberitahuan kesalahan [10].

Table 1. Pengujian *Authorize*

Deskripsi	Pengujian fungsionalitas *Authorize* OCPP

Parameter yang dimasukkan	{
	idTag": "{{IdTag}}"
	}

Kriteria Keberhasilan	Server merespon denganmessageTypeId 2 serta payload ïdTagInfo" yang merupakan sebuah objek dengan properti: a. status: string enum (Accepted, Blocked, Expired, Invalid, ConcurrentTx). b. expiryDate (opsional): date-time c. parentIdTag (opsional): string

Hasil Pengujian	[
	"idTagInfo": {
	"Status": "Expired"
	}
	]

Open Charge Point Protocol (OCPP)

Open Charge Point Protocol (OCPP) adalah protokol terbuka standar untuk komunikasi antara *charge point* dan sistem pusat, dirancang untuk mengakomodasi berbagai teknik pengisian daya. Beberapa fitur inti OCPP 1.6 mencakup otentikasi pengguna (*Authorize*), notifikasi boot (*BootNotification*), perubahan ketersediaan (*ChangeAvailability*), perubahan konfigurasi (*ChangeConfiguration*), penghapusan cache (*ClearCache*), transfer data tambahan (*DataTransfer*), pengambilan konfigurasi (*GetConfiguration*), notifikasi status (*Heartbeat*), pengiriman nilai meter (*MeterValues*), memulai dan menghentikan pengisian daya jarak jauh (*RemoteStartTransaction* dan *RemoteStopTransaction*), inisialisasi ulang (*Reset*), memulai dan menghentikan pengisian daya (*StartTransaction* dan *StopTransaction*), notifikasi status (*StatusNotification*), dan membuka pengunci konektor (*UnlockConnector*) [3].

Metodologi Penelitian

Tahap persiapan dimulai dengan memberikan pemahaman kepada peserta magang tentang arsitektur Stasiun Pengisian Kendaraan Listrik Umum (SPKLU) dan protokol *Open Charge Point Protocol* (OCPP), termasuk fitur-fitur intinya. Peserta akan mengikuti pelatihan dan melakukan penelitian terkait implementasi OCPP dalam sistem pengisian daya kendaraan listrik. Tahap pengembangan melibatkan pembuatan *backend* manajemen SPKLU dan implementasi fitur-fitur inti OCPP seperti *Authorize*, *BootNotification*, *ChangeAvailability*, *ChangeConfiguration*, *ClearCache*, *DataTransfer*, *GetConfiguration*, *Heartbeat*, *MeterValues*, *RemoteStartTransaction*, *RemoteStopTransaction*, *Reset*, *StartTransaction*, StatusNotification, dan *UnlockConnector*. Peserta juga akan membangun server lokal yang mengontrol dan memonitor stasiun pengisian daya sesuai standar OCPP. Tahap pengujian dan *debugging* dilakukan untuk memastikan sistem berfungsi optimal. Setiap fitur yang diimplementasikan akan diuji, dan *debugging* dilakukan untuk memperbaiki kesalahan. Peserta magang akan melaporkan perkembangan secara berkala kepada pembimbing lapangan, yang akan memberikan masukan untuk perbaikan. Proyek dianggap selesai setelah semua fitur diuji dan sistem berjalan optimal, memberikan kontrol penuh atas stasiun pengisian daya, termasuk monitoring, manajemen pengguna, dan analisis data penggunaan.

Table 2. Pengujian *Bootnotification*

Deskripsi	Pengujian fungsionalitas ***BootNotification*** **OCPP**

Parameter yang dimasukkan	{
	”chargePointVendor”: ”VendorA ”,
	”chargePointModel”:
	”SingleSocketCharger”
	}

Kriteria Keberhasilan	Server merespon dengan messageTypeId 3 serta payload yang memiliki properti: a. status: string enum (Accepted, Rejected, Pending) b. currentTime: date-time c. interval: integer

Hasil Pengujian	{
	”status”: Äccepted
	”messageId”: ”2024-06-01T06:42:51.810Z
	”data”: 60
	}

Table 3. Pengujian *Datatransfer*

Deskripsi	Pengujian fungsionalitas ***DataTransfer*** **OCPP**

Parameter yang dimasukkan	{
	”vendorId”: ”VendorA ”,
	”messageId”: ”customMsg”,
	”data”: Some data”
	}

Kriteria Keberhasilan	Server merespon dengan messageTypeId 3 serta payload yang memiliki properti: status: string enum (Accepted, Rejected, UnknownMessageId, UnknownVendorId)

Hasil Pengujian	{
	”status”: ”Äccepted”
	}

Table 4. Pengujian *Heartbeat*

Deskripsi	Pengujian fungsionalitas ***HeartBeat*** **OCPP**
Parameter yang dimasukkan	Pengujian tidak perlu parameter tambahan sehingga objek kosong atau (’{}’).
Kriteria Keberhasilan	Server merespon dengan messageTypeId 3 serta payload yang memiliki properti: status: string enum (Accepted, Rejected, UnknownMessageId, UnknownVendorId)

Hasil Pengujian	{
	”currentTime”: ”2024-07-17T21:14:38.437Z”
	}

Table 5. Pengujian *Metervalues*

Deskripsi	Pengujian fungsionalitas ***MeterValues*** **OCPP**

Parameter yang dimasukkan	{
	”connectorId”: 1,
	”meterValue”: [
	{
	”timestamp”: ”2024-03- 20T03:45:46.918Z”,
	sampledValue”: [
	{
	”value”: ”ABCDEFGHIJKLMNOPQR STUVWX”,
	”context”: ”Transaction.Begin”,
	”format”: ”Raw”,
	”measurand”: ”Power.Active.Export”,
	”phase”: ”L2”,
	”location”: ”Body”,
	unit”: ”Percent”
	}
	]
	}
	]
	}

Kriteria Keberhasilan	Server merespon dengan messageTypeId 3 serta payload yang kosong.
Hasil Pengujian	Pengujian berhasil dan response objek kosong ”{}”

Gambaran Sistem Saat Ini

Sistem manajemen stasiun pengisian kendaraan listrik umum memiliki tiga jenis klien. Yaitu, SPKLU (*charging station*), aplikasi mobile, dan aplikasi website (Gambar 1). Aplikasi mobile diperuntukkan untuk pengendara kendaraan listrik sehingga pengendara bisa mengakses 12 fasilitas SPKLU. Aplikasi web diperuntukkan untuk operator SPKLU sehingga dapat mengelola dan memantau ataupun mengubah konfigurasi SPKLU.

Pengembangan Sistem

Dalam rancangan aplikasi *back-end* sistem manajemen SPKLU, terdapat diagram *use case* yang menggambarkan interaksi antara aplikasi web dengan *server back-end* interaksi ini akan diimplementasikan sebagai fitur REST API sehingga sistem lain dapat mengakses *server back-end*. Diagram *use case* mendeskripsikan fungsionalitas yang ditawarkan oleh sistem digambarkan pada gambar 2.

Hasil dan Pembahasan

Telah dilakukan pengujian fungsionalitas pada aplikasi Postman dan seluruh fitur aplikasi dipastikan sudah berjalan dengan baik. Didapatkan hasil sebagai berikut. Tabel 1 memperlihatkan hasil pengujian *Authorize* yang berfungsi untuk memverifikasi identitas pengguna sebelum melakukan transaksi. Berdasarkan Tabel 2, proses *Bootnotification* diuji untuk memastikan koneksi awal antara perangkat dan server berjalan dengan benar. Tabel 3 menampilkan hasil pengujian *Datatransfer* yang digunakan untuk menguji kemampuan sistem dalam mengirim dan menerima data tambahan. Sebagaimana terlihat pada Tabel 4, pengujian *Heartbeat* dilakukan untuk memastikan perangkat dapat mengirim sinyal status secara berkala. Tabel 5 menunjukkan hasil pengujian *Metervalues* yang merekam nilai pengukuran daya secara *real-time*. Berdasarkan Tabel 6, hasil pengujian *Starttransaction* menunjukkan respons sistem saat transaksi pengisian dimulai.

Tabel 7 menampilkan pengujian *Statusnotification* yang memverifikasi pembaruan status konektor pada sistem. Sebagaimana disajikan dalam Tabel 8, pengujian Stoptransaction dilakukan untuk memastikan proses penghentian transaksi dapat berjalan dengan baik. Tabel 9 memperlihatkan hasil pengujian REST API Get *Charging Station Clients* yang digunakan untuk menampilkan daftar klien stasiun pengisian. Berdasarkan hasil pada Tabel 10, pengujian *REST API Change*

Table 6. Pengujian *Starttransaction*

Deskripsi	Pengujian fungsionalitas *StartTransaction* OCPP

Parameter yang dimasukkan	{
	”connectorId”: 1,
	”idTag”: ”{{IdTag}}”,
	”meterStart”: 0,
	”timestamp”:”{{$isoTimestamp}}”
	}

Kriteria Keberhasilan	Server merespon dengan messageTypeId 3 serta payload yang memiliki properti: 1. transactionId: integer 2. idTagInfo: object a. status: string enum (Accepted, Blocked, Expired, Invalid, ConcurrentTx). b.expiryDate (opsional): date-time c.parentIdTag (opsional): string

Hasil Pengujian	{
	”transactionId”: 3,
	”idTagInfo”: {
	”status”: Accepted”
	}
	}

Table 7. Pengujian *Statusnotification*

Deskripsi	Pengujian fungsionalitas StatusNotification OCPP

Parameter yang dimasukkan	{
	”connectorId”: 1,
	”errorCode”: ”NoError”
	”status”: Ävailable”
	}

Kriteria Keberhasilan	Server merespon dengan messageTypeId 3 serta payload yang kosong.
Hasil Pengujian	Pengujian berhasil dan response objek kosong ”{}”

Table 8. Pengujian *Stoptransaction*

Deskripsi	Pengujian fungsionalitas *StopTransaction* OCPP

Parameter yang dimasukkan	{
	”idTag”: ”{{IdTag}}”,
	”timestamp”: ”{{$isoTimestamp}}”,
	”meterStop”: 25,
	”transactionId”: {{transactionId}}
	}

Kriteria Keberhasilan	Server merespon dengan messageTypeId 3 serta payload yang memiliki properti: 1. transactionId: integer 2. idTagInfo: object a. status: string enum (Accepted, Blocked, Expired, Invalid, ConcurrentTx). b. expiryDate (opsional): date-time c. parentIdTag (opsional): string.

Hasil Pengujian	{
	”idTagInfo”: {
	”status”: ”Äccepted”
	}
	}

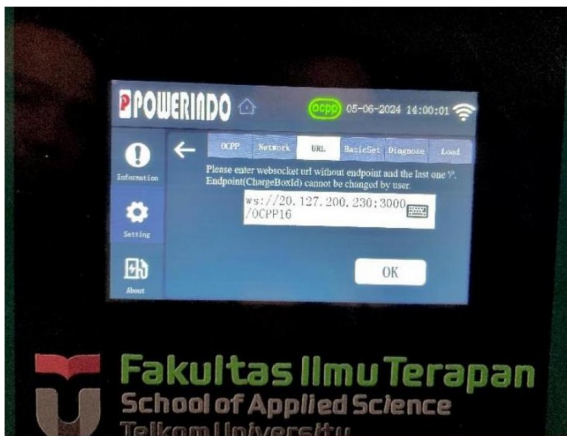
Table 9. Pengujian Rest Api Get Charging Station Clients

Deskripsi	Pengujian fungsionalitas REST API get charging station clients
Parameter yang dimasukkan	URL http: //localhost:3000/api/v1/centralsystem/clients dengan method GET.
Kriteria Keberhasilan	Server merespon dengan status ”200 Ok” dan membawa data json berupa daftar ID charging station yang sedang terhubung dengan server.
Hasil Pengujian	[”123”]

Availability memastikan perubahan status ketersediaan konektor dapat dilakukan melalui API.

Tabel 11 menyajikan pengujian *REST API Change Configuration* yang menguji kemampuan sistem dalam memperbarui parameter konfigurasi. Sebagaimana terlihat pada Tabel 12, hasil pengujian *REST API Clear Cache* menunjukkan fungsi penghapusan *cache* berjalan sesuai harapan. Tabel 13 menampilkan hasil pengujian *REST API Get Configuration* yang memverifikasi kemampuan sistem dalam mengambil konfigurasi terkini. Berdasarkan Tabel 14, pengujian *REST API Remote Start Transaction* dilakukan untuk memastikan transaksi dapat dimulai secara jarak jauh. Tabel 15 memperlihatkan hasil pengujian *REST API Remote Stop Transaction* yang berfungsi untuk menghentikan transaksi dari jarak jauh. Sebagaimana ditunjukkan pada Tabel 16, pengujian *REST API Reset* dilakukan untuk menguji proses penyegaran sistem setelah operasi tertentu. Terakhir, Tabel 17 menyajikan hasil pengujian *REST API Unlock Connector* yang memastikan konektor dapat dibuka secara jarak jauh melalui perintah API.

Saat dilakukan pengujian langsung di SPKLU terlihat gambar 3 merupakan menu pengaturan SPKLU yang berada di Telkom University Landmark Tower. Ketika melakukan testing dengan menyambungkan program yang sudah di *deploy* ke *Azure Virtual Machine*. Bila program sudah terhubung maka ikon OCPP di layar monitor akan berwarna hijau



Gambar 3. Menu SPKLU

seperti gambar di atas. Tampilan log server pada gambar 4 ketika program sudah terhubung maka akan terlihat ”ID0122120103” yang merupakan ID dari SPKLU di Gedung TULT. Terdapat juga *heartbeat*

Table 10. Pengujian *Rest Api Change Avaibility*

Deskripsi	Pengujian fungsionalitas REST API change availability

Parameter yang dimasukkan	URL http://localhost:3000/api/v1/centralsystem/-clients/123/change availability menggunakan method POST dengan body:
	{
	”connectorId”: 1,
	”type”: Ïnoperative”
	}

Kriteria Keberhasilan	Server merespon dengan status ”200 Ok” dan membawa data json dengan properti status keberhasilan operasi (Accepted, Rejected, Scheduled).

Hasil Pengujian	{
	”status”: ”Accepted”
	}

Table 11. Pengujian *Rest Api Change Configuration*

Deskripsi	Pengujian fungsionalitas REST API change configuration

Parameter yang dimasukkan	URL http://localhost:3000/api/v1/centralsystem-/clients/123/change configuration menggunakan method POST dengan body:
	{
	”key”: ”HertbeatInterval”,
	”value”: ”300”,
	}

Kriteria Keberhasilan	Server merespon dengan status ”200 Ok” dan membawa data json dengan properti status keberhasilan operasi (Accepted, Rejected, Scheduled).

Hasil Pengujian	{
	’Status”: ”Accepted”
	}

Table 12. Pengujian *Rest Api Clear Cache*

Deskripsi	Pengujian fungsionalitas REST API clear cache
Parameter yang dimasukkan	URL http://localhost:3000/api/v1/-centralsystem/clients/123/clear cache menggunakan method POST.
Kriteria Keberhasilan	Server merespon dengan status ”200 Ok” dan membawa data json dengan properti status keberhasilan operasi (Accepted, Rejected).

Hasil Pengujian	{
	’Status”: ”Accepted”
	}

status notification dan juga *Bootnotification* sudah muncul di log server dan sudah tersimpan ke *database*.

Table 13. Pengujian *Rest Api Get Configuration*

Deskripsi	Pengujian fungsionalitas REST API get configuration

Parameter yang dimasukkan	URL http://localhost:3000/api/v1/centralsystem/-clients/123/get configuration menggunakan method POST dengan body:
	{
	”key”: [”HeartbeatInterval”: ”Key2”]
	}

Kriteria Keberhasilan	Server merespon dengan status ”200 Ok” dan membawa data json dengan properti: 1. key: string 2. readonly: boolean 3. value: string

Hasil Pengujian	{
	”configurationKey”: [
	{
	”key”: ”someConfig”,
	”readonly”: true,
	”value”: ”someValue”
	}
	]
	}

Table 14. Pengujian *Rest Api Remote Start Transaction*

Deskripsi	Pengujian fungsionalitas REST API remote start transaction

Parameter yang dimasukkan	URL http://localhost:3000/api/v1/centralsystem/-clients/123/start transaction menggunakan method POST dengan body:
	{
	”idTag”: ”12345”
	}

Kriteria Keberhasilan	Server merespon dengan status ”200 Ok” dan membawa data json dengan properti status keberhasilan operasi (Accepted, Rejected).

Hasil Pengujian	{
	”Status”: ”Accepted”
	}

Table 15. Pengujian *Rest Api Remote Stop Transaction*

Deskripsi	Pengujian fungsionalitas REST API remote stop transcation

Parameter yang dimasukkan	URL http://localhost:3000/api/v1/centralsystem/-clients/123/stop transaction menggunakan method POST dengan body:
	{
	”transactionId”: 1
	}

Kriteria Keberhasilan	Server merespon dengan status ”200 Ok” dan membawa data json dengan properti status keberhasilan operasi (Accepted, Rejected).

Hasil Pengujian	{
	”Status”: ”Accepted”
	}

Table 16. Pengujian *Rest Api Reset*

Deskripsi	Pengujian fungsionalitas REST API *reset*

Parameter yang dimasukkan	URL http://localhost:3000/api/v1/centralsystem/-clients/123/reset
	menggunakan method POST dengan body:
	{
	”type”: ”Soft”
	}

Kriteria Keberhasilan	Server merespon dengan status ”200 Ok” dan membawa data json dengan properti status keberhasilan operasi (Accepted, Rejected).

Hasil Pengujian	{
	”Status”: ”Accepted”
	}

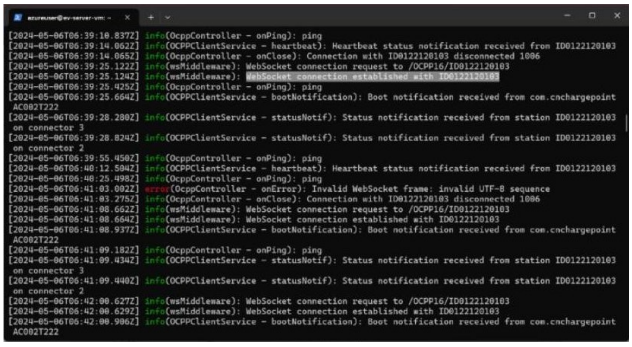
Table 17. Pengujian *Rest Api Unlock Connector*

Deskripsi	Pengujian fungsionalitas REST API unlock connector

Parameter yang dimasukkan	URL http://localhost:3000/api/v1/centralsystem/-clients/123/unlock connector
	menggunakan method POST dengan body:
	{
	”connectorId”: 1
	}

Kriteria Keberhasilan	Server merespon dengan status ”200 Ok” dan membawa data json dengan properti status keberhasilan operasi (Unlocked, UnlockFailed, NotSupported)

Hasil Pengujian	{
	”Status”: ”Unlocked”
	}



Gambar 4. Log Server

Kesimpulan

Berdasarkan hasil penelitian dan pengembangan yang telah dilakukan, beberapa kesimpulan dapat diambil. Modul OCPP yang menggunakan Node.js dan *framework* Express.js berhasil dibuat sesuai dengan rencana awal. Modul ini berhasil diuji di Stasiun Pengisian Kendaraan Listrik Umum (SPKLU) di Telkom University Landmark Tower, dan dapat berjalan dengan baik. Modul dapat terhubung ke SPKLU dan data dapat tersimpan ke dalam *database* dengan baik. Pengujian menggunakan Postman untuk simulasi berhasil, sehingga modul dapat terhubung dengan baik saat diuji langsung di SPKLU. Selama kegiatan magang di Torsi EV, penulis mendapatkan pengalaman baru yang berharga, termasuk wawasan mengenai cara kerja SPKLU dan protokol OCPP. Penulis juga berhasil mempelajari dan mengembangkan modul berbasis OCPP, mempelajari TypeScript, Node.js, dan framework Express.js. Kesimpulan ini menunjukkan bahwa tujuan penelitian telah tercapai. Penulis berhasil membantu dalam pengembangan aplikasi website berbasis OCPP yang dapat memfasilitasi pemantauan penggunaan, manajemen pengguna, dan analisis data penggunaan SPKLU. Selain itu, penulis turut membantu dalam mengembangkan struktur server lokal atau internal untuk meningkatkan kontrol atas operasi stasiun pengisian daya. Pengembangan modul ini juga membantu meningkatkan kemandirian dan keamanan sistem pengisian daya kendaraan listrik di Telkom University.

Daftar Pustaka

1. Presiden Republik Indonesia. Peraturan Presiden Republik Indonesia Nomor 55 Tahun 2019 Tentang Percepatan Program Kendaraan Bermotor Listrik Berbasis Baterai (Battery Electric Vehicle) Untuk Transportasi Jalan. Jakarta; 2019. Peraturan Presiden Republik Indonesia.
2. Trianto GA. Terus Tingkatkan Jumlah SPKLU Selama 2023, PLN Berhasil Penuhi Kebutuhan Pengguna Kendaraan Listrik di Indonesia; 2024. Accessed: 2024-02-24. Available from: `https://web.pln.co.id/media/siaran-pers/2024/01/terus-tingkatkan-jumlah-spklu-selama-2023-pln-berhasil-penuhi-kebutuhan-pengguna-kendaraan-listrik-di-indonesia`.
3. Open Charge Alliance. Open Charge Point Protocol JSON 1.6 (OCPP-J 1.6) Specification; 2015.
4. Reif K. Fundamentals of Automotive and Engine Technology. Germany: Springer; 2014.
5. Crisostomi E, Shorten R, Stüdli S, Wirth F. Electric and Plug-in Hybrid Vehicle Networks: Optimization and Control. Boca Raton, FL: CRC Press; 2018.
6. Kumara NS, Sukerayasa IW. Tinjauan Perkembangan Kendaraan Listrik Dunia Hingga Sekarang. Teknologi Elektro. 2009;8(1):74-82.
7. Vindyanandan KV. Overview of Electric and Hybrid Vehicles. A House Journal of Corporate Planning. 2018;32:7-14.
8. Autocrypt. Your Basic Guide to Electric Vehicle Technology and Trends. Seoul: Autocrypt; 2021.
9. Mozilla. The WebSocket API (WebSockets); 2024. Accessed: 2024-01-13. Available from: `https://developer.mozilla.org/enUS/docs/Web/API/WebSockets_API`.
10. Wang V, Salim F, Moskovits P. The Definitive Guide to HTML5 WebSocket. Apress; 2013.