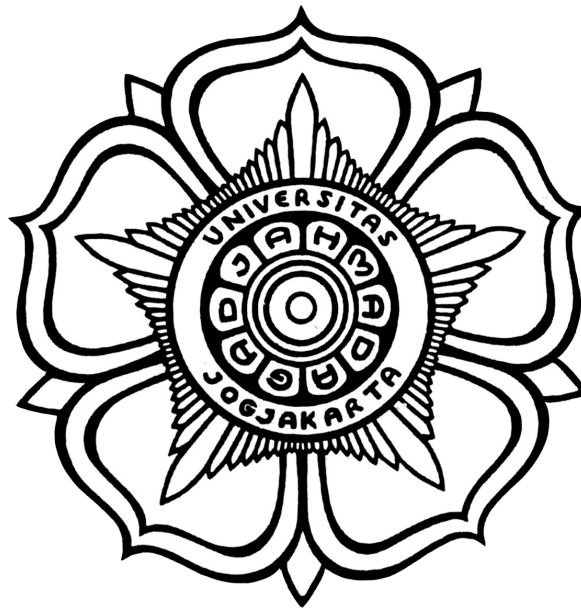


LAPORAN KERJA PRAKTIK
PENGEMBANGAN APLIKASI VISUALISASI DATA SENSOR CUACA DI
BALAI PENYELIDIKAN DAN PENGEMBANGAN TEKNOLOGI
KEBENCANAAN GEOLOGI (BPPTKG)



Disusun oleh :
ANANDA HAFIDH RIFA'I KUSNANTO
19/439605/TK/48335

PROGRAM STUDI TEKNIK ELEKTRO
DEPARTEMEN TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI
FAKULTAS TEKNIK UNIVERSITAS GADJAH MADA
YOGYAKARTA

2022

HALAMAN PENGESAHAN
PENGEMBANGAN APLIKASI VISUALISASI DATA SENSOR CUACA DI
BALAI PENYELIDIKAN DAN PENGEMBANGAN TEKNOLOGI
KEBENCANAAN GEOLOGI (BPPTKG)
LAPORAN KERJA PRAKTIK

Diajukan Sebagai Salah Satu Syarat untuk Memperoleh
Gelar Sarjana Teknik Program S-1
pada Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik
Universitas Gadjah Mada

Disusun oleh :
ANANDA HAFIDH RIFA'I KUSNANTO
19/439605/TK/48335

Telah disetujui dan disahkan
pada tanggal ... November 2022
Dosen Pembimbing Kerja Praktik

Ir. Prapto Nugroho, S.T., M.Eng., D.Eng., IPM.
NIP. 1975011520050110

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa yang telah melimpahkan berkat dan rahmatnya sehingga rangkaian Kerja Praktik dapat terlaksana dengan baik dan lancar hingga akhir. Laporan Kerja Praktik ini disusun oleh penulis berdasarkan kegiatan Kerja Praktik yang telah dilaksanakan di Balai Penyelidikan dan Pengembangan Teknologi Kebencanaan Geologi (BPPTKG) pada tanggal 15 Agustus 2022 sampai 30 September 2022

Penulis ingin mengucapkan terima kasih kepada pihak-pihak yang telah mendukung dan memberi saran serta bimbingan untuk menyelesaikan kegiatan Kerja Praktik serta penyusunan Laporan Kerja Praktik, yaitu kepada:

1. Bapak Ir. Hanung Adi Nugroho, S.T., M.E., Ph.D., IPM., selaku Ketua Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada.
2. Bapak Dr. Eng, Ir. Adha Imam Cahyadi, S.T., M.Eng., IPM., selaku Ketua Program Studi Sarjana Teknik Elektro.
3. Bapak Ir. Prapto Nugroho, S.T., M.Eng., D.Eng., IPM., selaku dosen pembimbing kerja praktik.
4. Orang tua, keluarga, dan rekan-rekan penulis yang telah memberikan dukungan selama pelaksanaan kerja praktik.
5. Ibu Sulistiyani selaku sub koordinator Gunung Merapi, yang telah memberikan kesempatan untuk melaksanakan kerja praktik di BPPTKG..
6. Bapak Ilham dan Bapak Rozin selaku pembimbing kerja praktik.

7. Segenap staff BPPTKG dan pihak-pihak lain yang telah membantu dalam pelaksanaan kerja praktik.

Penulis menyampaikan permintaan maaf apabila dalam pelaksanaan kerja praktik dan penyusunan laporan ini masih terdapat kesalahan. Semoga laporan kerja praktik ini dapat memberikan manfaat bagi para pembaca.

Yogyakarta, 25 November 2022

Penulis

DAFTAR ISI

HALAMAN PENGESAHAN	ii
KATA PENGANTAR	iii
DAFTAR ISI	v
DAFTAR GAMBAR	vii
DAFTAR TABEL	xi
BAB I: PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Tujuan	2
1.3 Batasan Masalah	2
1.4 Metode Pengumpulan Data	2
1.5 Tempat dan Waktu Pelaksanaan Kerja Praktik	3
1.6 Sistematika Penulisan	3
BAB II: PROFIL PERUSAHAAN	5
2.1. Profil Perusahaan	5
2.2. Logo Perusahaan	6
2.3. Alamat Perusahaan	6
2.4. Jam Kerja	7
BAB III: DASAR TEORI	8
3.1 Sensor Cuaca Vaisala WXT510	8
3.2 Bahasa Pemrograman JavaScript dan Node.js	9
3.3 Data parsing	9
3.3.1 ASCII	9
3.3.2 Format JSON	10
3.4 MySQL	10
3.5 User Interface	11
3.5.1 Library React.js	12
3.5.2 Framework Electron	12
BAB IV: POKOK PEMBAHASAN	13
4.1 Pra-pelaksanaan Kerja Praktik	13
4.2 Pelaksanaan Kerja Praktik	14
4.2.1 Analisis Kebutuhan	14

4.2.2 Data Parser	16
4.2.3 Database Penyimpanan	21
4.2.4 Pengembangan API	26
4.2.5 Pengembangan Tampilan User Interface	29
4.2.6 Wrapping ke Aplikasi Desktop	33
BAB V: PENUTUP	36
5.1 Kesimpulan	36
5.2 Saran	36
DAFTAR PUSTAKA	38

DAFTAR GAMBAR

Gambar 2.1	Struktur kepengurusan BPPTKG	6
Gambar 2.2	Logo instansi BPPTKG	6
Gambar 3.1	Penampang melintang dari WXT510	8
Gambar 3.2	Tabel kode ASCII untuk 128 karakter	10
Gambar 3.3	Tampilan MySQL yang diakses melalui command-line client.	11
Gambar 4.1	Skema interkoneksi antara sensor-sensor cuaca dengan komputer pada jaringan lokal milik BPPTKG	14
Gambar 4.2	Mengakses data sensor cuaca menggunakan <i>netcat</i>	15
Gambar 4.3	<i>Block diagram</i> dari rancangan sistem visualisasi data sensor cuaca	16
Gambar 4.4	<i>Flowchart</i> untuk blok TCP listener	17
Gambar 4.5	Kode dari blok TCP <i>listener</i> untuk mengambil data lewat TCP	18
Gambar 4.6	Ilustrasi proses parsing dari suatu frame data	20
Gambar 4.7	Kode dari object <i>DataParser</i> untuk melakukan parsing data	20
Gambar 4.8	Contoh hasil dari proses parsing <i>frame</i> data	21
Gambar 4.9	Kode untuk membuat koneksi ke database MySQL yang telah dibuat	22
Gambar 4.10	Contoh tabel MySQL untuk data dari sensor angin (<i>header xR1</i>)	22
Gambar 4.11	Contoh tabel MySQL untuk data pembacaan suhu, tekanan, dan kelembapan udara (<i>header xR2</i>)	23
Gambar 4.12	Contoh tabel MySQL untuk data dari pembacaan sensor curah hujan (<i>header xR3</i>)	23
Gambar 4.13	Contoh tabel MySQL untuk data <i>supervisor messages</i> (<i>header xR5</i>)	24
Gambar 4.14	Kode untuk melakukan logging data ke database MySQL	25
Gambar 4.15	Bagian dari <i>lookup-table.js</i> yang digunakan dalam pembuatan <i>query</i>	25
Gambar 4.16	Kode back-end yang melayani beberapa API <i>endpoint</i>	26

Gambar 4.17	SQL <i>query</i> pengambilan data untuk diagram windrose	27
Gambar 4.18	Contoh hasil <i>query</i> sebagai data untuk membuat diagram windrose	28
Gambar 4.19	SQL <i>query</i> pengambilan data historis dari sensor curah hujan	29
Gambar 4.20	Tampilan utama aplikasi pada tab <i>summary</i>	30
Gambar 4.21	Tampilan pada tab “angin”	31
Gambar 4.22	Tampilan historis dari data suhu, kelembapan, dan tekanan udara	31
Gambar 4.23	Tampilan pada tab “Curah Hujan”	32
Gambar 4.24	Tampilan pada tab <i>raw data</i>	32
Gambar 4.25	Kode untuk tampilan utama dari aplikasi dan tab-tab terkait	33
Gambar 4.26	Perbandingan proses instalasi dan penggunaan aplikasi berbasis web dengan aplikasi desktop	34
Gambar 4.27	Kode untuk <i>wrapping</i> aplikasi web menjadi aplikasi desktop	35

DAFTAR TABEL

Tabel 4.1	Format data yang ditransmisikan oleh Vaisala WXT510	19
-----------	---	----

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kerja praktik adalah salah satu kegiatan pembelajaran yang wajib diikuti oleh seluruh mahasiswa sarjana Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik UGM. Kegiatan Kerja Praktik ini diharapkan dapat membekali mahasiswa dengan mengasah ilmu di lingkungan industri secara langsung serta mengenali hal-hal yang dibutuhkan saat bekerja di industri terkait.

Sebagai negara yang terletak dalam Cincin Api Pasifik, Indonesia memiliki banyak gunung api aktif. Bencana erupsi gunung api tentu saja menjadi sebuah masalah yang harus dihadapi masyarakat, terutama yang tinggal dekat dengan gunung api. Untuk mengatasi kerugian akibat korban jiwa maupun material, suatu sistem untuk monitoring aktivitas gunung api diperlukan. Salah satu gunung api yang memiliki sistem monitoring yang intensif adalah Gunung Merapi, mengingat gunung ini adalah salah satu gunung api yang paling aktif di Indonesia.

Balai Penyelidikan dan Pengembangan Teknologi Kebencanaan Geologi (BPPTKG) memiliki kurang lebih 37 stasiun pengamatan Gunung Merapi yang tiap-tiapnya memiliki berbagai sensor untuk mendeteksi aktivitas kegunungapian. Dari data yang diperoleh berbagai sensor tersebut, diperlukan suatu metode visualisasi data agar informasi yang diperoleh mudah dipahami dan dianalisis oleh pihak yang terkait, sebagai sarana untuk peringatan dini kebencanaan. Dalam kerja praktik ini, penulis mendapatkan kesempatan untuk mengembangkan aplikasi untuk melakukan data parsing dan visualisasi data pada sensor cuaca Vaisala WXT510 yang terdapat pada stasiun-stasiun pengamatan Gunung Merapi yang dikelola oleh BPPTKG.

1.2 Tujuan

Tujuan dilaksanakannya kerja praktik ini adalah sebagai berikut:

1. Memenuhi persyaratan kelulusan program studi Strata-1 (S1) Teknik Elektro di Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada.
2. Mendapatkan pengalaman kerja yang menerapkan ilmu dan teknologi yang telah dipelajari di perkuliahan.
3. Mendapatkan kesempatan mempelajari dan mengembangkan pengetahuan mengenai *sensor network* dan visualisasi data, khususnya pada penerapannya dalam pencegahan dan mitigasi kebencanaan.

1.3 Batasan Masalah

Dalam pelaksanaan kerja praktik dan penulisan laporan kerja praktik ini, ditetapkan batasan masalah sebagai berikut:

1. Jumlah sensor yang digunakan untuk ditampilkan datanya dalam pembuatan aplikasi visualisasi data ini dibatasi hanya satu sensor saja.
2. Perancangan diprioritaskan pada pembuatan visualisasi data sensor secara real-time.
3. Pengembangan database dan visualisasi data historis dari sensor adalah opsional.

1.4 Metode Pengumpulan Data

Metode yang digunakan dalam penyusunan laporan kerja praktik ini adalah sebagai berikut:

- 1) Studi literatur

Penulis melakukan studi literatur dengan membaca artikel, dokumentasi, *datasheet*, dan referensi lainnya yang ada di internet untuk mempelajari

metode pengembangan data parser, *user interface* pada aplikasi visualisasi data, dan pengembangan perangkat lunaknya secara umum.

2) Metode wawancara

Penulis mengajukan pertanyaan dan berdiskusi dengan pembimbing dan supervisor di BPPTKG untuk mendapatkan informasi mengenai desain sistem operasional yang diharapkan dan memperoleh *feedback* terhadap aplikasi yang dibuat.

3) *Testing*

Penulis melakukan *testing* pada aplikasi yang dibuat pada server lokal di komputer kantor BPPTKG, mulai dari pengujian data parser dan database, hingga tampilan *user interface* pada aplikasi yang telah dibuat.

1.5 Tempat dan Waktu Pelaksanaan Kerja Praktik

Kerja Praktik dilaksanakan secara luring pada tanggal 15 Agustus 2022 sampai dengan 30 September 2022, di kantor BPPTKG yang berlokasi di Jalan Cendana No.15, Semaki, Kec. Umbulharjo, Kota Yogyakarta.

1.6 Sistematika Penulisan

Berikut merupakan sistematika penulisan laporan kerja praktik ini:

1. BAB I PENDAHULUAN

Bab I berisi latar belakang, tujuan, batasan masalah, metode pengumpulan data, tempat dan waktu pelaksanaan, serta sistematika penulisan laporan kerja praktik.

2. BAB II PROFIL PERUSAHAAN

Bab II berisi uraian singkat mengenai profil instansi tempat kerja praktik yaitu Balai Penyelidikan dan Pengembangan Teknologi Kebencanaan Geologi (BPPTKG).

3. BAB III DASAR TEORI

Bab III berisi penjelasan mengenai teori terkait dengan sensor yang digunakan, format komunikasi data, dan *toolchain* yang digunakan dalam pengembangan aplikasi visualisasi data real-time sensor cuaca Vaisala WXT510.

4. BAB IV PELAKSANAAN KERJA PRAKTIK

Bab IV berisi pembahasan mengenai kerja praktik secara keseluruhan, mulai dari kegiatan pra-kerja praktik hingga proses pengembangan aplikasi yang meliputi *data parsing*, pengaturan database, hingga pembuatan *user interface* untuk menampilkan data pembacaan sensor secara real-time.

5. BAB V PENUTUP

Bab V berisi kesimpulan kerja praktik beserta saran untuk pengembangan hasil kerja praktik.

BAB II

PROFIL PERUSAHAAN

2.1. Profil Perusahaan

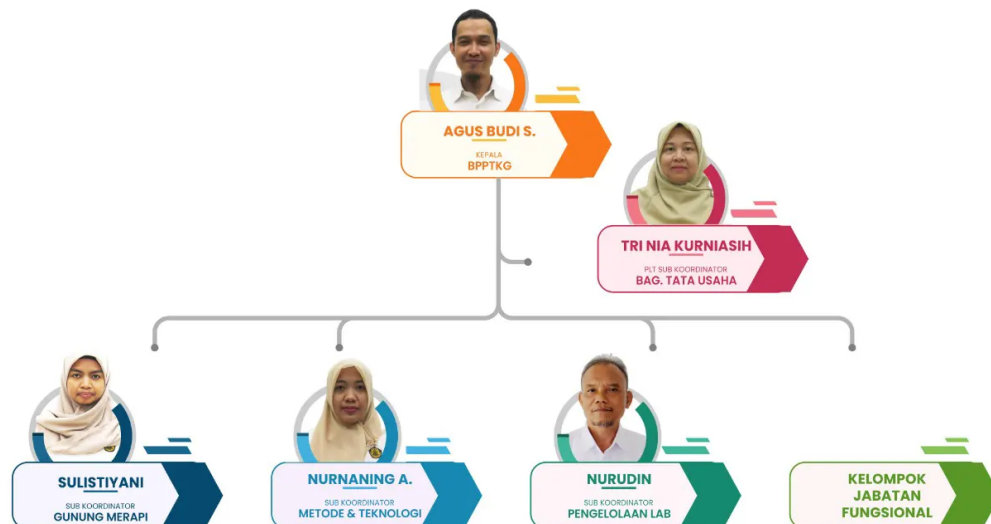
Balai Penyelidikan dan Pengembangan Teknologi Kebencanaan Geologi (BPPTKG) merupakan Unit Pelaksana Teknis di lingkungan Badan Geologi Kementerian Energi dan Sumber Daya Mineral (ESDM) yang dibentuk berdasarkan Keputusan Menteri ESDM Nomor 11 tahun 2013. BPPTKG bertanggung jawab langsung kepada Kepala Pusat Vulkanologi dan Mitigasi Bencana Geologi (PVMBG).

Tugas utama BPPTKG adalah melaksanakan mitigasi bencana Gunung Merapi, juga melakukan pengembangan metode, teknologi dan instrumentasi, dan pengelolaan laboratorium kebencanaan geologi. BPPTKG juga memiliki fungsi sebagai berikut:

1. Penyusunan rencana dan program serta pengelolaan kerja sama dan informasi.
2. Pelaksanaan mitigasi bencana Gunung Merapi.
3. Pemberian rekomendasi penetapan tingkat aktivitas dan rekomendasi teknis mitigasi Gunung Merapi.
4. Pelaksanaan penelitian, penyelidikan dan pengembangan metode, teknologi dan instrumentasi kebencanaan geologi.
5. Pengelolaan laboratorium kebencanaan geologi.
6. Pengelolaan sarana dan prasarana.
7. Pelaksanaan ketatausahaan, kepegawaian, keuangan, dan rumah tangga.

Dalam menjalankan tugas pokok dan fungsinya, terdapat tiga kelompok jabatan fungsional meliputi Subkoordinator Gunung Merapi, Subkoordinator Metode dan Teknologi, dan Subkoordinator Pengelolaan Laboratorium, yang

ketiganya berada dibawah pimpinan kepala BPPTKG. Lebih lengkapnya mengenai struktur organisasi BPPTKG terdapat pada bagan di bawah ini.



Gambar 2.1 Struktur kepengurusan BPPTKG.

2.2. Logo Perusahaan



Gambar 2.2 Logo instansi BPPTKG.

2.3. Alamat Perusahaan

Jl. Cendana No.15, Semaki, Kec. Umbulharjo, Kota Yogyakarta, Daerah Istimewa Yogyakarta 55166.

2.4. Jam Kerja

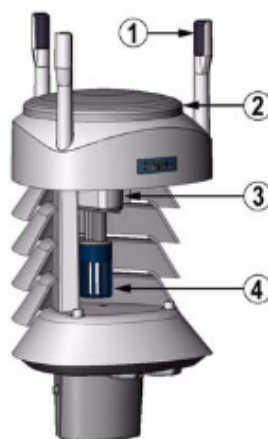
Jam kerja pada kegiatan kerja praktik ini mengikuti jam kerja kantor, yaitu pukul 08:00 hingga 16:00 pada hari Senin hingga Kamis, dan pukul 08:00 hingga 16:30 pada hari Jumat.

BAB III

DASAR TEORI

3.1 Sensor Cuaca Vaisala WXT510

Weather Transmitter WXT510 adalah sebuah sensor cuaca keluaran dari Vaisala, sebuah perusahaan dari Finlandia yang memproduksi peralatan untuk pengukuran dan instrumentasi cuaca, lingkungan, hingga pengukuran skala industrial. WXT510 dapat mengukur 6 parameter cuaca sekaligus, meliputi arah dan kecepatan angin, curah hujan, tekanan udara, suhu, dan kelembapan relatif. WXT510 dapat dicatu menggunakan tegangan 5 hingga 30 VDC sehingga dapat ditempatkan bahkan pada lokasi yang *remote*, menggunakan tenaga dari panel surya. Sensor cuaca ini mengirimkan data menggunakan komunikasi serial, dan mendukung beberapa protokol komunikasi, di antaranya RS-232, RS-485, RS-422, ataupun SDI-12. Data yang dikirimkan juga dapat diatur formatnya ke dalam format ASCII secara otomatis maupun dengan polling, format SDI-12, hingga format NMEA 0183 dengan opsi *query*.



Gambar 3.1 Penampang melintang dari WXT510; (1) transduser angin, (2) sensor curah hujan, (3) sensor tekanan, dan (4) sensor suhu dan kelembapan.

3.2 Bahasa Pemrograman JavaScript dan Node.js

JavaScript adalah bahasa pemrograman yang bersifat *interpreted* yang umum digunakan dalam pengembangan web baik bagian server atau back-end maupun bagian client atau front-end. JavaScript menjadi komponen utama dalam *world wide web* dan setiap browser memiliki JavaScript *engine* yang digunakan dalam mengakses internet dan menampilkan halaman web. Sedangkan Node.js adalah sebuah *runtime environment* dari JavaScript yang bersifat *open source* dan *cross-platform*. Node.js digunakan untuk membuat server atau bagian back-end dari suatu aplikasi web. Node.js dapat mengeksekusi kode JavaScript di luar *environment* dari web browser. Dengan Node.js, maka pengembangan aplikasi web dapat menggunakan satu bahasa pemrograman saja yaitu JavaScript untuk bagian back-end maupun bagian front-end dari aplikasi tersebut.

3.3 Data parsing

Data parsing adalah proses pengolahan data dari suatu format ke format lainnya. Data parsing dapat berupa mengubah runtun bit data menjadi bentuk data lain yang lebih bermakna seperti pada pengolahan data dari sensor ataupun GPS, compiler ataupun interpreter dari suatu bahasa pemrograman, hingga mengolah suatu runtun kalimat dan melakukan analisis menggunakan *natural language processing*. Pada laporan ini, data parsing yang dimaksud adalah proses mengolah data dari sensor cuaca yang berupa *string* ASCII menjadi format JSON yang akan digunakan pada program JavaScript dan penyimpanan ke database.

3.3.1 ASCII

American Standard Code for Information Interchange atau ASCII adalah jenis *character encoding* yang paling umum digunakan pada komunikasi dan pertukaran data secara elektronik. Pada ASCII, 128 karakter meliputi karakter *alphanumeric* yang *printable* dan karakter kontrol dinyatakan dalam 7-bit data.

USASCII code chart

b ₇ b ₆ b ₅					b ₄ b ₃ b ₂ b ₁					Column	0	1	2	3	4	5	6	7
Bits					Row						0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	NUL	DLE	SP	0	@	P	\	p	
0	0	0	0	0	0	0	1	1	1	SOH	DC1	!	1	A	Q	a	q	
0	0	0	0	0	0	1	0	0	2	STX	DC2	"	2	B	R	b	r	
0	0	0	0	0	1	1	1	3	ETX	DC3	#	3	C	S	c	s		
0	0	0	0	1	0	0	0	4	EOT	DC4	\$	4	D	T	d	t		
0	0	0	1	0	0	0	1	5	ENQ	NAK	%	5	E	U	e	u		
0	0	0	1	0	0	1	0	6	ACK	SYN	&	6	F	V	f	v		
0	0	0	1	0	1	0	0	7	BEL	ETB	'	7	G	W	g	w		
0	0	0	1	0	1	0	1	8	BS	CAN	(8	H	X	h	x		
0	0	0	1	0	1	1	0	9	HT	EM)	9	I	Y	i	y		
0	0	0	1	0	1	1	1	10	LF	SUB	*	:	J	Z	j	z		
0	0	1	0	0	0	0	0	11	VT	ESC	+	;	K	[k	{		
0	0	1	0	0	0	0	1	12	FF	FS	,	<	L	\	l			
0	0	1	0	0	0	1	0	13	CR	GS	-	=	M]	m	}		
0	0	1	0	0	1	0	1	14	SO	RS	.	>	N	^	n	~		
0	0	1	0	1	0	1	1	15	SI	US	/	?	O	_	o	DEL		

Gambar 3.2 Tabel kode ASCII untuk 128 karakter.

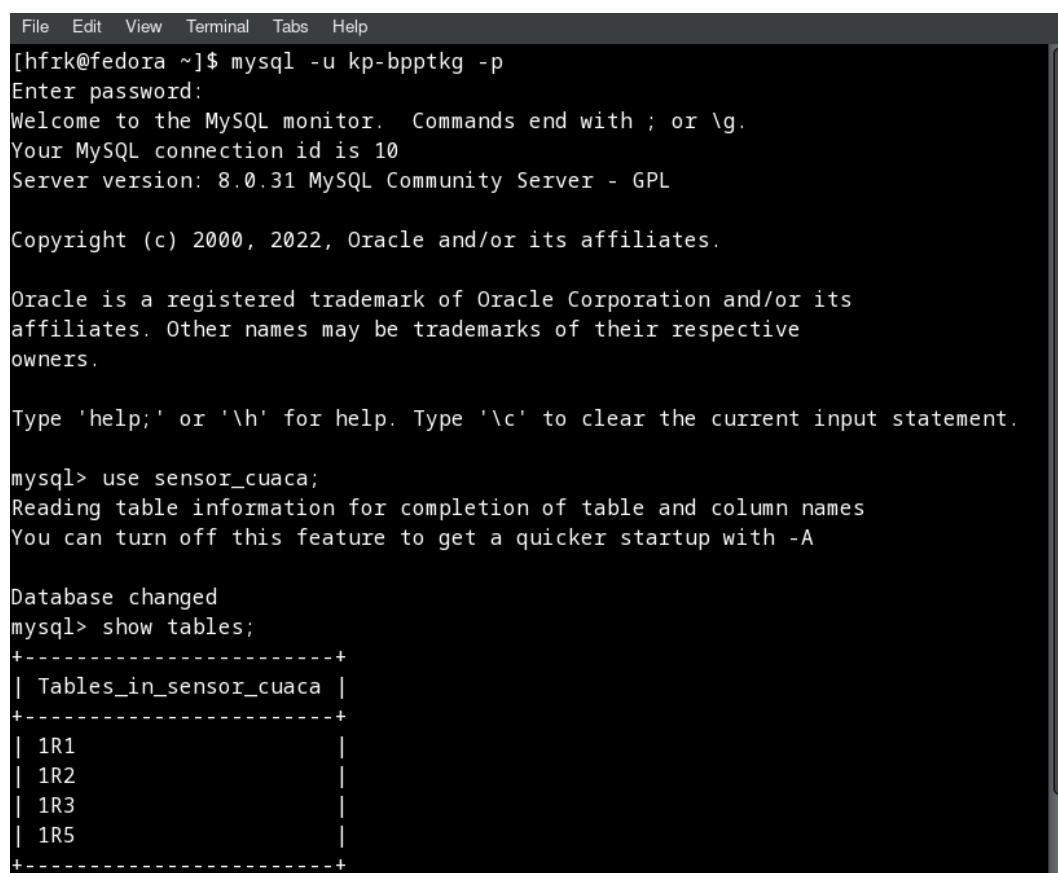
3.3.2 Format JSON

JSON (JavaScript Object Notation) adalah salah satu format pertukaran data yang berbentuk teks. Format JSON berdasarkan bahasa pemrograman JavaScript standar ECMA-262. Meskipun ada JavaScript di namanya, JSON bersifat independen dan tidak terikat bahasa pemrograman tertentu. JSON menggunakan sintaks yang familiar dengan keluarga bahasa pemrograman C meliputi C, C++, C#, Java, JavaScript, dan lainnya sehingga JSON ideal sebagai format pertukaran data. Pada JSON terdapat 2 struktur data, yaitu kumpulan *key-value pair* seperti *struct* pada bahasa C atau *object/dictionary* pada bahasa JavaScript/Python dan kumpulan data yang berurutan (*ordered list*) seperti *array*. Kedua struktur data tersebut bersifat universal dan didukung semua bahasa pemrograman.

3.4 MySQL

MySQL adalah sebuah SQL *database management system* yang bersifat *open source* dan cukup populer. MySQL dikembangkan dan di-maintain oleh Oracle Corporation. Sebagai *database management system*, MySQL digunakan

untuk menambahkan, menghapus, mengakses, hingga mengolah data-data yang disimpan. Sesuai dengan namanya, MySQL menerapkan Structured Query Language (SQL) yang merupakan bahasa yang digunakan untuk mengakses database. Untuk mengakses data pada database MySQL harus menggunakan perintah atau query SQL yang sesuai. Database pada MySQL bersifat relasional, yang artinya data-data pada database disimpan dalam tabel-tabel yang berkaitan satu sama lainnya.



```
File Edit View Terminal Tabs Help
[hfrk@fedora ~]$ mysql -u kp-bpptkg -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.31 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use sensor_cuaca;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_sensor_cuaca |
+-----+
| 1R1                     |
| 1R2                     |
| 1R3                     |
| 1R5                     |
+-----+
```

Gambar 3.3 Tampilan MySQL yang diakses melalui *command-line client*.

3.5 User Interface

User Interface adalah suatu titik komunikasi antara manusia sebagai *user* dengan komputer dalam sebuah interaksi manusia dengan komputer. Bentuk UI

dapat meliputi misalnya tampilan layar, *keyboard* dan *mouse*, tampilan desktop, hingga tampilan dari suatu aplikasi atau situs web. UI berevolusi dari awalnya yang berupa *command line interface* (CLI) yang berupa tampilan teks hingga menjadi *graphical user interface* (GUI) yang menampilkan gambar seperti pada aplikasi desktop dan web saat ini. Pada laporan ini, yang dimaksud dengan UI adalah tampilan pada aplikasi yang dibuat, yang merupakan bagian dari GUI. Elemen-elemen dari GUI meliputi window, menu, tab, tombol, ikon, dan lain-lain, yang berfungsi menampilkan informasi kepada pengguna secara interaktif dan nyaman.

3.5.1 Library React.js

Dalam pengembangan aplikasi web, digunakan library React.js untuk mengembangkan tampilan UI dari aplikasi yang dibuat. React.js adalah library JavaScript yang digunakan untuk membuat tampilan front-end atau UI dengan mudah sehingga pengembangan aplikasi menjadi lebih cepat. Dalam library React.js, terdapat komponen-komponen UI yang dapat digunakan untuk menyusun kerangka dari suatu tampilan *user interface*.

3.5.2 Framework Electron

Aplikasi berbasis web dapat diubah menjadi aplikasi desktop menggunakan wrapper, salah satunya adalah framework Electron. Electron adalah framework untuk pengembangan aplikasi desktop dengan menggunakan bahasa pemrograman JavaScript serta HTML dan CSS. Electron membuat suatu aplikasi desktop dengan menambahkan browser Chromium dan framework Node.js ke file *binary* atau file *executable* dari aplikasi. Dengan demikian, aplikasi web yang berbasis JavaScript dapat dibuat menjadi aplikasi desktop yang *cross-platform* sehingga tidak perlu mengembangkan aplikasi desktop tersendiri secara *native*.

BAB IV

POKOK PEMBAHASAN

4.1 Pra-pelaksanaan Kerja Praktik

Pada rangkaian kegiatan kerja praktik yang penulis lakukan, proses pertama yang dilakukan adalah pendaftaran kegiatan kerja praktik di BPPTKG lewat *form* pendaftaran kerja praktik yang disediakan serta menyertakan proposal rencana kegiatan kerja praktik. Setelah melewati proses administrasi tersebut, penulis ditempatkan pada bagian Subkoordinasi Gunung Merapi yang dipimpin oleh Ibu Sulistiyani. Selanjutnya dilakukan diskusi dengan beliau untuk menentukan kegiatan yang akan dilakukan. Adapun beberapa kegiatan yang ditawarkan di antaranya adalah:

- Modifikasi sensor infrasonik untuk dihubungkan dengan digitizer DM24 yang *industrial-grade*
- Instalasi sensor cuaca Vaisala WXT510 pada stasiun pengamatan Gunung Merapi
- Pengembangan aplikasi untuk parsing dan visualisasi data secara real-time dari sensor cuaca tersebut

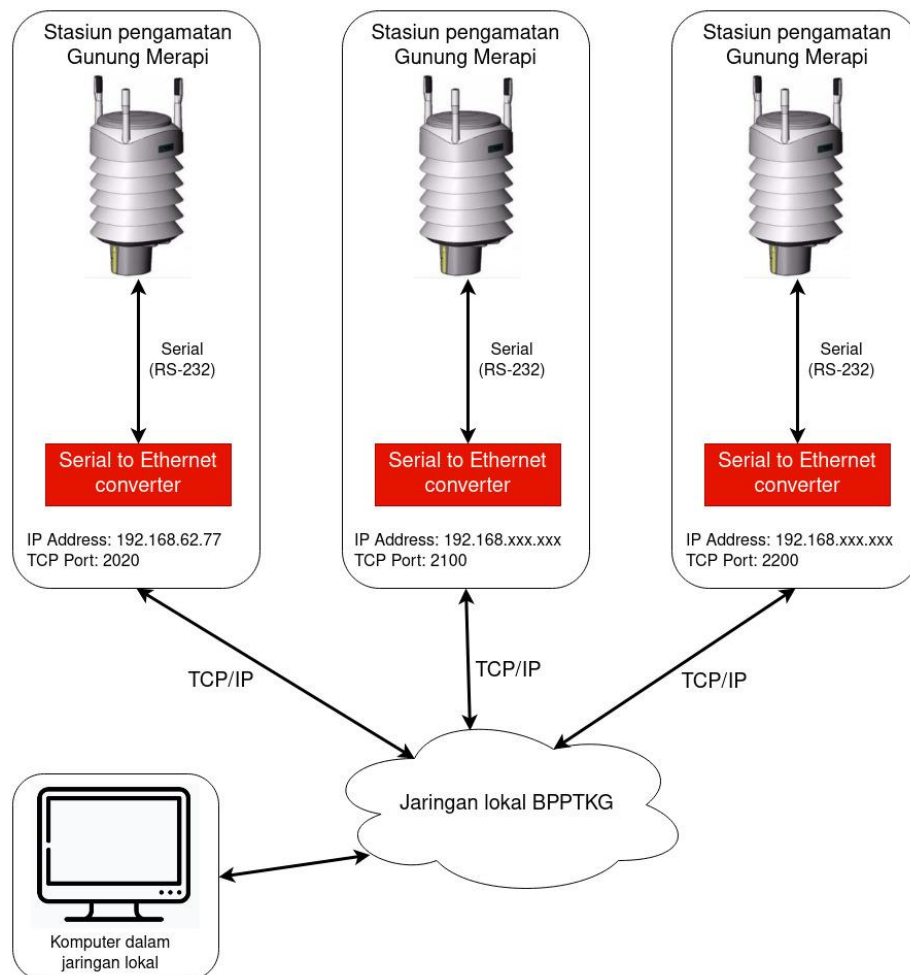
Selain itu, juga diberikan waktu untuk mengeksplorasi kegiatan-kegiatan tersebut dan berdiskusi dengan pembimbing terkait.

Dari beberapa kegiatan yang ditawarkan, penulis memilih untuk mengembangkan aplikasi untuk parsing dan visualisasi data. Pertimbangannya adalah relevansi dengan mata kuliah yang telah diambil penulis dan ketertarikan pribadi dengan topik tersebut, yaitu mengenai pengembangan sistem monitoring yang menerapkan *internet of things* (IoT), dan *web development*.

4.2 Pelaksanaan Kerja Praktik

4.2.1 Analisis Kebutuhan

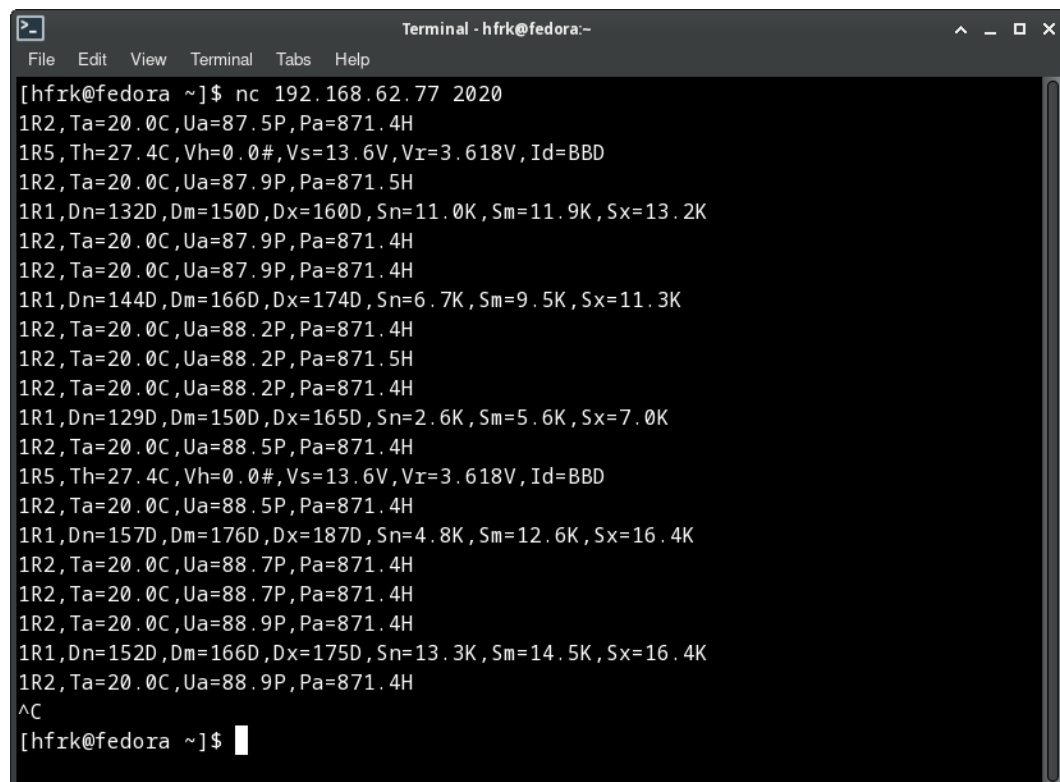
Dalam sistem pengamatan Gunung Merapi yang dikelola oleh BPPTKG, terdapat berbagai sensor yang tersebar dalam 37 stasiun pengamatan. Salah satu sensor yang digunakan adalah Vaisala WXT510 weather transmitter. Pada tiap stasiun pengamatan, sensor cuaca tersebut disambungkan melalui sebuah serial-to-ethernet *converter* agar dapat diakses melalui jaringan komputer di BPPTKG oleh beberapa pengguna secara bersamaan.



Gambar 4.1 Skema interkoneksi antara sensor-sensor cuaca dengan komputer pada jaringan lokal milik BPPTKG.

Sensor cuaca akan membaca parameter seperti arah dan kecepatan angin, suhu, kelembapan, dan tekanan udara, curah hujan dan durasi hujan, serta data-data telemetri lainnya seperti suhu komponen, tegangan catu, dan ID dari sensor. Setelah itu, tiap-tiap data akan ditransmisikan sesuai tipenya melalui komunikasi serial asinkron dengan protokol RS-232. Port serial tersebut kemudian dihubungkan ke serial-to-ethernet converter, yang memiliki alamat IP tersendiri yang dapat diakses dari jaringan komputer BPPTKG.

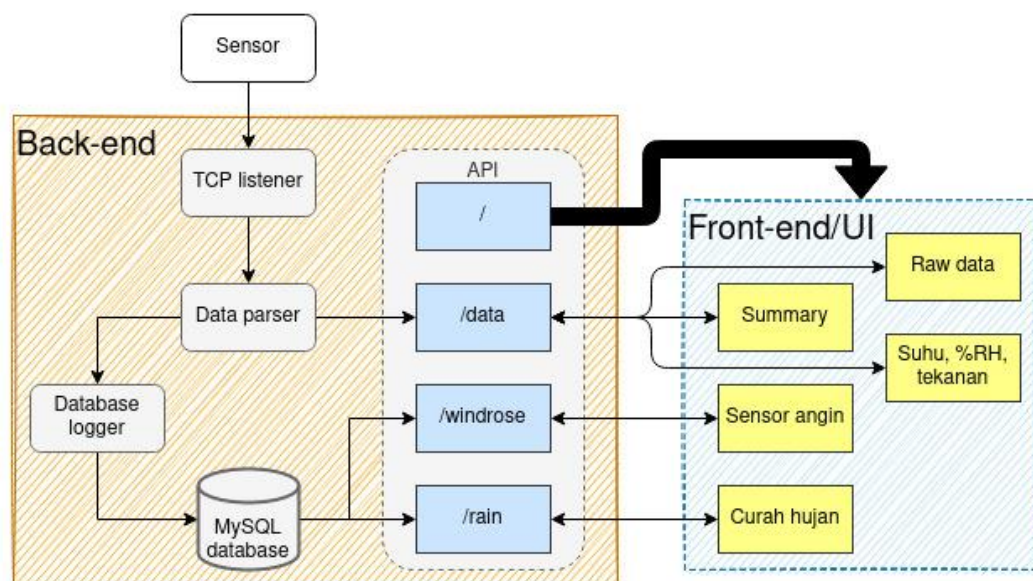
Untuk mengakses data pada sensor cuaca tersebut, dapat digunakan protokol komunikasi TCP/IP lalu mendengarkan alamat IP dan port TCP dari sensor yang diinginkan. Koneksi dapat dibuat melalui browser, *terminal emulation software* seperti HyperTerminal, ataupun *utility command* seperti netcat atau nc pada sistem operasi Linux. Apabila koneksi melalui TCP/IP telah terhubung, akan muncul *stream* data sebagaimana layaknya suatu serial monitor, seperti pada gambar 4.2.

A screenshot of a terminal window titled "Terminal - hfrk@fedora-". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal shows a netcat listener on IP 192.168.62.77 port 2020. It receives a connection and displays a stream of sensor data in a structured format. The data includes temperature (Ta), humidity (Ua), pressure (Pa), and various sensor IDs (R2, R1, R5). The stream ends with a Ctrl-C (^C) and the netcat prompt returns.

```
Terminal - hfrk@fedora-
File Edit View Terminal Tabs Help
[hfrk@fedora ~]$ nc 192.168.62.77 2020
1R2,Ta=20.0C,Ua=87.5P,Pa=871.4H
1R5,Th=27.4C,Vh=0.0#,Vs=13.6V,Vr=3.618V,Id=BBD
1R2,Ta=20.0C,Ua=87.9P,Pa=871.5H
1R1,Dn=132D,Dm=150D,Dx=160D,Sn=11.0K,Sm=11.9K,Sx=13.2K
1R2,Ta=20.0C,Ua=87.9P,Pa=871.4H
1R2,Ta=20.0C,Ua=87.9P,Pa=871.4H
1R1,Dn=144D,Dm=166D,Dx=174D,Sn=6.7K,Sm=9.5K,Sx=11.3K
1R2,Ta=20.0C,Ua=88.2P,Pa=871.4H
1R2,Ta=20.0C,Ua=88.2P,Pa=871.5H
1R2,Ta=20.0C,Ua=88.2P,Pa=871.4H
1R1,Dn=129D,Dm=150D,Dx=165D,Sn=2.6K,Sm=5.6K,Sx=7.0K
1R2,Ta=20.0C,Ua=88.5P,Pa=871.4H
1R5,Th=27.4C,Vh=0.0#,Vs=13.6V,Vr=3.618V,Id=BBD
1R2,Ta=20.0C,Ua=88.5P,Pa=871.4H
1R1,Dn=157D,Dm=176D,Dx=187D,Sn=4.8K,Sm=12.6K,Sx=16.4K
1R2,Ta=20.0C,Ua=88.7P,Pa=871.4H
1R2,Ta=20.0C,Ua=88.7P,Pa=871.4H
1R2,Ta=20.0C,Ua=88.9P,Pa=871.4H
1R1,Dn=152D,Dm=166D,Dx=175D,Sn=13.3K,Sm=14.5K,Sx=16.4K
1R2,Ta=20.0C,Ua=88.9P,Pa=871.4H
^C
[hfrk@fedora ~]$
```

Gambar 4.2 Mengakses data sensor cuaca menggunakan *netcat*.

Dari data pembacaan sensor cuaca yang telah diterima tersebut, akan dibuat sebuah aplikasi untuk menampilkan pembacaan sensor dengan tampilan yang menarik dan mudah dibaca. Gambaran sistem yang akan dibuat kurang lebihnya adalah sesuai block diagram pada gambar 4.3. Aplikasi terdiri dari bagian back-end dan front-end. Bagian back-end berisi logika aplikasi untuk mengolah data dari sensor, terdiri dari blok TCP listener, data parser, dan database logger. Selain itu, bagian back-end juga melayani permintaan data dari pengguna melalui beberapa *application programming interface* (API) *endpoint*. Sedangkan bagian front-end menampilkan visualisasi data, dengan data yang ditampilkan berasal dari API *endpoint* yang terkait. Visualisasi meliputi data sensor yang bersifat real-time yang langsung diambil dari parser, dan juga data historis yang diambil dari database.



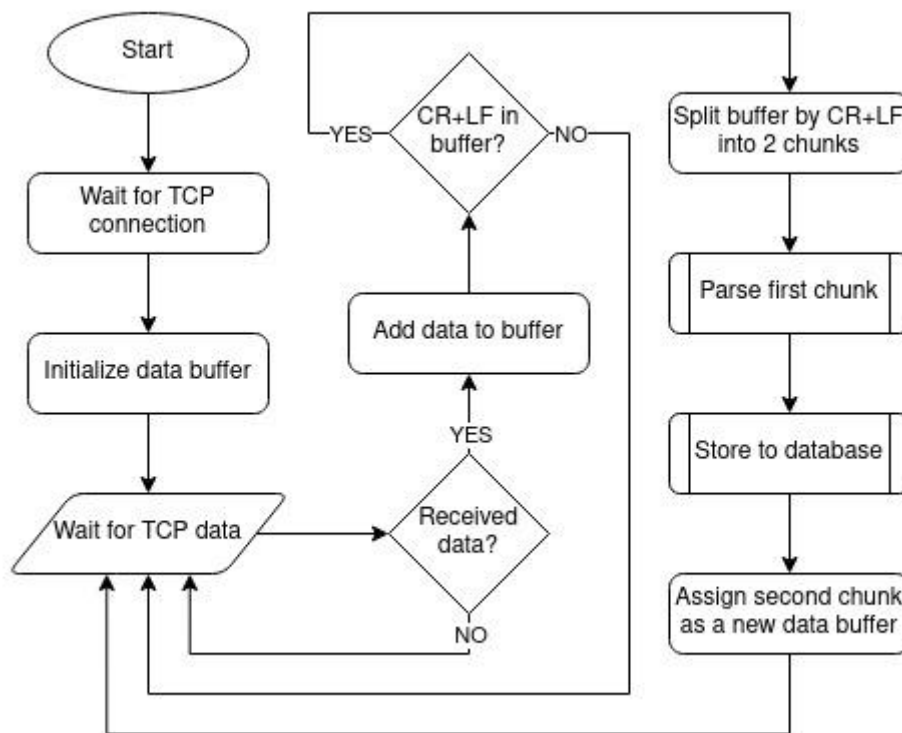
Gambar 4.3 Block diagram dari rancangan sistem visualisasi data sensor cuaca.

4.2.2 Data Parser

Dari data yang telah diperoleh melalui koneksi TCP/IP tadi, perlu dilakukan parsing agar data yang awalnya berbentuk string ASCII menjadi nilai yang dapat

disimpan ke database maupun ditampilkan pada tampilan UI nantinya. Proses data parsing pada aplikasi ini dibuat menggunakan JavaScript dan framework Node.js.

Koneksi TCP/IP dalam framework Node.js dibuat menggunakan library 'net'. Instance koneksi dibuat dengan memanggil fungsi dan mensuplai argumen berupa alamat IP dari sensor dan alamat port TCP-nya. Pada kasus ini, digunakan alamat IP yaitu 192.168.62.77 dan port TCP-nya adalah 2020. Adapun algoritma yang digunakan dalam blok TCP listener adalah sesuai pada gambar 4.4.



Gambar 4.4 Flowchart untuk blok TCP listener.

Setelah koneksi berhasil dibuat, variabel buffer akan menyimpan setiap data yang diterima untuk sementara sebelum dilakukan parsing. Variabel buffer diperlukan karena data yang diterima bukan merupakan sebuah frame utuh hasil pengamatan, melainkan terpotong-potong tergantung dengan kualitas koneksi TCP/IP dengan sensor. Data yang telah berada dalam buffer kemudian dipisah menjadi tiap-tiap frame, dengan cara mendeteksi ada tidaknya line ending berupa

carriage return (CR) dan line feed (LF). Splitting dilakukan dengan memanggil method split pada variabel buffer yang bertipe data string. Tiap-tiap frame data yang diperoleh kemudian dilakukan parsing melalui object DataParser. Sisa dari proses splitting atau bagian yang tidak terdapat line ending kemudian diproses kembali dengan dijadikan nilai baru dari variabel buffer. Implementasi dari proses pengambilan data dari port TCP terdapat pada gambar 4.5.

```
// Data Listener
require('dotenv').config();

const net = require('net');
const DBLogger = require("../dblogger.js");
const DataParser = require("../parser.js");

const host = process.env.SENSOR_HOST;
const port = parseInt(process.env.SENSOR_PORT);

const client = new net.Socket();

let sensorData = {};
let buffer = "";

client.connect(port, host, function() {
  console.log('Connected to sensor');
});

client.on('data', function(data) {
  buffer += String(data);
  if (buffer.includes("\r\n")) {
    let chunks = buffer.split("\r\n");
    buffer = chunks.pop();
    chunks.forEach((chunk) => {
      let parsedData = DataParser.parse(chunk);
      DBLogger.log(parsedData);
      sensorData = parsedData;
    });
  }
});

module.exports.getData = () => {
  return sensorData;
};
```

Gambar 4.5 Kode dari blok TCP listener untuk mengambil data lewat TCP.

Berdasarkan spesifikasi data yang ditransmisikan pada datasheet Vaisala WXT510, tiap-tiap frame data dapat dikategorikan sebagai berikut:

1. Hasil pembacaan sensor angin (anemometer), ditandai dengan header xR1
2. Hasil pembacaan sensor suhu, tekanan udara, dan kelembapan, ditandai dengan header xR2
3. Hasil pembacaan sensor curah hujan, ditandai dengan header xR3
4. Data telemetri lainnya atau disebut sebagai *supervisor messages*, ditandai dengan header xR5

Tipe frame	Contoh data
Sensor angin (xR1)	0R1,Dn=236D,Dm=283D,Dx=031D,Sn=0.0M,Sm=1.0M,Sx=2.2M<cr><lf>
Suhu, tekanan, dan kelembapan (xR2)	0R2,Ta=23.6C,Ua=14.2P,Pa=1026.6H<cr><lf>
Sensor curah hujan (xR3)	0R3,Rc=0.0M,Rd=0s,Ri=0.0M,Hc=0.0M,Hd=0s,Hi=0.0M,Rp=0.0M, Hp=0.0M<cr><lf>
<i>Supervisor messages</i> (xR5)	0R5,Th=25.9C,Vh=12.0N,Vs=15.2V,Vr=3.475V<cr><lf>

Tabel 4.1 Format data yang ditransmisikan oleh Vaisala WXT510.

Pada bahasa pemrograman JavaScript dan pengolahan database seringkali digunakan format JSON untuk pertukaran data. Oleh karena itu, data dari sensor akan diubah formatnya menjadi bentuk JSON agar mudah diproses lebih lanjut. Karena format datanya yang relatif simpel, dalam proses parsing hanya menggunakan method split pada string. Tiap frame akan dilihat *header*-nya dan tiap *field* datanya akan dipisahkan, lalu dibuat sebuah JSON object yang berisi *field-field* data tadi. Ditambahkan pula informasi *timestamp* pada JSON object yang dibuat, yang menginformasikan waktu di mana proses parsing selesai dilakukan. Informasi mengenai jenis sensor dan ID sensor disesuaikan berdasarkan *header* yang ada pada frame.

0R1,Dn=236D,Dm=283D,...,Sx=2.2M<cr><lf>

header , field = value , ... , field = value line
 name name ending

Gambar 4.6 Ilustrasi proses parsing dari suatu frame data.

```
const {FieldName, SensorName} = require("../lookup-table.js");

const DataParser = {
  parse: (data) => {
    /*
     * data format:
     * [header],[field_1]=[data_1],[field_2]=[data_2],...,[field_n]=[data_n]
     * header format:
     * xRy
     * x = sensor id
     * Ry = sensor type
     * R1: "Anemometer"
     * R2: "Barometer"
     * R3: "Precipitation"
     * R5: "Supervisor"
     */
    let arrData = data.split(",");

    let sensor = arrData.shift();
    let sensorId = sensor.slice(0, 1);
    let sensorType = sensor.slice(1);

    let parsedData = arrData.map(field => field.split('=', 2))
      .map(field => `"${field[0]}": "${field[1]}"`)
      .join();

    let result = {
      id: Number(sensorId),
      type: sensorType,
      name: SensorName[sensorType],
      data: JSON.parse(`{${parsedData}}`),
      timestamp: Date.now()
    }

    return result;
  }
}

module.exports = DataParser;
```

Gambar 4.7 Kode dari object DataParser untuk melakukan parsing data.

```

{
  "id": "0",
  "type": "R1",
  "name": "Anemometer",
  "data": {
    "Dn": "236D",
    "Dm": "283D",
    //...
    "Sx": "2.2M"
  },
  "timestamp": 1668585391052
}

```

Gambar 4.8 Contoh hasil dari proses parsing frame data.

4.2.3 Database Penyimpanan

Database yang digunakan untuk menyimpan data sensor yang telah dilakukan parsing adalah MySQL yang digunakan secara lokal. Sebelumnya dilakukan konfigurasi MySQL terlebih dahulu dengan membuat akun dan database. Setelah berhasil dibuat, database dapat diakses melalui *environment* Node.js dengan menggunakan library *mysql* dan memberikan argumen alamat dari host MySQL, username dan password, serta nama database yang digunakan.

Pada database yang telah dibentuk, yang diberi nama *sensor_cuaca*, terdiri dari tabel-tabel yang berisikan data dari sensor yang telah di-parsing. Penamaan tabel disesuaikan dengan nomor ID dari sensor dan tipe frame, sehingga setiap sensor memiliki 4 tabel yang berasosiasi dengan sensor tersebut. Misalkan terdapat sensor dengan nomor ID adalah 1, maka akan ada tabel 1R1, 1R2, 1R3, dan 1R5. Tiap tabel terdiri dari kolom *timestamp* yang menandakan waktu di mana frame data tersebut selesai diparsing, dan beberapa kolom lainnya yang merupakan field-field data pada frame yang terkait. Kolom *timestamp* diatur tipe datanya menjadi *datetime*, karena lebih mudah dikonversi dan dilakukan *query* dibandingkan tipe data *timestamp* bawaan MySQL. Kolom untuk tiap field data menyesuaikan dari bentuk data yang diterima.

```
// SQL Connection
require('dotenv').config();

let mysql = require('mysql');
let con = mysql.createConnection({
  host: process.env.SQL_HOST,
  user: process.env.SQL_USER,
  password: process.env.SQL_PASS,
  database: process.env.SQL_DB
});

con.connect(function(err) {
  if (err) {
    console.log("Failed to connect");
    throw err;
  }
  console.log("Connected to database");
});

module.exports = con;
```

Gambar 4.9 Kode untuk membuat koneksi ke database MySQL yang telah dibuat.

```
mysql> describe 1R1;
+-----+-----+
| Field      | Type      |
+-----+-----+
| timestamp  | datetime  |
| Dn         | int       |
| Dm         | int       |
| Dx         | int       |
| Sn         | float     |
| Sm         | float     |
| Sx         | float     |
+-----+-----+
```

Gambar 4.10 Contoh tabel MySQL untuk data dari sensor angin (header xR1).

Tabel yang pertama adalah untuk sensor angin/anemometer atau frame dengan header xR1. Tabel ini terdiri dari kolom *timestamp*, Dn, Dm, Dx, Sn, Sm, dan Sx. Kolom Dn, Dm, dan Dn masing-masing menandakan arah angin minimum, rerata, dan maksimum pada suatu pengukuran. Tipe datanya adalah

integer karena jangkauan nilainya hanya antara 0 hingga 359 derajat dan berupa bilangan bulat. Sedangkan kolom Sn, Sm, dan Sx masing-masing adalah kecepatan angin minimum, rerata, dan maksimum pada suatu pengukuran. Tipe datanya adalah float karena nilainya dapat berupa bilangan desimal.

```
mysql> describe 1R2;
+-----+-----+
| Field      | Type      |
+-----+-----+
| timestamp  | datetime  |
| Ta         | float     |
| Ua         | float     |
| Pa         | float     |
+-----+-----+
```

Gambar 4.11 Contoh tabel MySQL untuk data pembacaan suhu, tekanan, dan kelembapan udara (header xR2).

Tabel kedua adalah untuk sensor suhu, kelembapan, dan tekanan udara. Tabel ini terdiri dari kolom timestamp, Ta, Ua, dan Pa. Kolom Ta, Ua, dan Pa masing-masing adalah suhu, kelembapan relatif, dan tekanan udara pada suatu pengukuran. Tipe datanya adalah float karena nilainya dapat berupa bilangan desimal.

```
mysql> describe 1R3;
+-----+-----+
| Field      | Type      |
+-----+-----+
| timestamp  | datetime  |
| Rc         | float     |
| Rd         | int       |
| Ri         | float     |
| Rp         | float     |
+-----+-----+
```

Gambar 4.12 Contoh tabel MySQL untuk data dari pembacaan sensor curah hujan (header xR3).

Tabel ketiga adalah untuk sensor curah hujan. Terdapat kolom Rc, Rd, Ri, dan Rp, yang masing-masing adalah akumulasi curah hujan, durasi hujan dalam detik, intensitas hujan, dan intensitas maksimum hujan pada suatu rentang pengukuran dengan durasi 10 detik. Kolom Rc, Ri, dan Rp bertipe data float, sedangkan Rd bertipe integer karena durasi hujan yang terukur adalah dalam satuan detik antara 0 hingga 10. Meskipun ada data lain yaitu Hc, Hd, Hi, dan Hp yang merupakan pengukuran curah *hail*, hasil pengukuran ini tidak menjadi perhatian mendalam.

```
mysql> describe 1R5;
+-----+-----+
| Field      | Type      |
+-----+-----+
| timestamp  | datetime  |
| Th         | float     |
| Vh         | varchar(16)|
| Vs         | float     |
| Vr         | float     |
| Id         | varchar(16)|
+-----+-----+
```

Gambar 4.13 Contoh tabel MySQL untuk data *supervisor messages* (header xR5).

Tabel yang terakhir adalah dari *supervisor messages* atau frame data dengan header xR5. Field datanya adalah Th, Vh, Vs, Vr, dan Id, yang masing-masing adalah suhu dari *heater*, tegangan dari *heater* atau pesan error jika *heater* tidak diaktifkan, tegangan sumber, tegangan referensi, dan *identifier* dari sensor. *Identifier* adalah kode dari sensor yang secara unik akan membedakan dari sensor-sensor lainnya. *Identifier* ini berbeda dengan ID pada *header* dari frame.

Proses logging data hasil parsing ke database menggunakan kode seperti pada gambar 4.14. Sebelum disimpan ke database, terlebih dahulu dibuat tabelnya jika sensor belum memiliki tabel yang terkait, melalui fungsi `checkIfTableExist`. Setelah itu, dicek apakah hasil parsingnya sudah sesuai dengan tabel yang dibuat atau belum melalui fungsi `validateSensorData`. Setelah itu dilakukan query untuk menyimpan data ke dalam tabel, dengan beberapa bagian query diambil dari

lookup-table.js yang berisi konstanta-konstanta. Jika proses penyimpanan berhasil, maka data yang berada di database dapat diambil untuk diproses lebih lanjut pada aplikasi visualisasi.

```
// MySQL database logger
let mysql = require('mysql');
const SQLConnection = require("./sql.js");

const {SQLFields, SQLTables} = require("./lookup-table.js");
const {SensorName} = require("./lookup-table.js");

const timestampToSQL = (timestamp) => {
  return new Date(timestamp).toISOString().slice(0, 19).replace('T', ' ');
};

const checkIfTableExist = (sensor) => {
  let sql = `CREATE TABLE IF NOT EXISTS ${sensor.id}${sensor.type} ${SQLTables[sensor.type]}`;
  SQLConnection.query(sql, function (err, result) {
    if (err) throw err;
  });
}

const validateSensorData = (sensor) => { }

const log = (sensor) => {
  checkIfTableExist(sensor);
  let inserts = validateSensorData(sensor);

  let sql = mysql.format(`INSERT INTO ${sensor.id}${sensor.type} ${SQLFields[sensor.type]} VALUES ?`, inserts);
  SQLConnection.query(sql, function (err, result) {
    if (err) throw err;
    //console.log(`Number of records inserted: ${result.affectedRows}`);
  });
}

module.exports.log = log;
module.exports.checkIfTableExist = checkIfTableExist;
```

Gambar 4.14 Kode untuk melakukan logging data ke database MySQL.

```
const SQLTables = {
  R1: "(timestamp DATETIME, Dn INT(4), Dm INT(4), Dx INT(4), Sn FLOAT(4), Sm FLOAT(4), Sx FLOAT(4))",
  R2: "(timestamp DATETIME, Ta FLOAT(4), Ua FLOAT(4), Pa FLOAT(4))",
  R3: "(timestamp DATETIME, Rc FLOAT(4), Rd INT(4), Ri FLOAT(4), Rp FLOAT(4))",
  R5: "(timestamp DATETIME, Th FLOAT(4), Vh VARCHAR(16), Vs FLOAT(4), Vr FLOAT(4), Id VARCHAR(16))"
};

const SQLFields = {
  R1: "(timestamp, Dn, Dm, Dx, Sn, Sm, Sx)",
  R2: "(timestamp, Ta, Ua, Pa)",
  R3: "(timestamp, Rc, Rd, Ri, Rp)",
  R5: "(timestamp, Th, Vh, Vs, Vr, Id)"
};
```

Gambar 4.15 Bagian dari lookup-table.js yang digunakan dalam pembuatan *query*.

4.2.4 Pengembangan API

```
// Backend
require('dotenv').config();

const express = require('express');
const path = require('path');
const cors = require('cors');

const DataListener = require("./data-listener.js");
const Windrose = require("./windrose.js");
const Rain = require("./rain.js");

const app = express();
const serverPort = 4000;

app.use(cors());

app.use('/windrose', async (req, res) => {
  let data = await Windrose.getData();
  res.status(200).send(JSON.stringify(data, null, 2));
});

app.use('/rain', async (req, res) => {
  let data = await Rain.getData();
  res.status(200).send(JSON.stringify(data, null, 2));
});

app.use('/data', async (req, res) => {
  let data = DataListener.getData();
  res.status(200).send(JSON.stringify(data, null, 2));
});

app.use(express.static(path.join(__dirname, 'build')));
app.get('/', function (req, res) {
  res.sendFile(path.join(__dirname, 'build', 'index.html'));
});

app.use((err, req, res, next) => {
  console.log(err.message);
  console.log(JSON.stringify(err.stack));
});

app.listen(serverPort, () => {
  console.log(`Server started on port ${serverPort}`);
})
```

Gambar 4.16 Kode back-end yang melayani beberapa API *endpoint*.

Dalam pengembangan aplikasi, bagian back-end berisi logika aplikasi untuk mengolah dan melayani permintaan data dari pengguna. Pada aplikasi ini, terdapat beberapa API *endpoint* yang dapat diakses oleh sisi pengguna untuk mendapatkan data yang diinginkan. API *endpoint* yang dibuat yaitu `‘/data’`, `‘/windrose’`, dan

‘/rain’. Adapun *endpoint* ‘/’ digunakan untuk menampilkan halaman utama dari antarmuka aplikasi, yang merupakan hasil proses build dari bagian front-end.

Endpoint ‘/data’ dapat digunakan oleh sisi pengguna untuk meminta data terakhir yang berhasil dibaca dan dilakukan parsing oleh data parser. Data dapat berupa pembacaan sensor angin, suhu, curah hujan, maupun *supervisor messages*, tergantung pada frame jenis apa yang terakhir diproses oleh parser, sebagaimana kode pada gambar 4.5. Endpoint ini memberikan nilai return berupa JSON object yang dapat digunakan untuk melakukan update nilai dari tampilan real-time pada aplikasi visualisasi.

```
SELECT case
  when Dm >= 349 or Dm <= 011 then 'N'
  when Dm between 012 and 033 then 'NNE'
  when Dm between 034 and 056 then 'NE'
  when Dm between 057 and 078 then 'ENE'
  when Dm between 079 and 101 then 'E'
  when Dm between 102 and 123 then 'ESE'
  when Dm between 124 and 146 then 'SE'
  when Dm between 147 and 168 then 'SSE'
  when Dm between 169 and 191 then 'S'
  when Dm between 192 and 213 then 'SSW'
  when Dm between 214 and 236 then 'SW'
  when Dm between 237 and 258 then 'WSW'
  when Dm between 259 and 281 then 'W'
  when Dm between 282 and 303 then 'WNW'
  when Dm between 304 and 326 then 'NW'
  when Dm between 327 and 348 then 'NNW'
end as direction,
sum(case when Sm <= 2.0 then 1 else 0 end) AS calm,
sum(case when Sm <= 5.0 then 1 else 0 end) AS underFive,
sum(case when Sm <= 12.0 then 1 else 0 end) AS underTwelve,
sum(case when Sm <= 20.0 then 1 else 0 end) AS underTwenty,
sum(case when Sm <= 30.0 then 1 else 0 end) AS underThirty,
count(Sm) AS total
FROM IR1
WHERE
  (timestamp between CURRENT_TIMESTAMP() - interval 24 HOUR
   and CURRENT_TIMESTAMP())
GROUP BY direction;
```

Gambar 4.17 SQL query pengambilan data untuk diagram windrose.

Endpoint ‘/windrose’ dapat digunakan oleh sisi pengguna untuk meminta data dari sensor angin meliputi arah angin dan kecepatan angin dalam rentang waktu tertentu yang dikelompokkan berdasarkan arahnya. Pengelompokan dilakukan melalui SQL query sesuai pada gambar 4.17, sehingga hasil query dari database MySQL akan langsung menghasilkan kelompok-kelompok data sensor angin yang diinginkan seperti pada gambar 4.18. Data yang telah dikelompokkan tersebut kemudian menjadi nilai return dari endpoint ini, dan dapat digunakan dalam tampilan diagram windrose dan diagram distribusi kecepatan angin pada tampilan UI.

direction	calm	underFive	underTwelve	underTwenty	underThirty	total
SE	21	78	155	158	158	158
SSE	18	97	224	241	241	241
S	30	126	269	282	282	282
SSW	21	93	137	146	146	146
SW	17	101	155	155	155	155
ESE	16	62	99	99	99	99
W	14	49	75	75	75	75
WSW	20	63	84	84	84	84
E	13	59	83	84	84	84
ENE	8	58	97	97	97	97
N	9	33	42	42	42	42
WNW	11	75	124	125	125	125
NNW	15	62	88	88	88	88
NW	21	68	96	101	101	101
NNE	10	27	45	47	47	47
NE	8	46	75	78	78	78

Gambar 4.18 Contoh hasil query sebagai data untuk membuat diagram windrose.

Endpoint ‘/rain’ dapat digunakan oleh sisi pengguna untuk meminta data dari sensor curah hujan, meliputi akumulasi curah hujan dan durasi hujan. Data dikelompokkan berdasarkan jamnya, melalui SQL query seperti pada gambar 4.19. Hasil dari query menjadi nilai return dari endpoint ini, dan dapat digunakan dalam tampilan curah hujan pada tampilan UI.

```

SELECT HOUR(timestamp) as hour,
       sum(Rc) as accumulation,
       sum(Rd) as duration,
       avg(Ri) as intensity,
       max(Rp) as maximum
FROM 1R1
WHERE
  (timestamp between CURRENT_TIMESTAMP() - interval 24 HOUR
   and CURRENT_TIMESTAMP())
GROUP BY hour;

```

Gambar 4.19 SQL query pengambilan data historis dari sensor curah hujan.

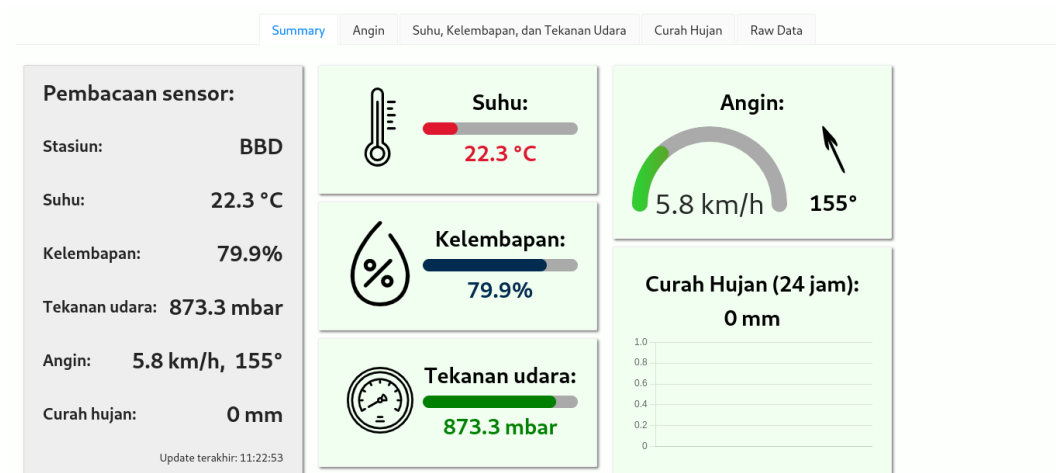
4.2.5 Pengembangan Tampilan *User Interface*

Pada pembuatan tampilan UI, digunakan library React.js dalam pembuatan komponen webnya dan library ant-design dalam styling komponen web. Tampilan UI terdiri dari beberapa tab, yaitu tab summary yang berisi visualisasi data real-time dan beberapa tab lainnya yang menampilkan data historis.

Pada tab summary, terdapat dua tipe presentasi data sensor cuaca. Tipe pertama adalah tampilan pembacaan langsung tanpa divisualisasikan dengan menarik, yang berada pada sisi kiri dari tampilan UI. Pada bagian ini, hasil pembacaan sensor ditampilkan langsung dalam bentuk teks dengan satuan pengukuran yang sesuai. Terdapat field ID dari stasiun, suhu dalam Celsius, kelembapan dalam %RH, tekanan udara dalam mbar, kecepatan angin dalam km/h dan arahnya dalam derajat, dan akumulasi curah hujan yang terukur dalam satuan mm. Ditampilkan pula waktu *update* atau *timestamp* dari pembacaan sensor yang terakhir diterima, agar dapat diamati apakah data diterima secara real-time ataupun terdapat gangguan.

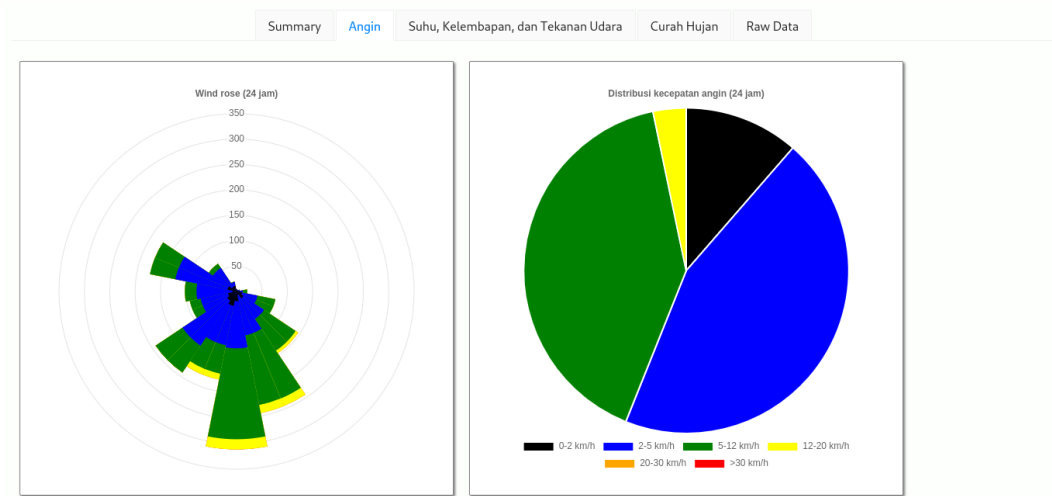
Tipe kedua adalah inti utama dari aplikasi yang dibuat, yaitu visualisasi dari tiap-tiap pembacaan sensor cuaca. Pada tampilan ini, data suhu, kelembapan, dan tekanan udara ditampilkan dalam bentuk bar dan disertai dengan ikon yang terkait. Data kecepatan angin ditampilkan dalam bentuk gauge, sedangkan ikon panah menunjukkan arah angin. Data curah hujan yang real-time ditampilkan

dalam bentuk teks saja, namun terdapat bar plot yang menunjukkan akumulasi curah hujan per jamnya dalam 24 jam terakhir.



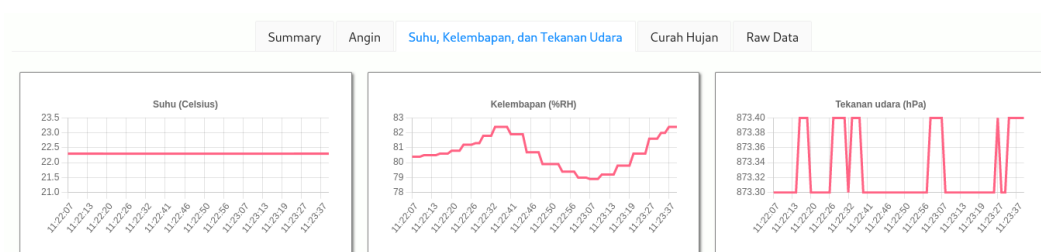
Gambar 4.20 Tampilan utama aplikasi pada tab *summary*.

Tab selanjutnya adalah tab “angin”, yang menampilkan dua buah diagram yaitu diagram windrose dan diagram distribusi kecepatan angin dalam bentuk pie chart. Pada visualisasi data dari sensor angin, kecepatan angin dikelompokkan menjadi beberapa kategori dan diberi warna tertentu untuk tiap-tiap kategorinya. Diagram windrose merepresentasikan sebaran arah angin dan kecepatan angin pada lokasi tertentu dalam suatu kurun waktu, sehingga dapat digunakan untuk mendeteksi tendensi gerakan angin. Pada dasarnya diagram windrose dapat dibuat menggunakan stacked bar plot namun dalam sistem koordinat polar. Sumbu r menunjukkan jumlah kumulatif dari pembacaan angin dengan kecepatan tertentu dan sumbu theta menunjukkan arah angin. Sedangkan diagram distribusi kecepatan angin menunjukkan persentase dari tiap-tiap kategori kecepatan yang juga diambil dalam kurun waktu tertentu.



Gambar 4.21 Tampilan pada tab “angin”.

Tab selanjutnya menampilkan data historis dari suhu, kelembapan, dan tekanan udara dalam bentuk diagram garis. Sensor suhu, kelembapan, dan tekanan udara diketahui memiliki periode pengiriman data yaitu tiap 2 detik, sehingga hanya ditampilkan maksimum 1000 titik data atau sekitar 15 menit terakhir saja.



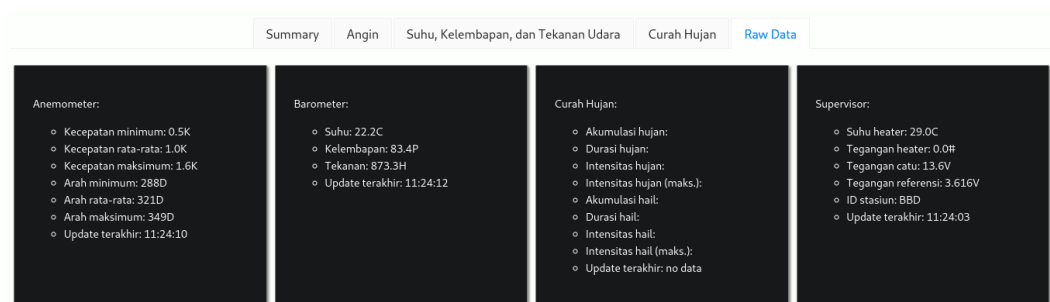
Gambar 4.22 Tampilan historis dari data suhu, kelembapan, dan tekanan udara.

Tampilan selanjutnya adalah tab curah hujan, yang menampilkan dua buah diagram batang yang menunjukkan curah hujan dan durasi hujan selama periode tertentu. Pada gambar 4.23, diagram masih kosong karena selama pengerjaan kerja praktik tidak terdeteksi hujan, sehingga tidak ada data yang diterima dan tampilan masih kosong.



Gambar 4.23 Tampilan pada tab “Curah Hujan”.

Tab yang terakhir adalah tab “raw data”, yang sesuai namanya menampilkan keseluruhan data real-time yang diterima tanpa pengolahan lebih lanjut. Fungsi dari tab ini adalah debugging, untuk mengecek apakah ada bagian dari sensor cuaca yang tidak mengirimkan data sebagaimana mestinya. Jika ada salah satu atau beberapa frame data yang bermasalah, maka informasi “update terakhir” akan tetap pada suatu nilai, yang berarti tidak ada data baru yang masuk. Selain itu, juga dapat digunakan untuk melihat data yang tidak divisualisasikan, seperti field data pada *supervisor messages*.



Gambar 4.24 Tampilan pada tab *raw data*.

```

function App() {
  const [labels, setLabels] = useState({});
  const [data, setData] = useState({});

  useEffect(() => {
    const id = setInterval(() => {
      fetch('http://localhost:4000/data')
        .then((response) => response.json())
        .then((sensorData) => {
          switch(sensorData.type) {
            // ...
          }
        })
        .catch((err) => {
          console.log(err.message);
        });
    }, 1000);
    return () => clearInterval(id);
  }, []);

  return (
    <div className="App">
      <Tabs centered size="large" type='card'
        style={{backgroundColor: "#FDFFFC", width: '100%', display: "inline-flex", padding: 10 }}
        items={[
          { label: 'Summary', key: 'item-summary', children:
            <div style={{display: "inline-flex", width: '100%'}}> ... </div> },
          { label: 'Angin', key: 'item-r1', children:
            <Row style={{ width: '100%', display: "inline-flex" }}> ... </Row> },
          { label: 'Suhu, Kelembapan, dan Tekanan Udara', key: 'item-r2', children:
            <Row style={{ width: '100%', display: "inline-flex" }}> ... </Row> },
          { label: 'Curah Hujan', key: 'item-r3', children:
            <Row style={{ width: '100%', display: "inline-flex" }}> ... </Row> },
          { label: 'Raw Data', key: 'item-raw', children:
            <Row style={{ width: '100%', display: "inline-flex" }}> ... </Row> }
        ]}
      />
    </div>
  );
}

export default App;

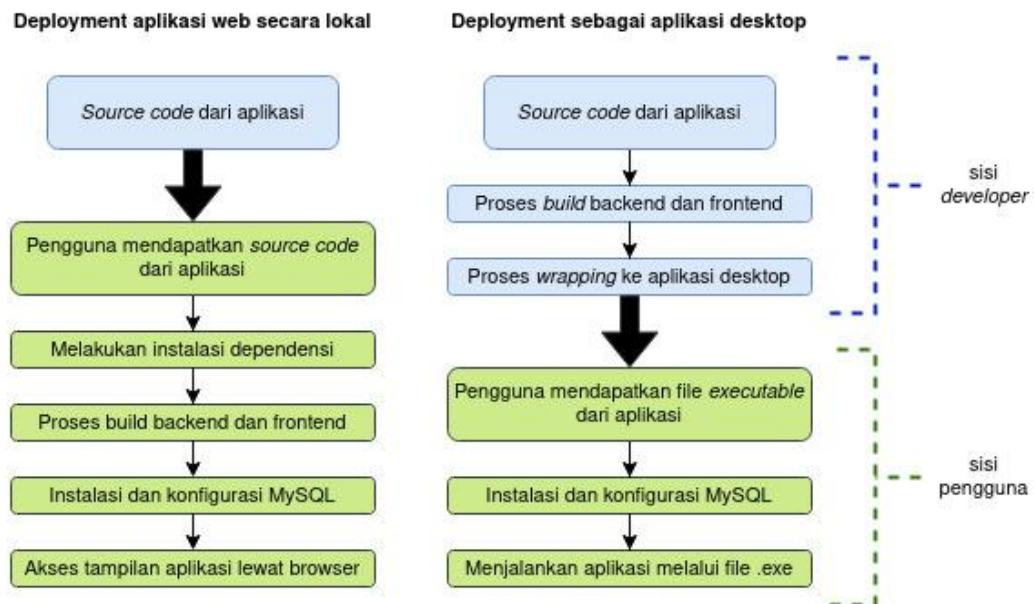
```

Gambar 4.25 Kode untuk tampilan utama dari aplikasi dan tab-tab terkait.

4.2.6 Wrapping ke Aplikasi Desktop

Setelah komponen dan fitur-fitur utama dari aplikasi telah diselesaikan, yang dilakukan selanjutnya adalah proses deployment. Karena aplikasi hanya bisa mengolah dan menampilkan data hanya dari satu sensor saja, maka opsi deployment ke suatu server dianggap terlalu berlebihan oleh pembimbing dan disarankan untuk diinstall pada satu komputer saja di stasiun pengamatan Gunung Merapi. Selain itu aplikasi yang telah dibuat adalah berbasis web, yang mungkin bagi orang awam cukup sulit untuk melakukan setup pada web server dan

tampilan front-end untuk menjalankan aplikasi. Oleh karena itu, dilakukan wrapping dari aplikasi web menjadi aplikasi desktop yang dapat dijalankan lewat executable file saja. Framework Electron.js digunakan untuk wrapping menjadi aplikasi desktop karena berbasis JavaScript dan cukup populer.



Gambar 4.26 Perbandingan proses instalasi dan penggunaan aplikasi berbasis web dengan aplikasi desktop.

Proses wrapping menggunakan Electron.js dilakukan menggunakan kode seperti pada gambar 4.26. Dibuat sebuah window yang akan menampilkan URL dari aplikasi, misalnya di localhost:4000/. Program back-end yang telah dibuat dimasukkan ke Electron dengan cara *import* seperti layaknya *module* pada umumnya. Dengan demikian, bagian back-end dapat bekerja pada aplikasi Electron yang telah dibuat. Adapun PowerSaveBlocker dan PowerMonitor digunakan untuk mengantisipasi interupsi dari sistem operasi pada komputer jika komputer masuk ke mode *sleep* atau *hibernate*, yang dapat melakukan interupsi bahkan terminasi jalannya aplikasi. Data parser dan logger tetap dapat bekerja dan visualisasi data tetap berjalan meskipun komputer dalam mode sleep.

```

// Electron app to wrap as a desktop app
require('dotenv').config();
const path = require('path');

const { BrowserWindow, app } = require('electron');
const { powerSaveBlocker } = require('electron');
const { powerMonitor } = require('electron');
const server = require('./app.js');

const serverPort = 4000;

let mainWindow = null;
const id = powerSaveBlocker.start('prevent-app-suspension')

function createWindow() {
  mainWindow = new BrowserWindow({
    width: 1366,
    height: 768
  });

  mainWindow.loadURL(`http://localhost:${serverPort}`);
  mainWindow.on("closed", function () {
    mainWindow = null;
    powerSaveBlocker.stop(id);
  });
}

app.on("ready", createWindow);
app.setAppLogsPath(path.join(__dirname, 'app.log'));

powerMonitor.on('resume', () => {
  console.log('The system is resuming');
  mainWindow.reload();
});

```

Gambar 4.27 Kode untuk wrapping aplikasi web menjadi aplikasi desktop.

BAB V

PENUTUP

5.1 Kesimpulan

Dalam pelaksanaan kerja praktik dan pembuatan laporan kerja praktik ini terdapat beberapa kesimpulan yang penulis dapatkan, yaitu:

1. Kerja praktik telah terlaksana dengan lancar dan penulis memperoleh ilmu mengenai sistem pengamatan pada Gunung Merapi, interkoneksi antara sensor dengan komputer pada jaringan BPPTKG, hingga pengembangan data parser dan aplikasi untuk visualisasi data.
2. Aplikasi yang telah dibuat telah berhasil menampilkan visualisasi data dari hasil parsing data sensor cuaca. Database untuk menyimpan hasil pengukuran sensor dan menampilkan visualisasi data historis juga berhasil dikembangkan.
3. Sebelum mengembangkan suatu aplikasi, perlu dipahami terlebih dahulu sistem yang akan dibuat dan hubungannya dengan keinginan atau kebutuhan dari pengguna. Dengan melakukan analisis terlebih dahulu, aplikasi maupun sistem yang dibuat akan sesuai dengan yang dibutuhkan pengguna.

5.2 Saran

Setelah melaksanakan rangkaian kerja praktik terdapat beberapa saran yang ingin penulis sampaikan, yaitu:

1. Pada aplikasi yang telah dikembangkan, masih terdapat *bug* atau *error* yang mungkin dapat menyebabkan *crash* pada aplikasi, sehingga perlu pengujian lebih lanjut.

2. Masih ada fitur-fitur pada aplikasi yang dapat dikembangkan lebih lanjut, misalnya tampilan yang lebih menarik dan responsif, fitur pengaturan rentang waktu data historis, hingga visualisasi untuk banyak sensor.
3. Ditambahkan metode untuk konfigurasi aplikasi dengan mudah dan nyaman. Pada aplikasi yang telah dibuat, konfigurasi dilakukan melalui *environment file* (.env) yang mungkin menyusahkan bagi pengguna awam.

DAFTAR PUSTAKA

- [1] BPPTKG. “Tugas dan Fungsi”. [Online]. Available: <https://bpptkg.esdm.go.id/pub/page.php?idf=3>. [Accessed 11 September 2022]
- [2] “Introducing JSON” [Online]. Available: <https://www.json.org/json-en.html>. [Accessed 15 September 2022].
- [3] “Introduction | Electron”. [Online]. Available: <https://www.electronjs.org/docs/latest/>. [Accessed 19 November 2022]
- [4] “MySQL 8.0 Reference Manual” [Online]. Available: <https://dev.mysql.com/doc/refman/8.0/en/>. [Accessed 12 September 2022].
- [5] “Vaisala - a global leader in environmental and industrial measurement” [Online]. Available: <https://www.vaisala.com/en>. [Accessed 5 September 2022]
- [6] Vaisala Oyj, “WXT510 User Guide in English” [Online]. Available: https://www.vaisala.com/sites/default/files/documents/WXT510_User_Guide_in_English.pdf. [Accessed 5 September 2022].
- [7] “What is User Interface (UI)?” [Online]. Available: <https://www.target.com/searchapparchitecture/definition/user-interface-UI>. [Accessed 15 September 2022]
- [8] “What is Data Parsing?” [Online]. Available: <https://nanonets.com/blog/what-is-data-parsing/>. [Accessed 7 September 2022].