BAB III
PELAKSANAAN KERJA PROFESI

3.1 Bidang Kerja

Pelaksanaan program Kerja Profesi (KP) di PT. Mangunggaling Rizki Karyatama Telnics (MARKTEL) dengan penempatan di cabang utama yaitu di Bandung. Praktikan bertanggung jawab atas pembuatan operasi *Create, Read, Update, dan Delete (CRUD) perangkat dan vendor* untuk sistem pemantauan kualitas udara pada PT. MARKTEL untuk di daerah Sumatera Utara, yang lebih tepatnya berada di kota Medan. Sesuai hasil dari sensor HCD-6818 yang ditempatkan di kota Medan, API *Postman* adalah sebuah aplikasi (berupa plugin), yang berfungsi sebagai *rest Client*, yang digunakan untuk melakukan uji coba rest API. Dengan adanya API data berhasil dikirim melalui API *Postman* yang terhubung ke server. Proses ini memungkinkan untuk membuat *CRUD* lebih efektif. API *Postman* adalah perangkat lunak yang memungkinkan pengembang menguji dan mengelola permintaan API dengan mudah. Dalam konteks ini, digunakan untuk mengirim data dari sensor HCD-6818 ke server dalam sistem pemantauan kualitas udara di kota Medan, Sumatera Utara.

Selama menjadi *engginer,* Praktikan mengerjakan proyek pembuatan operasi *CRUD* untuk pemantauan kualitas udara. Proyek ini sesuai dengan bidang studi Praktikan, yaitu bidang informatika. Selama di PT. MARKTEL, Praktikan mendapat bimbingan dari seorang mentor yang berpengalaman di bidangnya. Mentor tersebut membantu Praktikan memahami bagaimana bekerja langsung dengan seseorang yang ahli di bidang tersebut.

Sebagai Praktikan di PT. MARKTEL, Praktikan bertanggung jawab untuk mengembangkan sistem pemantauan kualitas udara HCD-6818 melalui serangkaian tahapan dalam pembuatan *CRUD*. *CRUD* tersebut menjadi sistem pendukung bagi berjalannya sistem pemantauan kualitas udara HCD 6818. Selain itu, *CRUD* tersebut juga berfungsi untuk mengetahui perangkat dan vendor, sehingga dapat memudahkan pemantauan sensor HCD-6818.

3.2 Pelaksanaan Kerja

Dalam upaya mengembangkan sistem sesuai dengan target yang telah diberikan, Praktikan dan Pembimbing Eksternal berkolaborasi dalam diskusi untuk merancang langkah-langkah kerja yang sistematis dan terstruktur, yaitu dengan membuat linimasa yang jelas untuk memandu proses pengembangan. Langkah-langkah ini dirancang dengan cermat agar mencapai tujuan yang telah ditentukan, dengan memastikan bahwa tahapan yang dilakukan secara berurutan.

Tabel 3.1 Linimasa Pelaksanaan Kerja Profesi

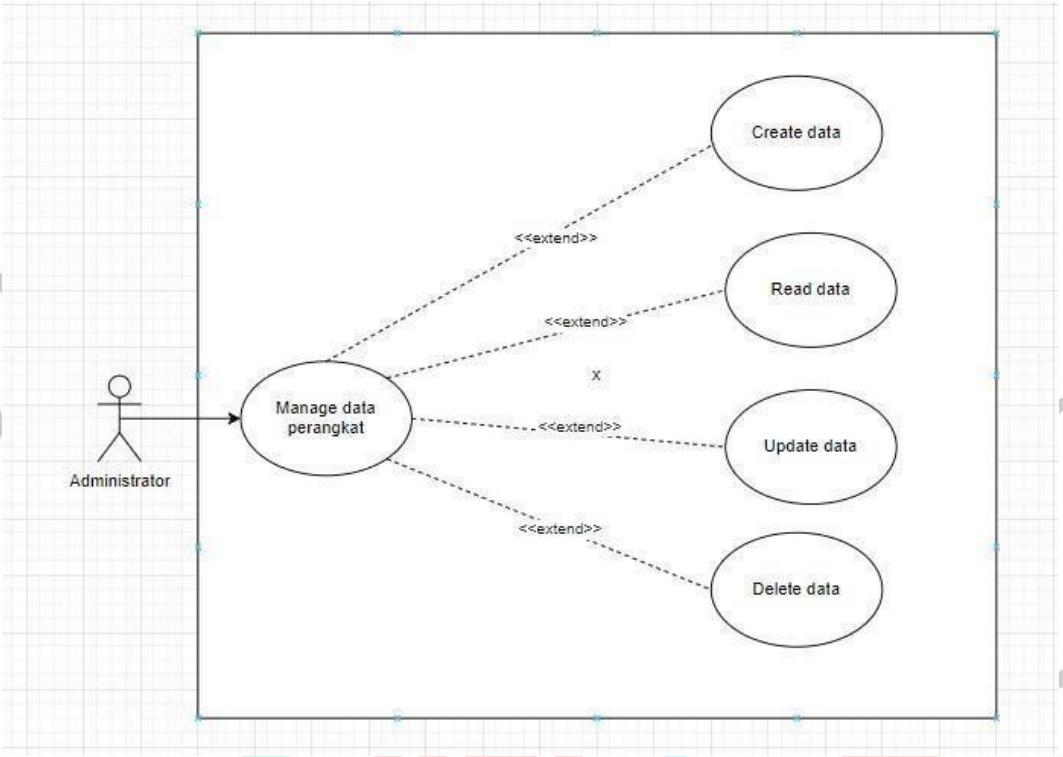
Aktivitas	Juni				Juli				Agustus			
	M1	M2	M3	M4	M1	M2	M3	M4	M1	M2	M3	M4
Analisis Kebutuhan sistem												
Perancangan struktur *database*												
Implementasi fungsional *CRUD*												
Pengujian integritas												
Penyusunan dokumentasi												
Presentasi Akhir												

Tabel 3.1 merupakan tahapan keseluruan yang di lakukan oleh Praktikan selama menjalankan program KP di PT. MARKTEL. Praktikan bertanggung jawab untuk pembuatan operasi *CRUD* perangkat dan vendor untuk sistem pemantauan kualitas udara. Tahap-tahapan tersebut mencakup analisa kebutuhan sistem, perancangan struktur *database*, implementasi fungsional *CRUD,* pengujian integritas, penyusunan dokumentasi dan presentasi akhir. Selama proses ini, Praktikan bekerja sama dengan tim di PT. MARKTEL untuk memastikan bahwa setiap tahap dilaksanakan dengan teliti guna mecapai hasil yang optimal sesuai dengan target yang telah ditetapkan sebelumnya.

3.2.1 Rekayasa Proses

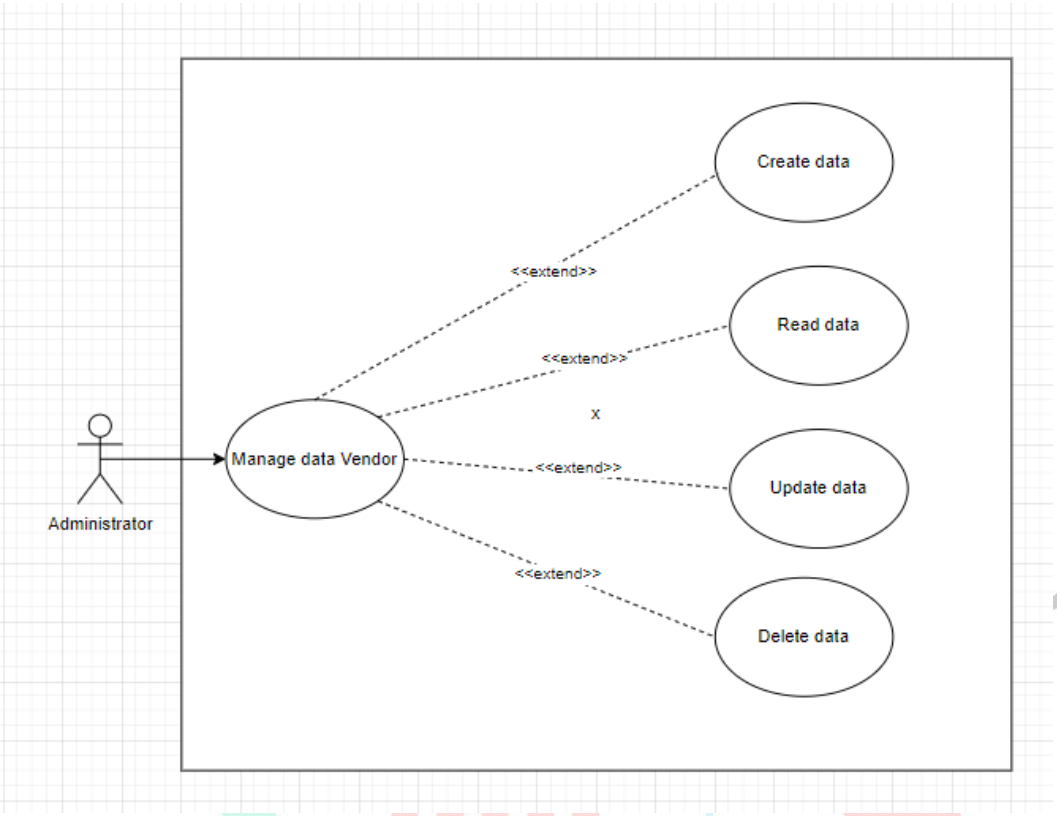
Dalam pembuatan operasi *CRUD*, Praktikan membuat usecase dan flowchart untuk menggambarkan alur proses kerja sistem secara detail

dan berurutan. Hal ini bertujuan untuk menjelaskan secara sistematis alur proses kerja aplikasi.



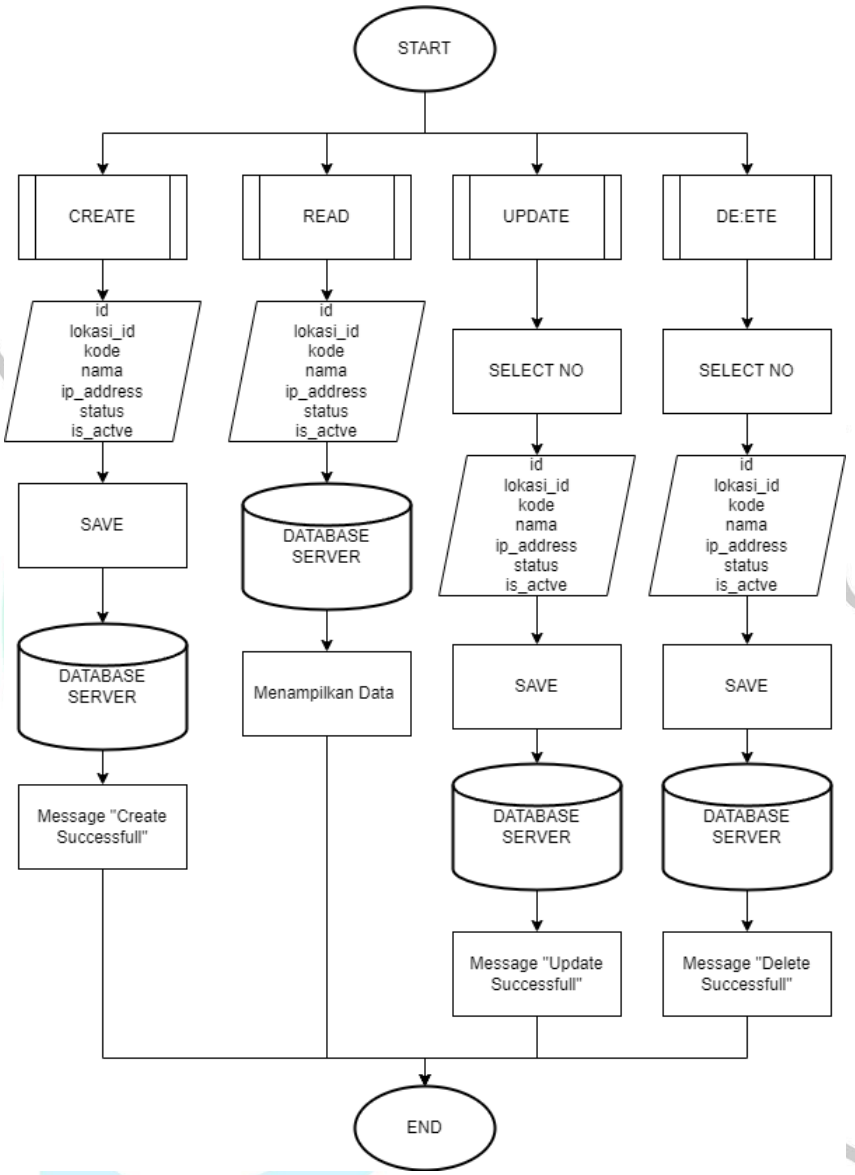
Gambar 3. 1 *Usecase* Perangkat

Gambar 3.4 menunjukkan operasi *CRUD* pada perangkat dapat direpresentasikan dalam bentuk *usecase*. Pada tahap pertama, seorang *Administrator* bertanggung jawab untuk mengelola data perangkat dalam sebuah sistem. *Usecase* mencakup beberapa operasi dasar pada data, yaitu *create, read, update,* dan *delete* data. Dalam manajamen perangkat, *Administrator* membuat entri data baru untuk setiap perangkat, serta menghapus entri data jika diperlukan.



Gambar 3. 2 *Usecase* Vendor

Gambar 3.2 menunjukkan operasi *CRUD* pada vendor dapat direpresentasikan dalam bentuk *usecase*. Pada tahap ini, seorang *Administrator* bertanggung jawab untuk mengelola data vendor dalam sebuah sistem. *Usecase* mencakup beberapa operasi dasar pada data, yaitu *create, read, update,* dan *delete* data. Dalam manajamen data vendor, *Administrator* membuat entri data baru untuk setiap vendor, serta menghapus entri data jika diperlukan.



Gambar 3. 3 *Flowchart* Perangkat

Gambar 3.3 menunjukkan operasi *CRUD* pada perangkat dapat direpresentasikan dalam bentuk *flowchart* untuk memvisualisasikan alur kerja. *Flowchart* ini menunjukan langkah-langkah dalam pengelolaan data perangkat dalam bentuk operasi *create, read, update,* dan *delete.* Praktikan dapat memahami secara sistematis proses yang terlibat dalam manajemen data perangkat.

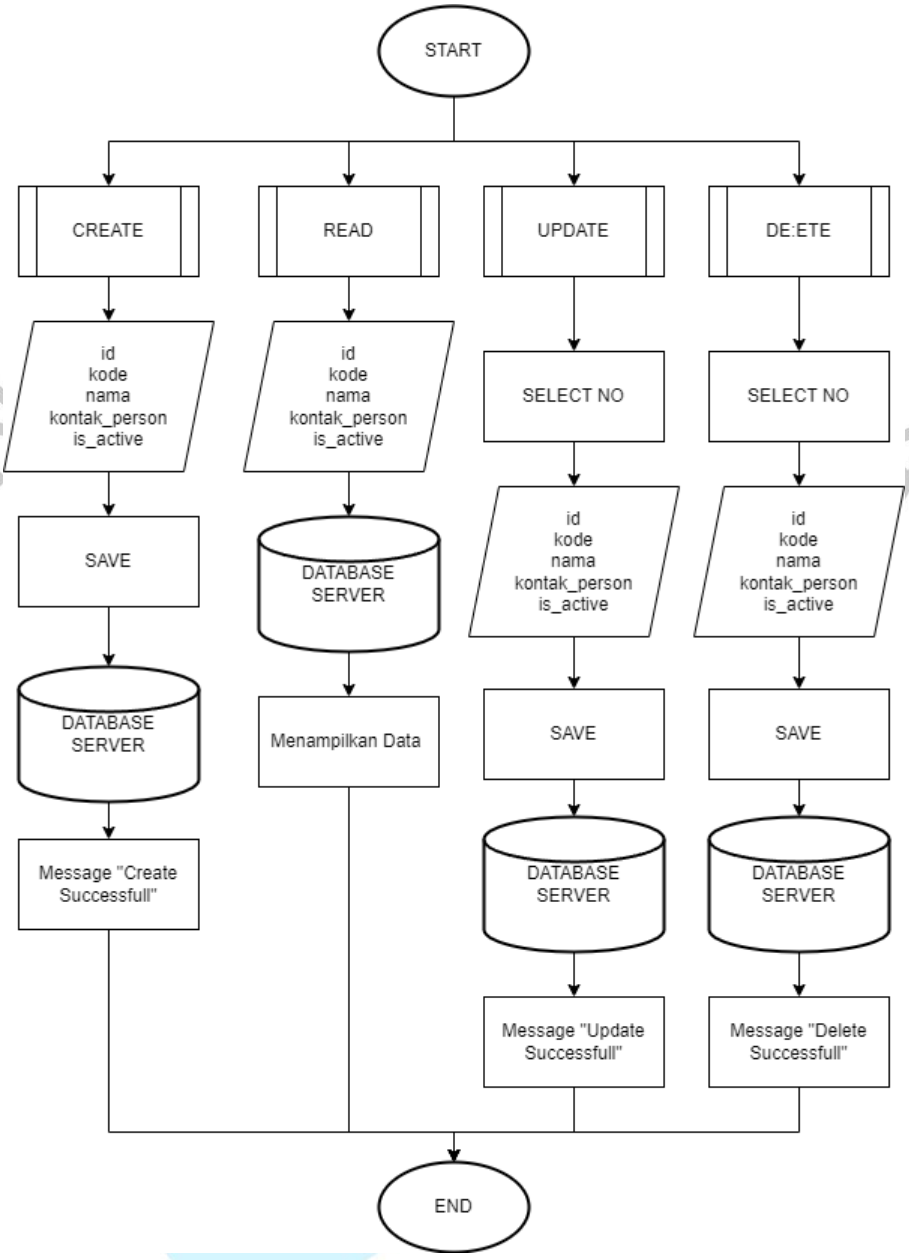
Dalam langkah pertama, Praktikan memulai data perangkat pada operasi *CRUD*, kemudian dilanjutkan meng*input* atau *create table* yang

berisi id, lokasi_id, kode, nama, *ip_address*, status, dan *is_active*. Langkah selanjutnya adalah menyimpan data *table* yang masuk ke dalam server. Jika data berhasil disimpan di server *database*, maka akan muncul pesan "*create successful*".

Pada langkah kedua, dilanjutkan untuk membaca atau mengambil informasi dalam *table* atau sistem. Dengan menggunakan operasi *read*, Praktikan atau sistem dapat mengakses data yang telah dibuat yang berisi id, lokasi_id, kode, nama, *ip_address*, status, dan *is_active*. Langkah selanjutnya adalah menampilkan data setelah sistem membaca *table.*

Langkah selanjutnya adalah *update* data yang berfungsi untuk memperbarui atau mengubah informasi yang telah ada dalam *table* atau sistem. Apabila terdapat data yang perlu diubah, Praktikan akan memilih *ID* atau nomor yang akan diubah. Selanjutnya, data tersebut akan diperbarui dan disimpan ke dalam *table database* pada server. Jika proses *update* berhasil, pesan "*update successful"* akan tampil, menandakan bahwa perubahan pada data telah berhasil masuk ke dalam *table database* server.

Pada langkah selanjutnya, operasi *delete* berfungsi untuk menghapus atau mengeliminasi data tertentu dari *table database* atau sistem. Apabila terdapat data yang perlu dihapus, pengguna akan memilih *ID*, lalu data tersebut akan dihapus dari *table* . Jika proses hapus berhasil, maka akan terkonfirmasi bahwa data telah berhasil dihapus dari sistem.



Gambar 3. 4 Flowchart Vendor

Gambar 3.4 menunjukkan operasi *CRUD* pada vendor dapat direpresentasikan dalam bentuk *flowchart* untuk memvisualisasikan alur kerja. *Flowchart* ini menunjukan langkah-langkah dalam pengelolaan data vendor dalam bentuk operasi *create, read, update,* dan *delete.* Praktikan dapat memahami secara sistematis proses yang terlibat dalam manajemen data vendor.

Dalam langkah pertama, Praktikan memulai data perangkat pada operasi *CRUD*, kemudian dilanjutkan meng*input* atau *create table* yang berisi id, kode, nama, kontak_person, dan is_active. Langkah selanjutnya adalah menyimpan data *table* yang masuk ke dalam server. Jika data berhasil disimpan di server *table database*, maka akan muncul pesan "*create successful*".

Pada langkah kedua, dilanjutkan untuk membaca atau mengambil informasi dalam *table* atau sistem. Dengan menggunakan operasi *read*, Praktikan atau sistem dapat mengakses data yang telah dibuat yang berisi id, kode, nama, kontak_person, dan is_active. Langkah selanjutnya adalah menampilkan data setelah sistem membaca *table.*

Langkah selanjutnya adalah *update* data yang berfungsi untuk memperbarui atau mengubah informasi yang telah ada dalam *table* atau sistem. Apabila terdapat data yang perlu diubah, Praktikan akan memilih *ID* atau nomor yang akan diubah. Selanjutnya, data tersebut akan diperbarui dan disimpan ke dalam *table database* pada server. Jika proses *update* berhasil, pesan "*update successful"* akan tampil, menandakan bahwa perubahan pada data telah berhasil masuk ke dalam *table database* server.

Pada langkah selanjutnya, operasi *delete* berfungsi untuk menghapus atau mengeliminasi data tertentu dari *table database* atau sistem. Apabila terdapat data yang perlu dihapus, pengguna akan memilih *ID*, lalu data tersebut akan dihapus dari *table* . Jika proses hapus berhasil, maka akan terkonfirmasi bahwa data telah berhasil dihapus dari sistem.

3.2.2 Spesifikasi *Database*

Spesifikasi *database* dalam pengembangan sistem operasi *CRUD*, Praktikan merancang *database* melalui localhost *database* terlebih dahulu dengan menggunakan PhpMyAdmin.

Table structure | Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	lokasi_id	int(11)			Yes	NULL			Change Drop More
3	kode	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
4	nama	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
5	ip_address	varchar(255)	utf8mb4_general_ci		Yes	NULL			Change Drop More
6	status	int(11)			Yes	NULL			Change Drop More
7	is_active	tinyint(1)			Yes	NULL			Change Drop More
8	createdAt	datetime			No	None			Change Drop More
9	updatedAt	datetime			No	None			Change Drop More
10	vendor_id	int(11)			Yes	NULL			Change Drop More
11	tipe_sensor_id	int(11)			Yes	NULL			Change Drop More

Gambar 3. 5 *Database* Tabel Perangkat

Gambar 3.5 merupakan *database* tabel perangkat. *Database* atau basis data adalah kumpulan informadi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data yang dibuat. (ANDARUA, 2018) untuk mengakomodir berbagai kebutuhan *database*. Praktikan menggunakan MySQL merupakan salah satu jenis server basis data yang sangat terkemuka. MySQL menggunakan bahasa SQL sebagai alat untuk mengakses basis data-nya. Lisensi MySQL melibatkan FOSS *License Exception*, dan terdapat juga versi komersialnya. MySQL dikenal dengan tagline "*The World's most popular open source database.*" (Sofwan, 2006) Sedangkan RDMS dirancang untuk mengatur sebuah *database* sebagai sekumpulan data yang disimpan secara terstruktur, dan melakukan operasi atas data atas permintaan penggunanya. (Pratama, 2006)

Praktikan menggunakan beberapa nama *database* untuk mengelola data tabel perangkat. Berikut merupakan nama *database* yang digunakan seperti id, lokasi_id, kode, nama, *ip_address*, status, *is_active*, *createAt, updateAt*, vendor_id, dan tipe_sensor_id.

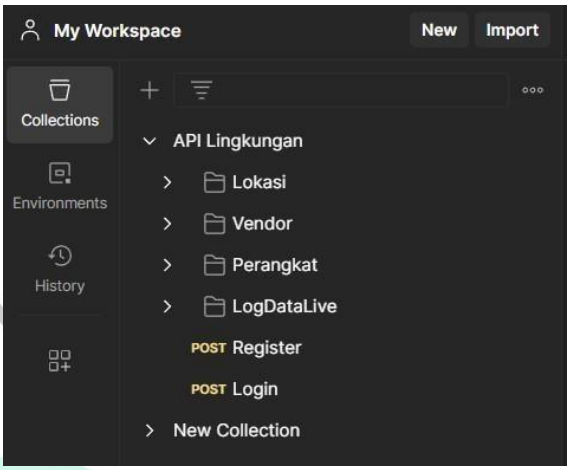
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	kode	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
3	nama	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
4	kontak_person	varchar(100)	utf8mb4_general_ci		No	None			Change Drop More
5	is_active	int(11)			No	None			Change Drop More
6	isActive	tinyint(1)			Yes	NULL			Change Drop More

Gambar 3. 6 *Database* Tabel Vendor

Gambar 3.6 merupakan *database* tabel vendor. Praktikan menggunakan beberapa nama *database* untuk mengelola data tabel vendor. Berikut merupakan nama *database* yang digunakan seperti id, kode, nama, kontak*_person*, dan *is_active*.

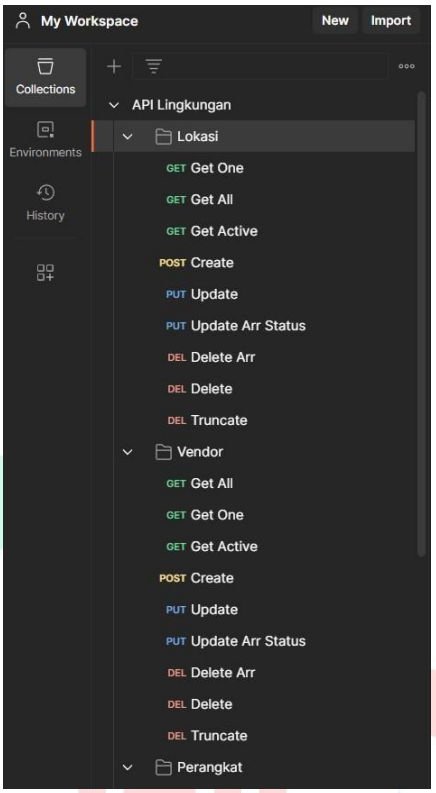
3.2.3. Tahap Perancangan Sistem

Dalam fase perancangan, *CRUD* merupakan proses standar yang biasa ditemukan pada sebuah sistem aplikasi. (Rohman, 2008) Praktikan menganalisa kebutuhan operasi sistem *CRUD* perangkat dan vendor untuk sistem pemantauan kualitas udara. Perancangan ini meliputi tahapan-tahapan operasi *CRUD*. Pada tahap pertama, pemanfaatan API lingkungan, seperti *Postman*, *Postman* memudahkan dalam menguji, mengembangkan, dan mendokumentasikan API. Fitur *Postman* yang dapat dipahami membuat pengujian API dapat dilakukan dengan baik dan cepat. Cara kerja *Postman* dengan mengklasifikasi request berdasarkan request method, URL dan parameter-parameter request (Anisa Selena Sari, 2022), *Postman* berperan penting. Melalui API ini, data yang diperlukan, seperti informasi perangkat dan vendor, dapat disediakan. Operasi *CRUD*, termasuk *POST, GET, PUT, dan DEL*, diterapkan untuk memastikan fungsionalitas yang lengkap. Data perangkat menjadi fokus utama, memberikan informasi tentang kondisi dan kualitas perangkat di kota Medan. Tahapan kedua adalah proses pembuatan *CRUD* menggunakan data yang diambil dari API *Postman* yang telah berisi data-data yang diperlukan.



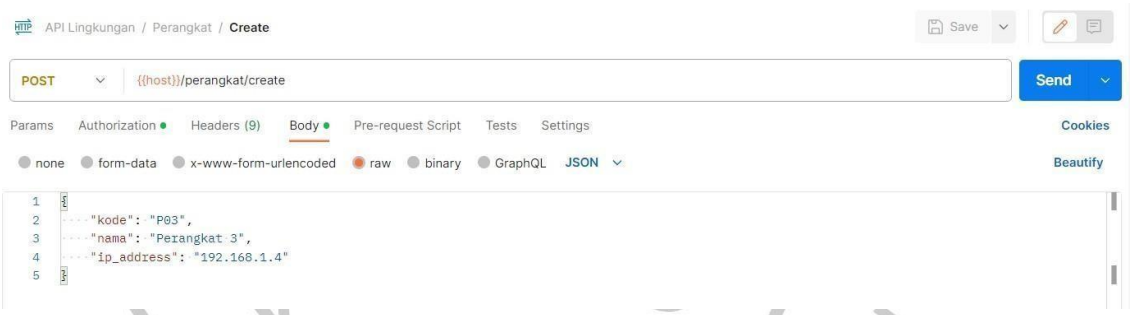
Gambar 3. 7 Data API Lingkungan

Gambar 3.7 menunjukkan tampilan antarmuka pengguna API *Postman* yang memuat data mengenai API lingkungan. Pada API *Postman* terdapat bagian *my workspace*, pada menu *collection* yang memuat data mengenai API lingkungan. Data yang di *input* dalam gambar tersebut mencakup beberapa *folder* yang berisi informasi terkait perangkat dan vendor. Melalui tampilan antarmuka tersebut, pengguna dapat dengan mudah mengakses dan mengelola informasi yang terstruktur mengenai lingkungan.



Gambar 3. *8 Post, Get, Put* dan *Del* API Lingkungan

Gambar 3.8 menunjukkan API lingkungan yang berfungsi sebagai penyimpan informasi terkait perangkat dan vendor, yang terorganisir dalam *folder-folder* terpisah. Setiap *folder* menyimpan data yang berbeda melalui metode *POST, GET, PUT, dan DEL.* Dengan menggunakan metode-metode tersebut pada setiap *folder*, Praktikan dapat melakukan operasi *POST, GET, PUT, dan DEL* data secara spesifik sesuai dengan kebutuhan. Struktur ini memudahkan pengelolaan dengan informasi API lingkungan, serta memberikan pengelolaan data perangkat dan vendor secara terpisah.



Gambar 3. 9 *Postman* Create perankat

Gambar 3. 9 menampilkan *create* perangkat yang divisualisasikan oleh *Postman* untuk pembuatan *CRUD* perangkat. Pada halaman *post* bagian *body*, pengguna dapat memasukkan data yang ingin di*input*kan untuk proses pembuatan *CRUD* perangkat. Terdapat tiga parameter utama pada bagian body yang dapat diidentifikasi, yaitu kode perangkat dengan nilai data P03, nama perangkat dengan nilai data perangkat 3, dan *ip address* dengan nilai data 192.168.1.4. Proses *input* data ini memberikan informasi penting terkait perangkat yang sedang dibuat. Melalui *Postman*, Praktikan dapat mengonfigurasi dan mengirim data ke sistem dengan efektif, memastikan bahwa setiap perangkat yang ditambahkan memiliki atribut yang sesuai dan akurat.

perangkat > perangkat_create.py

```python
import requests
import json
import mysql.connector

def create_perangkat():
    # Mengambil input dari pengguna melalui terminal
    kode = input("Masukkan kode: ")
    nama = input("Masukkan nama: ")
    ip_address = input("Masukkan ip_address: ")

    # Membuat objek lokasi dalam bentuk dictionary
    data = {
        "kode": kode,
        "nama": nama,
        "ip_address": ip_address
    }

    try:
        # Membuat koneksi ke database MySQL
        db_connection = mysql.connector.connect(
            host="localhost",       # Ganti dengan host database AndaP99
            user="root",            # Ganti dengan username database Anda
            password="",            # Ganti dengan password database Anda
            database="db_sensor"    # Ganti dengan nama database Anda
        )
```

Gambar 3. 10 Program *Create* Data Perangkat

Gambar 3.10 menunjukkan program untuk *add* data perangkat. Program tersebut menggunakan beberapa *library*, termasuk 'request', 'json', dan 'mysql.connector'. Pada tahap pertama, *library* 'request' digunakan untuk menyambungkan dengan API untuk melakukan pemintaan Hypertext Transfer – Transfer Protocol (HTTP). *Library* 'json' digunakan untuk manipulasi data dalam format json, serta 'mysl.connector' digunakan untuk menyambungkan dengan *database* MySQL. Selanjutnya, fungsi 'create_perangkat()' pada program bertujuan untuk entri perangkat baru. Program akan meminta *input* untuk kode perangkat, nama perangkat, dan *ip address* melalui terminal. Data yang di*input* kemudian disusun ke dalam bentuk dictionary dengan "kode", "nama", dan "ip_address". Selanjutnya program membuat koneksi ke *database* dengan menggunakan *library* 'mysql.connector' dan parameter seperti *host, username*, *password*, dan nama *database* yang disesuaikan sesuai konfigurasi *database*.

```
    # Membuat kursor
    cursor = db_connection.cursor()

    print(db_connection)

    # Menjalankan query INSERT untuk menyimpan data peranngkat baru
    insert_query = "INSERT INTO perangkat SET kode = %s, nama = %s,P ip_address = %s"
    values = (kode, nama, ip_address)
    cursor.execute(insert_query, values)

    # Melakukan commit perubahan ke database
    db_connection.commit()

    print("Data perangkat baru berhasil ditambahkan.")
except mysql.connector.Error as error:
    print("Terjadi kesalahan dalam menambahkan data perangkat:", error)
finally:
    # Menutup koneksi ke database
    if db_connection.is_connected():
        cursor.close()
        db_connection.close()
```

Gambar 3. 11 Sambungan Program *Create* Data Perangkat

Gambar 3.11 menunjukkan sambungan program *add* data perangkat yang menunjukkan langkah-langkah menambahkan data perangkat baru ke dalam *database*. Pada bagian pertama, *kursor* dibuat menggunakan objek ‘cursor()’ untuk menjalankan *query* ke *database*. Program kemudian mencetak status koneksi *database*. Pada bagian kedua, *query insert* dijalankan dengan menggunakan format parameterized *query* untuk menghindari SQL injection. Data perangkat yang telah di*input*kan sebelumnya melalui terminal dimasukkan ke dalam *query* menggunakan tuple ‘values’. Setelah *query* dijalankan, perubahan dicommit ke *database* dan program mencetak pesan sukses. Jika terjadi kesalahan selama eksekusi *query*, program akan menangani error dan mencetak pesan kesalahan.

```
PS C:\Users\KingBib\Desktop\vendor> & C:/Users/KingBib/AppData/Local/Programs/Python/Python311/python.exe c:/Users/KingBib/Desktop/vendor/perangka
t/perangkat_create.py
Masukkan kode: 11
Masukkan nama: perangkat 2
Masukkan ip_address: 192.168.1.2
<mysql.connector.connection_cext.CMySQLConnection object at 0x00000288E770610>
Data perangkat baru berhasil ditambahkan.
PS C:\Users\KingBib\Desktop\vendor>
```

Gambar 3. 12 Hasil *Penambahan* Program *Create* Data Perangkat

Gambar 3.12 menunjukkan hasil *penambahan* program *add* data perangkat. Pada eksekusi program, Praktikan diminta untuk memasukkan kode perangkat, nama perangkat dan alamat IP melalui terminal. Setelah pengguna memberikan *input* (P03, perangkat 3, 192.168.1.2). Program membuat koneksi ke *database* MySQL, membuat *kursor*, dan menjalankan *query insert* untuk menyimpan data perangkat baru. Jika koneksi *database* berhasil, maka program akan mengirimkan pesan “data perangkat baru berhasil ditambahkan”.

```
import requests
import json
import mysql.connector

def update_perangkat():
    # Mengambil kode dari user input
    id = input("Masukkan id perangkat yang akan diperbarui: ")

    # Mengambil input parameter dari user
    nama = input("Masukan nama:")
    kode = input("Masuan kode: ")
    ip_address = input("Masukkan ip_address: ")
    is_active = bool(input("Masukkan status is_active (True/False): "))

    # Membuat objek data perangkat dalam bentuk dictionary
    data = {
        "id" : id,
        "kode": kode,
        "nama": nama,
        "ip_address" : ip_address,
        "is_active": is_active
    }
```

Gambar 3. 13 Program *Update* Perangkat

Gambar 3.13 menunjukkan program untuk *update* data perangkat. Program tersebut menggunakan beberapa *library*, termasuk ‘request’, ‘json’, dan ’mysql.connector’. Pada tahap pertama, *library* ‘request’ digunakan untuk menyambungkan ke API untuk melakukan permintaan Hypertext Transfer – Transfer Protocol (HTTP). *Library* ‘json’ digunakan untuk manipulasi data dalam format json, serta ‘mysl.connector’ digunakan untuk menyambungkan dengan *database*

MySQL. Selanjutnya, fungsi ‘update_perangkat()’ bertujuan untuk memperbarui informasi perangkat pada sistem. Program diawali dengan meminta Praktikan untuk memasukan *id* perangkat yang akan diperbarui melalui terminal. Program kemudian mengambil *input* pengguna untuk parameter seperti nama perangkat, kode perangkat, alamat IP, dan status is_active yang dibutuhkan. *Input* ini kemudian disusun ke dalam bentuk dictionary dengan kunci-kunci "*id*", "kode", "nama", "*ip_address*", dan "*is_active*". Program diminta untuk memberikan status is_active dalam format Boolean (True/False). Selanjutnya, dictionary ini digunakan untuk memperbarui informasi perangkat dalam *database*.

```
    try:
        # Membuat koneksi ke database MySQL
        db_connection = mysql.connector.connect(
            host="localhost",       # Ganti dengan host database Anda
            user="root",            # Ganti dengan username database Anda
            password="",            # Ganti dengan password database Anda
            database="db_sensor"    # Ganti dengan nama database Anda
        )

        # Membuat kursor
        cursor = db_connection.cursor()

        print(db_connection)

        # Menjalankan query INSERT untuk menyimpan data perangkat baru
        insert_query = "UPDATE perangkat SET kode = %s, nama = %s, ip_address = %s, is_active = %s WHERE id = %s"
        values = (kode, nama, ip_address, is_active, id)
        cursor.execute(insert_query, values)

        # Melakukan commit perubahan ke database
        db_connection.commit()

        print("Data perangkat baru berhasil di Update.")
    except mysql.connector.Error as error:
        print("Terjadi kesalahan dalam Update data perangkat:", error)
    finally:
        # Menutup koneksi ke database
        if db_connection.is_connected():
            cursor.close()
            db_connection.close()
# Memanggil fungsi untuk memperbarui data perangkat
update_perangkat()
```

Gambar 3. 14 Sambungan Program *Update* Perangkat

Gambar 3.14 merupakan sambungan program *update* data perangkat yang menunjukan langkah-langkah memperbarui data perangkat baru ke dalam *database*. Program membuat *kursor* untuk menjalankan *query*. Praktikan memasukan *id* perangkat yang akan diperbarui dan memberikan *input* untuk parameter-parameter seperti kode, nama, *IP address*, dan status aktivasi is_active. Informasi

program disusun dalam bentuk dictionary dan digunakan dalam *query update* SQL untuk memperbarui data perangkat yang sesuai dengan *id* yang telah dimasukkan. Setelah menjalankan *query*, program akan mencetak pesan sukses setelah perubahan dicommit ke *database*. 'except' berfungsi untuk menangani kesalahan yang mungkin terjadi selama proses *update*. 'finnaly' befungsi untuk memastikan penutupan koneksi ke *database.*

Gambar 3. 15 Hasil Program *Update* Perangkat

Gambar 3.15 menunjukkan hasil *input* program *update* data perangkat. Pada eksekusi program, Praktikan diminta untuk memasukan *ID* perangkat, kode perangkat, *IP address*, dan status aktivasi is_active melalui terminal. Setelah pengguna memberikan *update input* (16, 2, perangkat 2, 192.168.1.2, true), program membuat koneksi ke *database* MySQL, membuat *kursor*, dan menjalankan *query update* untuk menyimpan data perangkat baru. Jika koneksi *database* berhasil, maka program akan mengirimkan pesan "data perangkat baru berhasil diubah".

```python
import requests
import mysql.connector

def get_perangkat():
    try:
        conn = mysql.connector.connect(
            host='localhost',
            user='root',
            password='',
            database='db_sensor',
        )

        # Mengambil kode dari user input
        id = input("Masukkan id perangkat: ")

        cursor = conn.cursor()
        insert_query = "SELECT * FROM perangkat WHERE id = %s"
        values = (id,)
        cursor.execute(insert_query, values)

        # Mengambil hasil dari kueri
        result = cursor.fetchone()
```

Gambar 3. 16 Program *Read* Perangkat

Gambar 3.16 menunjukkan program untuk *read* data perangkat. Program tersebut menggunakan beberapa *library*, termasuk 'request', 'json', dan 'mysql.connector'. Pada tahap pertama, *library* 'request' digunakan untuk menyambungkan dengan API untuk melakukan permintaan *Hypertext Transfer – Transfer Protocol* (*HTTP*). *library* 'json' digunakan untuk manipulasi data dalam format json, serta 'mysl.connector' digunakan untuk menyambungkan dengan *database* MySQL. Selanjutnya, fungsi 'get_perangkat()' bertujuan untuk mengambi informasi perangkat dari *database*, berdasarkan *ID* yang dimasukkan. Kemudian, program membuat *kursor* dan menjalankan *query select* SQL dengan paramater *query* untuk mengambil data perangkat yang sesuai dengan *ID* yang dimasukkan. Hasil *query* dieksekusi dengan menggunakan metode 'fetchone()'.

```
        if result:
            # Jika data ditemukan
            print(result)
        else:
            # Jika data tidak ditemukan
            print("Data tidak ditemukan")

    except Exception as e:
        print(f"Terjadi kesalahan: {e}")

    finally:
        # Pastikan untuk menutup kursor dan koneksi
        cursor.close()
        conn.close()


# Memanggil fungsi untuk mengambil data perangkat
get_perangkat()
```

Gambar 3. 17 Sambungan Program *Read* Perangkat

Gambar 3.17 menunjukkan sambungan program *read* data perangkat yang menunjukkan langkah-langkah mengambil data perangkat baru ke dalam *database*. 'if result' berfungsi sebagai memeriksa apakah hasil pengambilan data tidak kosong. Jika data perangkat ditemukan, program mencetak hasilnya ke layar menggunakan 'print(result)'. Jika hasilnya kosong, program akan mencetak pesan "data tidak ditemukan". 'except' berfungsi untuk menangani kesalahan yang mungkin terjadi selama proses pengambilan. 'finnaly' berfungsi untuk memastikan penutupan koneksi ke *database* setelah proses selesai.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\KingBib\Desktop\vendor> & C:/Users/KingBib/AppData/Local/Programs/Python/Python311/python.exe c:/Users/KingBib/Desktop/vendor/perangka
t/perangkat_get_one.py
Masukkan id perangkat: 16
(16, None, '2', 'perangkat 2', '192.168.1.2', None, 1, None, None, None, None)
PS C:\Users\KingBib\Desktop\vendor>
```

Gambar 3. 18 Hasil *read* Perangkat

Gambar 3.18 menunjukkan hasil *read* data perangkat. Pada eksekusi program, Praktikan diminta untuk memasukan *ID* untuk memilih data melalui terminal. Setelah pengguna memberikan *get input* (16), program membuat koneksi ke *database* MySQL, membuat *kursor*, dan menjalankan query *select* untuk memilih data perangkat baru. Koneksi *database* berhasil dan program kemudian mengirim pesan “16, none, perangkat 2, 192.168.1.2, none, 1” merupakan data yang dipilih.

perangkat > perangkat_delete.py > delete_perangkat

```python
import requests
import mysql.connector

def delete_perangkat():
    # Mengambil kode dari user input
    id = input("Masukkan id perangkat yang akan dihapus: ")

    try:
        # Membuat koneksi ke database MySQL
        db_connection = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="db_sensor"
        )

        # Membuat kursor
        cursor = db_connection.cursor()

        # Menjalankan query DELETE untuk menghapus data perangkat
        delete_query = "DELETE FROM perangkat WHERE id = %s"
        values = (int(id),)  # Ubah ID ke tipe data int dan buat tupel
        cursor.execute(delete_query, values)

        # Melakukan commit perubahan ke database
        db_connection.commit()

        print("Data perangkat berhasil dihapus.")
    except mysql.connector.Error as error:
        print("Terjadi kesalahan dalam menghapus data perangkat:", error)
    finally:
        # Menutup koneksi ke database
        if db_connection.is_connected():
            cursor.close()
            db_connection.close()
# Memanggil fungsi untuk menghapus data perangkat
delete_perangkat()
```

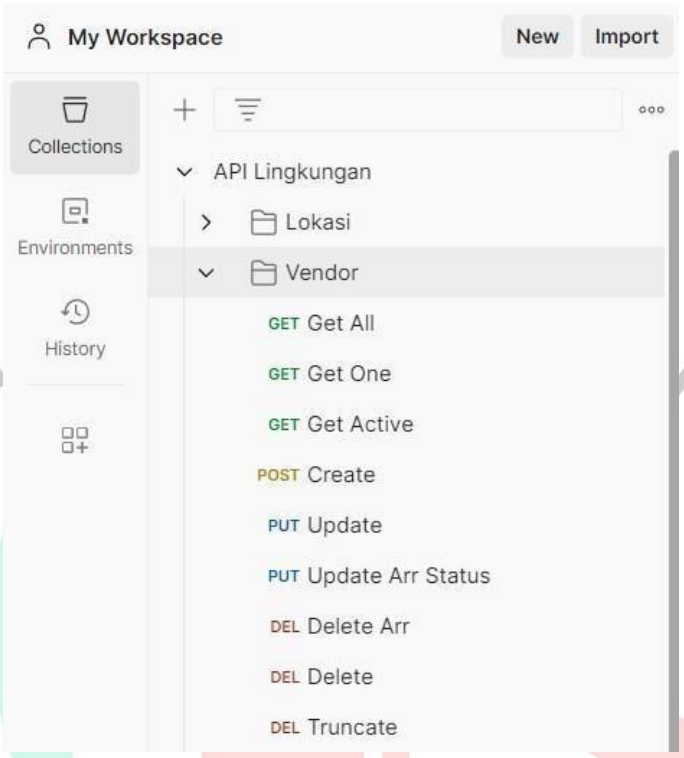
Gambar 3. 19 Program *Delete* Perangkat

Gambar 3.19 menunjukkan program untuk *delete* data perangkat. Program tersebut menggunakan beberapa *library*, termasuk ‘request’,

'json', dan 'mysql.connector'. Pada tahap pertama, *library* 'request' digunakan untuk menyambungkan dengan API untuk melakukan permintaan *Hypertext Transfer – Transfer Protocol* (*HTTP*). *Library* 'json' digunakan untuk manipulasi data dalam format json, serta 'mysl.connector' digunakan untuk menyambungkan dengan *database* MySQL. Program memasukkan *ID* perangkat yang akan dihapus. Selanjutnya, program membuat *kursor* untuk menjalankan *query delete* SQL dengan parameter *query* , dengan *ID* yang dimasukkan ke dalam tipe data integer. Setelah *query* dijalankan, perubahaan dicommit ke *database*. Lalu, program mencetak pesan "data perangkat berhasil dihapus". 'except' berfungsi untuk menangani kesalahan yang mungkin terjadi selama proses *delete*. 'finnaly' befungsi untuk memastikan penutupan koneksi ke *database* setelah proses selesai.

Gambar 3. 20 Hasil Delete Perangkat

Gambar 3.20 menunjukkan hasil *input* program *delete* data perangkat. Pada eksekusi program, Praktikan diminta untuk memasukan *ID* untuk memilih data melalui terminal. Setelah pengguna memberikan *get input* (16), program membuat koneksi ke *database* MySQL, membuat *kursor*, dan menjalankan *query delete* untuk menghapus data perangkat. Koneksi *database* berhasil dan program berhasil menghapus data tersebut.



Gambar 3. 21 Data *Post* Vendor

Gambar 3.21 menunjukkan tampilan API lingkungan yang berfungsi sebagai penyimpan informasi terkait perangkat dan vendor. API ini menyediakan akses ke berbagai data vendor melalui metode HTTP, yaitu *POST, GET, PUT, dan DEL*. Metode *get* terdapat *get all*, *get one*, dan *get active*, metode *post* mencakup operasi *create*, Metode *put* melibatkan *put update* dan *put* arr status, dan metode *del* terdapat *delete* arr, *delete*, dan *truncate*.

vendor > post_vendor_final.py > create_vendor

```python
import mysql.connector

def create_vendor():

    # Mengambil input dari pengguna melalui terminal
    kode = input("Masukkan kode: ")
    nama = input("Masukkan nama: ")
    kontak_person = input("Masukkan kontak_person: ")
    is_active = input("Masukkan is_active: ")

    # Membuat objek vendor dalam bentuk dictionary
    data = {
     "kode": kode,
     "nama": nama,
     "kontak_person": kontak_person,
     "is_active": is_active
    }
    try:
        # Membuat koneksi ke database MySQL
        db_connection = mysql.connector.connect(
            host="localhost",       # Ganti dengan host database Anda
            user="root",            # Ganti dengan username database Anda
            password="",            # Ganti dengan password database Anda
            database="db_sensor"    # Ganti dengan nama database Anda
        )

        # Membuat kursor
        cursor = db_connection.cursor()

        print(db connection)
```

Gambar 3. 22 Program *Create* Data Vendor

Gambar 3.22 menunjukkan program untuk *add* data vendor. Program tersebut menggunakan *library* 'mysql.connector'. Pada tahap pertama, *library* 'request' digunakan untuk menyambungkan dengan API untuk melakukan pemintaan Hypertext Transfer – Transfer Protocol (HTTP). *Library* 'mysl.connector' digunakan untuk menyambungkan dengan *database* MySQL. 'create vendor' berfungsi sebagai entri vendor baru dalam *database* MySQL. Program memasukan beberapa informasi seperti kode, nama, kontak *person*, dan status aktif atau tidak aktif. *Input* tersebut kemudian disimpan dalam bentuk dictionary, seperti "kode", "nama", "kontak_*person*", dan "is_active". Selanjutnya, Praktikan mencoba membuat koneksi ke *database* MySQL menggunakan *library* 'mysql.connector' dengan menyediakan informasi koneksi seperti *host, username, password*, dan nama *database*.

```
        # Menjalankan query INSERT untuk menyimpan data vendor baru
        insert_query = "INSERT INTO tabel_vendor SET kode = %s, nama = %s, kontak_person = %s, is_active = %s"
        values = (kode, nama, kontak_person, is_active)
        cursor.execute(insert_query, values)

        # Melakukan commit perubahan ke database
        db_connection.commit()

        print("Data vendor baru berhasil ditambahkan.")
    except mysql.connector.Error as error:
        print("Terjadi kesalahan dalam menambahkan data vendor:", error)
    finally:
        # Menutup koneksi ke database
        if db_connection.is_connected():
            cursor.close()
            db_connection.close()

# Memanggil fungsi untuk membuat vendor baru
create_vendor()
```

Gambar 3. 23 Sambungan Program *Create* Data Vendor

Gambar 3. 23 menunjukkan sambungan program *add* data vendor. Program ini melanjutkan dari bagian sebelumnya dengan menambahkan langkah-langkah untuk menyimpan data vendor yang di*input* ke *database*. Selanjutnya, program membentuk sebuah *query* SQL dengan format string. Data yang diterima dimasukkan sebagai parameter (%s) ke dalam *query database*. *Query* dirancang untuk melakukan operasi *insert* ke dalam tabel vendor dengan menyertakan nilai-nilai yang telah diambil dari *input*. Program menggunakan objek *kursor* untuk mengeksekusi *query insert* dengan menyertakan nilai-nilai yang telah disiapkan. Setelah eksekusi *query*, perubahan tersebut dicommit ke *database*, sehingga data baru berhasil ditambahkan ke dalam tabel. Program menangkap *exception* yang disediakan oleh *library* 'mysql_connector' dan mencetak pesan kesalahan.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\KingBib\Desktop\vendor> & C:/Users/KingBib/AppData/Local/Programs/Python/Python311
ost_vendor_final.py
Masukkan kode: TST
Masukkan nama: Testing
Masukkan kontak_person: 08572239123
Masukkan is_active: 1
<mysql.connector.connection_cext.CMySQLConnection object at 0x000001954BE0C390>
Data vendor baru berhasil ditambahkan.
PS C:\Users\KingBib\Desktop\vendor>
```

Gambar 3. 24 Hasil Program *Create* Data Vendor

Gambar 3.24 menunjukkan hasil eksekusi program *add* data vendor. Praktikan diminta untuk memasukkan informasi vendor seperti kode, nama, kontak *person*, dan status aktif atau tidak aktif. Setelah

meng*input* informasi tersebut, program membentuk dictionary dan terhubung ke *database* MySQL dengan menggunakan *library* 'mysql_connector'. Hasil koneksi ke *database* ditampilkan sebagai objek koneksi. Program kemudian menjalankan *query insert* untuk menyimpan data vendor baru ke dalam database. Setelah *query* dijalankan, program mencetak pesan bahwa data vendor baru telah berhasil ditambahkan.

Gambar 3. 25 Program *Update* Data Vendor

Gambar 3.25 menunjukkan program untuk *update* data vendor. Program tersebut menggunakan beberapa *library*, termasuk 'request', 'json', dan 'mysql.connector'. Pada tahap pertama, *library* 'request' digunakan untuk menyambungkan dengan API untuk melakukan pemintaan Hypertext Transfer – Transfer Protocol (HTTP). *Library* 'json' digunakan untuk manipulasi data dalam format json, serta 'mysl.connector' digunakan untuk menyambungkan dengan *database* MySQL. 'update_vendor' berfungsi untuk memperbarui data vendor dalam *database*. Praktikan mengawali dengan memasukkan *ID* vendor yang akan diperbarui. Selanjutnya, program mengambil *input* parameter seperti kode,

nama, kontak *person*, dan status aktif. Informasi tersebut disimpan dalam dictionary. Program kemudian membuat koneksi ke *database* MySQL dan membentuk objek *kursor*. Selanjutnya, program menjalankan *query update* dengan menggunakan format string. *Query* tersebut mengubah data vendor dengan *ID* yang seusai. Setelah itu, program mencetak pesan bahwa data vendor berhasil diperbarui.

Gambar 3. 26 Hasil Program *Update* Data Vendor

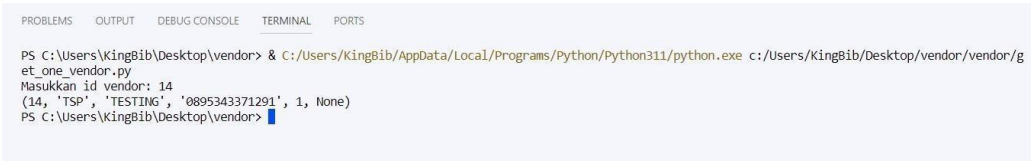
Gambar 3.26 menunjukkan hasil eksekusi program *update* data vendor. Praktikan memasukan *ID* vendor (15), kode (TSR), nama (Testing), kontak *person* (0895343371291), dan status aktif (1). Setelah eksekusi, output menunjukkan bahwa koneksi ke *database* MySQL berhasil tersambung dan data vendor berhasil diperbarui. Program menggunakan *library* mysql.*connector* untuk *database* MySQL dan menjalankan operasi *update* data vendor sesuai dengan *input*an yang diberikan.

vendor > get_one_vendor.py > select_one_vendor

```python
import mysql.connector

def select_one_vendor():
    try:
        conn = mysql.connector.connect(
            host='localhost',
            user='root',
            password='',
            database='db_sensor',
        )

        # Mengambil kode dari user input
        kode = input("Masukkan id vendor: ")

        cursor = conn.cursor()
        insert_query = "SELECT * FROM tabel_vendor WHERE id = %s"
        values = (kode,)
        cursor.execute(insert_query, values)

        # Mengambil hasil dari kueri
        result = cursor.fetchone()
        if result:
            # Jika data ditemukan
            print(result)
        else:
            # Jika data tidak ditemukan
            print("Data tidak ditemukan")

    except Exception as e:
        print(f"Terjadi kesalahan: {e}")

    finally:
        # Pastikan untuk menutup kursor dan koneksi
        cursor.close()
        conn.close()

# Memanggil fungsi untuk mengambil data vendor
select_one_vendor()
```

Gambar 3. 27 Program *Get* Data Vendor

Gambar 2.27 menunjukkan program untuk *get* data vendor. Program tersebut menggunakan *library* 'mysql.connector'. 'select_one_vendor()' berfungsi untuk mengambil data vendor berdasarkan *ID* yang dimasukkan melalui *input*. Program membuat koneksi ke *database* dengan parameter *host, user, password*, dan nama *database* yang sesuai. Selanjutnya, program mengambill *ID* vendor dari *input*, lalu mengeksekusi *query select* pada tabel_vendor dengan parameter *ID* tersebut. Jika data ditemukan, program mendapatkan hasil *query* menggunakan 'fetchone()' dan mencetak pesan bahwa data ditemukan. Jika tidak, program mencetak pesan bahwa data tidak ditemukan.



Gambar 3. 28 hasil program *get* data vendor

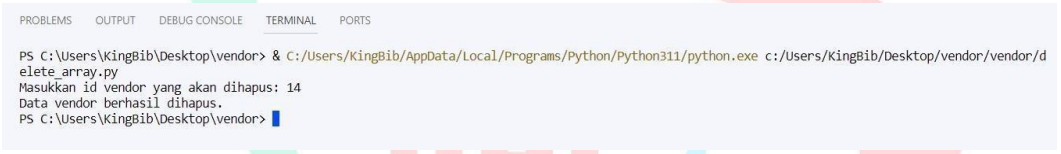
Gambar 3.28 menunjukkan hasil eksekusi program *update* data vendor. Praktikan memasukkan *ID* vendor (15), kode (TSR), nama (Testing), kontak *person* (0895343371291), dan status aktif (1). Pesan output menunjukkan bahwa koneksi ke *database* MySQL berhasil tersambung dan data vendor berhasil diperbarui. Pesan "DATA vendor baru berhasil di*update* " menandakan operasi pembaruan data berhasil.

vendor > delete_array.py > delete_vendor

```python
import requests
import mysql.connector

def delete_vendor():
    # Mengambil kode dari user input
    id = input("Masukkan id vendor yang akan dihapus: ")

    try:
        # Membuat koneksi ke database MySQL
        db_connection = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="db_sensor"
        )

        # Membuat kursor
        cursor = db_connection.cursor()

        # Menjalankan query DELETE untuk menghapus data vendor
        delete_query = "DELETE FROM tabel_vendor WHERE id = %s"
        values = (int(id),)  # Ubah ID ke tipe data int dan buat tupel
        cursor.execute(delete_query, values)

        # Melakukan commit perubahan ke database
        db_connection.commit()

        print("Data vendor berhasil dihapus.")
    except mysql.connector.Error as error:
        print("Terjadi kesalahan dalam menghapus data vendor:", error)
    finally:
        # Menutup koneksi ke database
        if db_connection.is_connected():
            cursor.close()
            db_connection.close()

# Memanggil fungsi untuk menghapus data vendor
delete_vendor()
```

Gambar 3. 29 Program *Delete* Data Vendor

Gambar 3.29 menunjukkan hasil eksekusi program *delete* data vendor. Program menggunakan modul ‘request’ dan ‘mysql.connector’ untuk menghapus data vendor dari *database* MySQL. ‘delete_vendor()’ berfungsi untuk menghapus data vendor berdasarkan *ID* yang dimasukkan melalui *input*. Program kemudian membuat koneksi ke *database* MySQL dengan parameter *host, user, password*, dan nama *database* yang sesuai. Setelah itu, program menjalankan *query delete* untuk menghapus data vendor dari tabel_vendor berdasarkan *ID* yang dimasukkan. Jika operasi *delete* berhasil, program melakukan commit perubahahan ke *database*. Program mencetak pesan “Data vendor berhasil dihapus”. Jika terjadi kesalahan dalam operasi *delete*, program mencetak pesan “Terjadi kesalahan dalam menghapus data vendor”. Blok finally memastikan penutupan *kursor* dan koneksi *database*.



Gambar 3. 30 Hasil Program *Delete* Data Vendor

Gambar 3.30 menunjukkan hasil eksekusi program *delete* data vendor. Praktikan memasukkan *ID* vendor yang akan dihapus (14). Pesan output menunjukkan bahwa koneksi ke *database* MySQL berhasil tersambung dan data vendor berhasil diperbarui. Pesan “DATA vendor berhasil dihapus ” menandakan operasi *delete* data berhasil.

3.2.4. Implementasi

Setelah tahap perancangan operasi *CRUD* untuk pemantauan kualitas udara, Praktikan melakukan operasi *CRUD* yang telah dirancang sebelumnya.

3.2.3.1. Spesifikasi Kebutuhan Perangkat lunak

Dalam operasi *CRUD* untuk pemantauan kualitas udara, berikut spesifikasi perangkat lunak yang digunakan Praktikan

dalam menunjang pengerjaan sistem ini, disajikan dalam bentuk tabel.

Tabel 3.2 Spesifikasi Kebutuhan Perangkat Lunak

No.	Nama Perangkat Lunak	Spesifikasi Perangkat Lunak
1.	*Operating System*	Windows 11
2.	*Local Web Server*	Apache 2.4
3.	*Local Development Environment*	XAMPP v3.3.0
4.	*Programming Language*	PYTHON
5.	*Database*	MySQL
8.	IDE	Visual Studio Code

3.2.3.1. Spesifikasi Kebutuhan Perangkat Keras

Adapun beberapa spesifikasi perangkat keras dalam mengembangkan sistem ini, berikut spesifikiasi yang disajikan dalam bentuk tabel.

Tabel 3.3 Spesifikasi Kebutuhan Perangkat Keras

No.	Nama Perangkat Keras	Spesifikasi Perangkat Keras
1.	Processor	Intel Core i3-6100
2.	VGA	NVIDIA Geforce 920
3.	Memory	12 GB

3.3 Kendala yang Dihadapi

Saat mengikuti program KP di PT. MARKTEL, Praktikan mengalami kesulitan karena perangkat sensor belum tiba. Oleh karena itu, diperlukan penyediaan *input* manual dalam program untuk sementara waktu. Setelah sensor tersedia, maka perlu dilakukan koneksi ulang ke server dengan menggunakan API *Postman* yang sebelumnya terhubung dengan MySQL. Di saat sensor tidak terdeteksi atau tidak dapat membaca data, Praktikan mengirim ulang data untuk melakukan pengujian apakah sensor dapat membaca data atau tidak.

3.4 Cara Mengatasi Kendala

Berikut ini beberapa cara Praktikan dalam mengatasi kendala yang dihadapi saat melaksanakan program KP di PT. MARKTEL, yaitu:

a. Praktikan berdiskusi kembali dengan mentor untuk membahas pengelompokan data yang telah ada.
b. Praktikan melakukan evaluasi secara rutin selama pengerjaan proyek *CRUD*.

3.5 Pembelajaran yang Diperoleh Dari Kerja Profesi

Selama melaksanakan program KP di PT. MARKTEL, Praktikan mendapatkan pembelajaran antara lain :

1) Praktikan diberikan pemahaman mengenai pentingnya disiplin dan menghargai waktu, karena faktor waktu memiliki dampak signifikan terhadap hasil pekerjaan yang dilakukan.
2) Melatih kerjasama tim dengan saling membantu ketika terdapat kendala saat mengerjakan proyek.
3) Praktikan belajar untuk menunjukkan keahlian dalam pemecahan masalah dengan efisien. Mereka juga diajarkan cara meningkatkan produktivitas kerja dengan cepat, serta berani memberikan pendapat atau kritik secara konstruktif selama proses diskusi atau presentasi.