

Tugas Kecil 2
Pembuatan Docker Container
II2210 - Teknologi Platform

Dipersiapkan oleh:
Asisten Lab Sistem Terdistribusi & Co.



Disusun oleh :
Nakeisha Valya Shakila
18223133

PROGRAM STUDI SISTEM DAN TEKNOLOGI INFORMASI
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JATINANGOR
2025

DAFTAR ISI

| | |
|--|-----------|
| DAFTAR ISI..... | 2 |
| BAB I : PENDAHULUAN..... | 3 |
| 1.1. Latar Belakang Masalah..... | 3 |
| 1.2. Rumusan Masalah..... | 3 |
| 1.3. Tujuan dan Manfaat..... | 3 |
| BAB II : PEMBAHASAN..... | 4 |
| 2.1. Instalasi Docker Engine..... | 4 |
| 2.2. Pembuatan Repository Menggunakan Git..... | 5 |
| 2.2.1. Instalasi Git..... | 5 |
| 2.2.2. Pembuatan Repository..... | 5 |
| 2.2.3. Cloning Repository..... | 7 |
| 2.3. Pembuatan Web Server..... | 7 |
| 2.3.1. Melengkapi Template Web Server FastAPI dan Penjasannya..... | 7 |
| 2.3.2. Pembuatan Dockerfile untuk Menjalankan Web Server..... | 12 |
| 2.3.3. Commit dan Push Terhadap Repository..... | 14 |
| 2.4. Deployment Web Server..... | 14 |
| 2.4.1. Pull Terhadap Repository..... | 14 |
| 2.4.2. Deployment Web Server..... | 14 |
| 2.5. Uji Coba..... | 16 |
| 2.5.1. Melakukan Run pada File tester.py..... | 16 |
| 2.5.2. Tampilan Web Browser..... | 16 |
| BAB III : PENUTUP..... | 18 |
| 3.1. Kesimpulan..... | 18 |
| DAFTAR PUSTAKA..... | 19 |

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Dalam pengembangan aplikasi web modern, berbagai teknik dan alat digunakan untuk meningkatkan efisiensi, keamanan, dan kemudahan pengelolaan aplikasi. Salah satu teknik yang semakin populer adalah **containerization**, yang memungkinkan aplikasi untuk berjalan dengan konsisten di berbagai lingkungan. **Docker Engine** memainkan peran penting dalam hal ini, memungkinkan pengembang untuk membuat dan mengelola **container** yang menyertakan aplikasi beserta seluruh dependensinya. Dengan menggunakan Docker, aplikasi dapat dijalankan secara terisolasi dan di-deploy dengan mudah di berbagai platform.

Selain itu, **GitHub** digunakan untuk pengelolaan kode sumber aplikasi melalui sistem **repository** yang memungkinkan pengembang untuk melakukan **commit**, **push**, dan **pull** terhadap perubahan kode. Dengan menggunakan GitHub, pengembang dapat berbagi kode secara efisien, mengelola versi aplikasi, dan memastikan bahwa kode yang paling baru selalu tersedia di server atau lingkungan produksi. Untuk menjaga keamanan dan autentikasi dalam proses **push** dan **pull**, digunakanlah **SSH key**, yang menyediakan akses aman tanpa memerlukan kata sandi setiap kali melakukan operasi git.

Pada pengembangan aplikasi web ini, digunakan **Virtual Machine (VM)** dengan sistem operasi **Ubuntu 22.04** sebagai platform tempat aplikasi dijalankan. Aplikasi ini menggunakan **FastAPI** sebagai framework untuk membangun API dengan Python. **SQLite** digunakan sebagai database ringan untuk menyimpan data aplikasi, yaitu **Message of the Day (MOTD)**, yang dapat diambil atau ditambahkan oleh pengguna melalui beberapa endpoint yang disediakan.

Aplikasi ini di-deploy dalam **Docker container**, yang memudahkan pengelolaan lingkungan aplikasi. **Dockerfile** digunakan untuk mendefinisikan cara membangun **Docker image**, sedangkan **docker-compose.yml** digunakan untuk menyederhanakan deployment aplikasi dan mengelola beberapa container. **pycache** dihasilkan oleh Python saat aplikasi dijalankan dan berisi file bytecode Python yang digunakan untuk meningkatkan performa aplikasi. Semua file dan konfigurasi ini bekerja bersama untuk memastikan aplikasi berjalan dengan lancar, efisien, dan aman.

1.2. Rumusan Masalah

Dari latar belakang yang ada, terdapat beberapa rumusan masalah yang diharapkan dapat terjawab setelah membaca laporan ini, antara lain sebagai berikut:

- Bagaimana cara mengelola lingkungan pengembangan aplikasi web dengan efisien menggunakan Docker?
- Bagaimana cara menghubungkan aplikasi FastAPI dengan database SQLite dalam sebuah lingkungan container Docker?
- Bagaimana cara menambahkan otentikasi dan pengelolaan data dengan aman di aplikasi web?
- Bagaimana cara mengelola versi kode sumber menggunakan GitHub?

1.3. Tujuan dan Manfaat

- Membangun aplikasi web dengan FastAPI dalam Docker container, menggunakan SQLite untuk penyimpanan data dan TOTP untuk otentikasi dua faktor, dikelola melalui GitHub dan di-deploy menggunakan Docker Compose.
- Penggunaan Docker mempermudah deployment, TOTP meningkatkan keamanan, SQLite menyediakan penyimpanan ringan, dan GitHub memudahkan pengelolaan kode serta kolaborasi.

BAB II

PEMBAHASAN

2.1. Instalasi Docker Engine

Docker Engine merupakan platform open-source yang digunakan untuk membangun, menjalankan, dan mengelola aplikasi dalam container. Sebelum menginstalnya, kita harus memastikan tidak ada konflik dengan paket lama atau paket yang tidak kompatibel, seperti versi Docker sebelumnya agar proses instalasi berjalan lancar.

```
nakei@UbuntuVM-II2210:~$ for pkg in docker.io docker-doc docker-compose
docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg;
done
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'docker.io' is not installed, so not removed.
0 upgraded, 0 newly installed, 0 to remove and 19 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'docker-doc' is not installed, so not removed.
0 upgraded, 0 newly installed, 0 to remove and 19 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'docker-compose' is not installed, so not removed.
0 upgraded, 0 newly installed, 0 to remove and 19 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'docker-compose-v2' is not installed, so not removed.
0 upgraded, 0 newly installed, 0 to remove and 19 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'podman-docker' is not installed, so not removed.
0 upgraded, 0 newly installed, 0 to remove and 19 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'containerd' is not installed, so not removed.
0 upgraded, 0 newly installed, 0 to remove and 19 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'runc' is not installed, so not removed.
0 upgraded, 0 newly installed, 0 to remove and 19 not upgraded.
```

Gambar 2.1 Tampilan Tidak Ada Conflict pada Docker Engine

Setelah, itu kita bisa langsung melakukan instalasi dengan menambahkan GPG Key Docker bertujuan untuk memastikan keaslian dan keamanan paket yang diunduh

```
nakei@UbuntuVM-II2210:~$ sudo apt-get update
```

```
nakei@UbuntuVM-II2210:~$ sudo apt-get install ca-certificates curl
```

```
nakei@UbuntuVM-II2210:~$ sudo install -m 0755 -d /etc/apt/keyrings
```

```
nakei@UbuntuVM-II2210:~$ sudo curl -fsSL
https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
```

```
nakei@UbuntuVM-II2210:~$ sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Secara default, repository resmi Ubuntu tidak menyertakan versi terbaru Docker. Oleh karena itu, kita perlu menambahkan repository resmi Docker ke *apt sources* agar dapat memperoleh pembaruan langsung dari sumber resminya.

```
nakei@UbuntuVM-II2210:~$ echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" |
\
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
# Memperbarui Daftar Paket
nakei@UbuntuVM-II2210:~$ sudo apt-get update
```

Untuk memastikan bahwa Docker terinstal dengan benar, kita perlu melakukan verifikasi

```
nakei@UbuntuVM-II2210:~$ sudo systemctl status docker
```

```
nakei@UbuntuVM-II2210:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2025-03-30 05:38:07 UTC; 2min 7s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Main PID: 67043 (dockerd)
      Tasks: 9
     Memory: 36.1M
        CPU: 408ms
    CGroup: /system.slice/docker.service
            └─67043 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Gambar 2.2 Tampilan Docker Telah Aktif

Saat ini, hanya user yang memiliki akses **sudo** yang bisa menjalankan Docker. Untuk memungkinkan user **sister** menjalankan Docker tanpa **sudo**, tambahkan user tersebut ke dalam grup **docker**

```
nakei@UbuntuVM-II2210:~$ sudo usermod -aG docker sister
```

Untuk memastikan bahwa Docker berfungsi dengan baik, kita dapat mengujinya menggunakan user **sister**

```
sister@UbuntuVM-II2210:~$ sudo docker run hello-world
```

```
sister@UbuntuVM-II2210:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Gambar 2.3 Tampilan Docker Telah Berfungsi pada User sister

2.2. Pembuatan Repository Menggunakan Git

Setelah proses instalasi pada Docker telah selesai, langkah selanjutnya adalah memanfaatkan repository Git sebagai tempat penyimpanan dan pengelolaan kode sumber secara terpusat

2.2.1. Instalasi Git

Lakukan instalasi git dan pastikan bahwa git sudah terinstal

```
nakei@UbuntuVM-II2210:~$ sudo apt-get install git
```

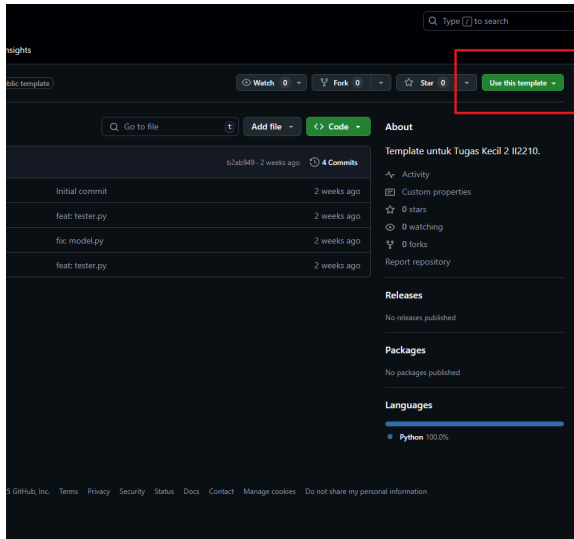
```
nakei@UbuntuVM-II2210:~$ git --version
```

```
nakei@UbuntuVM-II2210:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.34.1-1ubuntu1.12).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 19 not upgraded.
nakei@UbuntuVM-II2210:~$ git --version
git version 2.34.1
```

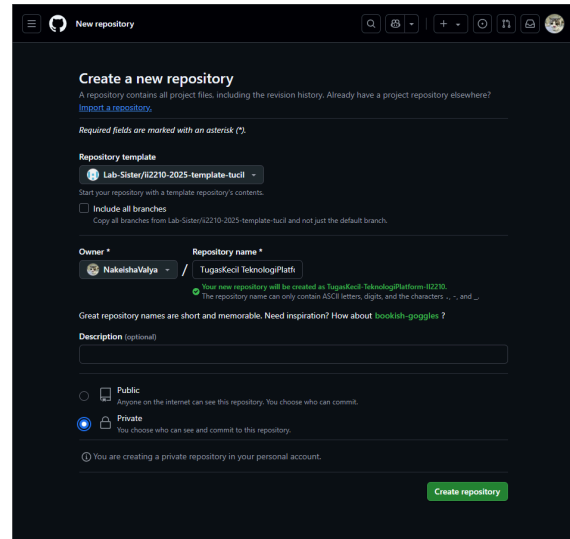
Gambar 2.4 Tampilan Git Berhasil Terinstall

2.2.2. Pembuatan Repository

Selanjutnya, kita akan membuat repository berdasarkan template yang telah diberikan dan memastikan bahwa repository tersebut bersifat **private**. Pertama, buka tautan [template](#), lalu klik "Use this template" di pojok kanan atas. Setelah itu, isi nama repository, ubah pengaturannya menjadi **Private**, dan klik "Create Repository" setelah selesai.



Gambar 2.5 Tampilan Template Repository



Gambar 2.6 Tampilan Pengaturan Repository

Setelah repository berhasil dibuat, kita akan menambahkan **Deploy Key** agar VM dapat meng-clone repository private dari template GitHub dengan aman tanpa memerlukan autentikasi manual.

```
# Membuat SSH
nakei@UbuntuVM-II2210:~$ ssh-keygen -t rsa -b 4096 -C "<Komentar pada kunci>"

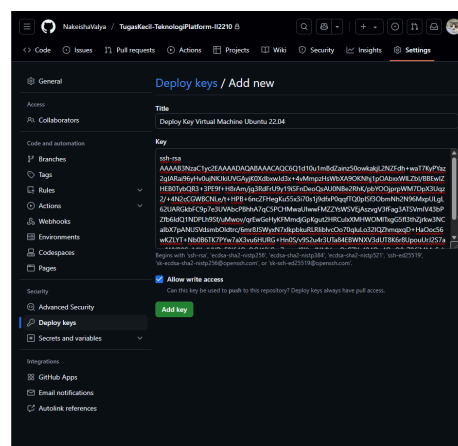
# Menyalin Public Key
nakei@UbuntuVM-II2210:~$ cat ~/.ssh/id_rsa.pub

nakei@UbuntuVM-II2210:~$ ssh-keygen -t rsa -b 4096 -C "Deploy-Key"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/nakei/.ssh/id_rsa):
/home/nakei/.ssh/id_rsa already exists.

nakei@UbuntuVM-II2210:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQCAQ6Qid10ulmBdZainz50owkakjL2NZFdh+waT7KyPYaz2gIARai96yHr0UjNKKJk1UVGAyJK0XdbxwJd3x+4vMmpzHsWbXA9OKNhjlpQbxxxW1L2bI/BBEw
cCGWBCNLe/t+HPB+6ncZFHeGKu55x3170e1j9dFkP0qgFT00pISf3QbmeN2N96MxpDL6L62UARgkbFC9p7e3UVAbcPBhha7qc5PCHMwaUIwFMZZYsWSVEJAazvgV3fFag3ATSvmIV43bPZfb6IdQINDPLH
c/6mr8J5WyxN7x1kpbkuRLRIiblvOo70qIuLo32IQzhmqxqD+HaOoc56wKZLYT+Nb0B6TK7PYw7aX3vu6HURGHn05/v9S2u4r3UTa84EBWNXV3dUT8K6r8UpouUrJ2S7ao/WXB9SgMKc/HYDzF0I64QaQ
8QpAxsWQm5Q== vaelya@UbuntuVM-II2210
```

Gambar 2.7 Tampilan Public Key Berhasil Dibuat

Untuk menambahkannya sebagai Deploy Key di GitHub, buka repository, lalu masuk ke **Settings > Deploy keys**. Selanjutnya, isi judul, masukkan **Public Key** yang telah dibuat, **checklist Allow Write Access**, dan klik **Add Key**

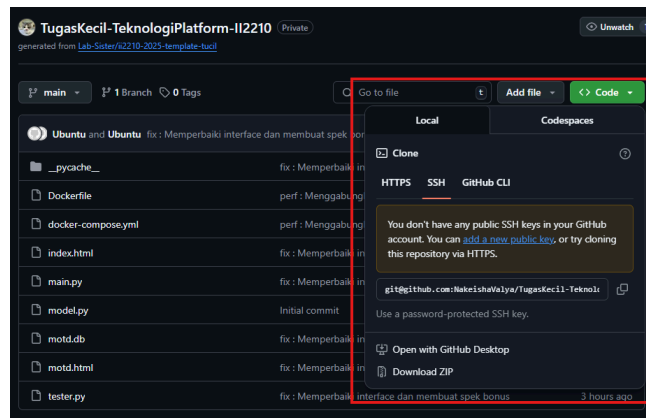


Gambar 2.8 Tampilan Pengaturan Deploy keys Pada Repository GitHub

Lakukan uji koneksi ke GitHub untuk memastikan bahwa kunci SSH telah dibuat dan berfungsi dengan baik untuk autentikasi GitHub

```
# Membuat SSH
nakei@UbuntuVM-II2210:~$ ssh -T <git@github.com>
```

Untuk mengautentikasikan GitHub pada VM, buka repository pada web GitHub, setelah itu klik **Code > Codespace > SSH > Copy Link SSH**



Gambar 2.9 Tampilan Link untuk Autentikasi pada GitHub

```
nakei@UbuntuVM-II2210:~$ ssh -T git@github.com
The authenticity of host 'github.com (20.205.243.166)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6TuJhbpZisF/zLDA0zFMSvHdKriUvCoQ.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Hi NakeishaValya/TugasKecil-TeknologiPlatform-II2210! You've successfully authenticated, but GitHub does not provide shell access.
```

Gambar 2.10 Tampilan Autentikasi Berhasil

2.2.3. Cloning Repository

Selanjutnya, kita akan melakukan cloning repository untuk menyalin seluruh isi repository GitHub ke dalam VM. Dengan cloning, kita dapat mengakses, mengedit, dan mengelola file dalam repository tanpa perlu membuatnya ulang dari awal.

```
nakei@UbuntuVM-II2210:~$ git clone
git@github.com:NakeishaValya/TugasKecil-TeknologiPlatform-II2210.git

# menampilkan daftar file dan folder pada direktori
nakei@UbuntuVM-II2210:~$ ls TugasKecil-TeknologiPlatform-II2210<

nakei@UbuntuVM-II2210:~$ ls TugasKecil-TeknologiPlatform-II2210
Dockerfile main.py model.py tester.py
```

Gambar 2.11 Tampilan Cloning Repository Berhasil Dilakukan

2.3. Pembuatan Web Server

Selanjutnya, dilakukan pembuatan web sederhana di dalam Virtual Machine (VM) dengan memanfaatkan container dan Docker. Web server yang di-build dengan framework **FastAPI** untuk pengembangan aplikasi web berbasis Python. Proses ini tetap bergantung pada **Docker Engine** sebagai inti dari sistem.

2.3.1. Melengkapi Template Web Server FastAPI dan Penjelasannya

Dalam tahap pembuatan web server, pertama-tama kita perlu untuk melengkapi beberapa file yang sudah di-clone pada VM dengan command dibawah ini dan pastikan direktori/folder sudah sesuai

```
# Masuk kedalam folder
nakei@UbuntuVM-II2210:~$ cd TugasKecil-TeknologiPlatform-II2210

# Mengubah/menambahkan file baru
nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ nano <nama file>
```

Apabila mengubah/menambahkan file telah selesai, maka tekan **Ctrl X > Y > Enter** agar perubahan dapat tersimpan. Karena index.html dan motd.html akan dipanggil pada main.py, maka file tersebut perlu dibuat di direktori yang sama dengan main.py. Untuk menambahkan file dapat menggunakan command yang sama seperti untuk melengkapi beberapa file. Berikut merupakan file main.py secara lengkap

```
import secrets
import base64
import pyotp
from fastapi import FastAPI, Depends, HTTPException, status, Request
from fastapi.security import HTTPBasic, HTTPBasicCredentials
from fastapi.responses import HTMLResponse, FileResponse
from fastapi.templating import Jinja2Templates
from sqlmodel import create_engine, Session, SQLModel
from typing import Annotated
from model import MOTD, MOTDBase
from sqlalchemy.sql import func

# SQLite Database
sqlite_file_name = "motd.db"
sqlite_url = f"sqlite:///{"sqlite_file_name}"
connect_args = {"check_same_thread": False}
engine = create_engine(sqlite_url, connect_args=connect_args)

def create_db_and_tables():
    SQLModel.metadata.create_all(engine)

def get_session():
    with Session(engine) as session:
        yield session

SessionDep = Annotated[Session, Depends(get_session)]

# FastAPI
app = FastAPI(docs_url=None, redoc_url=None)
templates = Jinja2Templates(directory=".")
security = HTTPBasic()

# Users
users = {"sister": "ii2210_sister_semangatTucil", "vaelya": "ii2210_vaelya"}

@app.get("/")
async def root():
    return FileResponse("index.html")

@app.get("/motd", response_class=HTMLResponse)
async def get_motd_page(request: Request, session: SessionDep):
    motd = session.query(MOTD).order_by(func.random()).first()
    if motd:
        return templates.TemplateResponse("motd.html", {
            "request": request,
            "motd": motd.motd,
            "creator": motd.creator,
            "created_at": motd.created_at
        })
    raise HTTPException(status_code=404, detail="No message found.")

@app.get("/api/motd")
async def get_motd_json(session: SessionDep):
    motd = session.query(MOTD).order_by(func.random()).first()
    if motd:
        return {
            "motd": motd.motd,
            "creator": motd.creator,
            "created_at": motd.created_at
        }
    raise HTTPException(status_code=404, detail="No message found.")
```



```
@app.post("/motd")
async def post_motd(message: MOTDBase, session: SessionDep, credentials:
Annotated[HTTPBasicCredentials, Depends(security)]):
    current_password_bytes = credentials.password.encode("utf8")

    if credentials.username in users:
        s =
base64.b32encode(users.get(credentials.username).encode("utf-8")).decode("utf-8"
)
        totp = pyotp.TOTP(s=s, digest="SHA256", digits=8)
        if secrets.compare_digest(current_password_bytes,
totp.now().encode("utf8")):
            new_motd = MOTD(motd=message.motd, creator=credentials.username)
            session.add(new_motd)
            session.commit()
            session.refresh(new_motd)
            return {"message": "MOTD created successfully."}

        raise HTTPException(status_code=status.HTTP_401_UNAUTHORIZED,
detail="Invalid userid or password.")

if __name__ == "__main__":
    import uvicorn
    create_db_and_tables()
    uvicorn.run("main:app", host="0.0.0.0", port=17787, reload=True)
```

Endpoint merupakan suatu URL di web server yang mengatur permintaan (*request*) dan memberikan respons (*response*). Di FastAPI, endpoint dikendalikan oleh decorator yang mengikat fungsi tertentu ke URL dan metode HTTP. Berikut merupakan endpoint yang dicantumkan pada kode tersebut

Tabel 2.1 Penjelasan Masing-Masing Endpoint pada main.py

| Penulisan Pada Kode | Metode HTTP | URL | Fungsi | | |
|-----------------------|-------------|-----------|--|---|---|
| | | | Keterangan | Penjelasan | Respons |
| @app.get("/") | GET | / | Mengembalikan file index.html sebagai respons | Ketika ada permintaan GET ke root URL (/), server akan mengirimkan file HTML kepada pengguna | File HTML (index..html) |
| @app.get("/motd") | GET | /motd | Mengambil Message of The Day (MOTD) dari database (secara acak) dan menampilkannya dalam bentuk HTML | Endpoint ini akan mencari pesan acak dari database SQLite dan mengembalikannya dalam halaman HTML yang di-random menggunakan motd.html | Halaman HTML yang menampilkan pesan acak (termasuk pembuat dan waktu pembuatan) |
| @app.get("/api/motd") | GET | /api/motd | Mengambil Message of The Day (MOTD) dalam format JSON | Endpoint ini memberikan data dalam format JSON, yang cocok digunakan oleh frontend seperti JavaScript untuk menampilkan data secara dinamis | JSON yang berisi pesan, pembuat, dan waktu pembuatan dari pesan tersebut |
| @app.post | POST | /motd | Mengizinkan | Endpoint ini menggunakan | Setelah |

| | | | | | |
|-----------|--|--|---|--|---|
| ("/motd") | | | pengguna untuk menambahkan Message of the Day (MOTD) baru ke database | metode POST untuk menerima data (pesan) dari pengguna. Namun, sebelum menyimpan, pengguna harus terotentikasi dengan memasukkan username dan password yang valid | berhasil, server mengirimkan pesan bahwa MOTD berhasil dibuat |
|-----------|--|--|---|--|---|

Perlu diperhatikan bahwa endpoint baru `@app.get("/api/motd")` ditambahkan untuk memisahkan fungsionalitas antara menampilkan halaman HTML dan menyediakan data dalam format JSON. Endpoint `/motd` menampilkan halaman HTML, sementara `/api/motd` menyediakan data JSON yang dapat diproses oleh JavaScript di `index.html` tanpa memuat ulang halaman. Berikut adalah penjelasan tentang cara kerja kode pada `main.py`

- Penjelasan endpoint POST /motd

Dalam kode tersebut terdapat endpoint `@app.post("/motd")`, metode otentikasi yang digunakan melibatkan dua langkah utama untuk memastikan keamanan, yaitu :

1. Verifikasi Username dan Password

Pengguna mengirimkan username dan password melalui header HTTP menggunakan Basic Authentication. Server memverifikasi kombinasi username dan password sesuai dengan data di sistem.

2. Verifikasi TOTP (Time-based One-Time Password)

Setelah username dan password berhasil diverifikasi, pengguna juga harus mengirimkan TOTP. TOTP adalah kode sekali pakai yang dihasilkan dari shared secret dan waktu saat ini. Pengguna menghasilkan TOTP menggunakan aplikasi seperti Google Authenticator, yang kemudian dibandingkan dengan yang dihitung oleh server. Jika keduanya cocok, otentikasi berhasil.

Alur Kerja

1. Server pertama-tama memeriksa apakah username dan password yang dikirim valid.
2. Kemudian, server memverifikasi apakah TOTP yang diberikan cocok dengan yang dihitung menggunakan shared secret.
3. Jika kedua langkah ini berhasil, pengguna diberikan izin untuk menambah pesan (MOTD) ke dalam database.

Dengan menggunakan dua metode otentikasi ini, kita memastikan bahwa hanya pengguna yang valid dan terotorisasi yang dapat mengakses dan mengubah data dalam sistem.

- Penjelasan SQLite



Gambar 2.12 Logo SQLite

SQLite merupakan sistem manajemen basis data relasional ringan yang berbasis file, tanpa memerlukan server terpisah. Digunakan dalam aplikasi dengan kebutuhan penyimpanan data kecil hingga menengah, SQLite dalam tugas ini berfungsi untuk menyimpan dan mengelola data Message of the Day (MOTD). Aplikasi ini menggunakan SQLite bersama SQLAlchemy dan

SQLModel untuk menyimpan dan mengambil data secara efisien. Berikut adalah cara kerja SQLite:

1. Inisialisasi Database SQLite

Pada bagian awal kode, database SQLite diinisialisasi dengan menggunakan **SQLModel** dari **sqlmodel**. Database disimpan dalam file bernama **motd.db**. Berikut adalah konfigurasi inisialisasi database pada file **main.py**

```
# Database disimpan dalam file lokal bernama motd.db
sqlite_file_name = "motd.db"
sqlite_url = f"sqlite:/// {sqlite_file_name}"

# Mengatasi masalah apabila SQLite tidak bisa digunakan dalam beberapa thread secara bersamaan
connect_args = {"check_same_thread": False}

# Fungsi untuk membuat objek engine yang menghubungkan aplikasi dengan database SQLite
engine = create_engine(sqlite_url, connect_args=connect_args)
```

2. Membuat Tabel di Database

Di awal file **main.py**, ada fungsi untuk membuat tabel yang diperlukan dalam database jika tabel tersebut belum ada.

```
def create_db_and_tables():
    SQLModel.metadata.create_all(engine)
```

Fungsi ini memanggil **SQLModel.metadata.create_all(engine)** untuk memastikan bahwa tabel-tabel yang didefinisikan di dalam model (seperti **MOTD**) dibuat di dalam database. Jika tabel sudah ada, perintah ini tidak akan membuatnya lagi.

3. Mengambil Koneksi ke Database

Koneksi ke database dilakukan melalui fungsi **get_session()**, yang menggunakan **Session** dari **sqlmodel**

```
def get_session():
    with Session(engine) as session:
        yield session
```

Fungsi ini memastikan bahwa setiap permintaan untuk berinteraksi dengan database dilakukan dalam sesi yang terpisah:

4. Menyimpan dan Mengambil Data

Untuk mengambil atau menyimpan data, aplikasi menggunakan query SQLAlchemy melalui objek **session**

> Mengambil Data (Message of the Day - MOTD)

Pada endpoint **GET /motd** (atau **/api/motd**), aplikasi mengambil pesan acak dari tabel **MOTD** menggunakan query **order_by(func.random())**

```
motd = session.query(MOTD).order_by(func.random()).first()
```

Jika data ditemukan, server mengembalikan data tersebut dalam format HTML atau JSON, tergantung pada endpoint yang diminta.

> Menyimpan Data (POST /motd)

Pada endpoint **POST /motd**, aplikasi menambahkan pesan baru ke dalam database menggunakan perintah **session.add(new_motd)** dan **session.commit()**

```
new_motd = MOTD(motd=message.motd, creator=credentials.username)
session.add(new_motd)
session.commit()
```

Setelah itu, objek **new_motd** di-refresh agar mendapatkan data terbaru yang disimpan di database

5. Penggunaan SQLAlchemy dan SQLModel

- > **SQLAlchemy** digunakan untuk melakukan query ke database SQLite (seperti `session.query(MOTD)`)
- > **SQLModel** digunakan untuk mendefinisikan tabel (MOTD) dan struktur data dalam aplikasi. **MOTD** adalah model yang mendefinisikan tabel yang menyimpan pesan (motd), pembuat pesan (creator), dan waktu pembuatan (created_at).

2.3.2. Pembuatan Dockerfile untuk Menjalankan Web Server

Lakukan instalasi FastAPI dan uvicorn dalam VM untuk memastikan bahwa kode berjalan sesuai dengan ketentuan

```
nakei@UbuntuVM-II2210:~$ pip3 install fastapi

nakei@UbuntuVM-II2210:~$ python3 -c "import fastapi; print(fastapi.__version__)"
0.113.0
nakei@UbuntuVM-II2210:~$ uvicorn --version
Running uvicorn 0.23.2 with CPython 3.10.12 on Linux
```

Gambar 2.13 Tampilan Instalasi Berhasil Dilakukan dengan Mengecek Versi Masing-Masing

Apabila instalasi berhasil dilakukan, selanjutnya kita perlu untuk melengkapi Dockerfile yang masih kosong,

```
FROM python:3.10.12-slim

WORKDIR /app

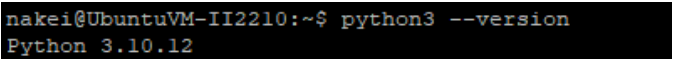
COPY . .

RUN pip install --no-cache-dir \
    fastapi>=0.0.7 \
    uvicorn>=0.23.2 \
    sqlmodel>=0.0.8 \
    pydantic>=2.3.0 \
    jinja2>=3.1.2 \
    pyotp>=2.8.0 \
    python-multipart>=0.0.6

EXPOSE 17787

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "17787"]
```

Tabel 2.2 Penjelasan Masing-Masing Baris pada Dockerfile

| Kode | Penjelasan |
|--------------------------|--|
| FROM python:3.10.12-slim | Instruksi untuk menentukan base image yang digunakan untuk membuat container  <i>Gambar 2.14 Tampilan Versi Python</i> |
| WORKDIR /app | Instruksi untuk Menetapkan directory yang bekerja di dalam container. Semua perintah selanjutnya akan dijalankan di dalam direktori /app. Jika direktori tersebut belum ada, Docker akan membuatnya secara otomatis |
| COPY . . | Instruksi untuk Menyalin file atau direktori dari sistem host ke dalam container. Dalam hal ini, perintah ini menyalin semua file dan folder (yang ada pada direktori tempat Dockerfile berada) ke dalam direktori kerja (/app) di container |

| | |
|--|--|
| <pre> RUN pip install --no-cache-dir \ `fastapi>=0.0.7` \ `uvicorn>=0.23.2` \ `sqlmodel>=0.0.8` \ `pydantic>=2.3.0` \ `jinja2>=3.1.2` \ `pyotp>=2.8.0` \ `python-multipart>=0.0.6` </pre> | <p>Instruksi ini digunakan untuk menginstal beberapa paket Python yang diperlukan untuk aplikasi, seperti:</p> <ul style="list-style-type: none"> - fastapi untuk framework web. - uvicorn sebagai server ASGI untuk menjalankan aplikasi FastAPI. - sqlmodel untuk bekerja dengan model database. - pydantic untuk validasi data. - jinja2 untuk templating. - pyotp untuk One-Time Password (OTP). - python-multipart untuk menangani upload file multipart. <p>Flag <code>--no-cache-dir</code> digunakan agar pip tidak menyimpan cache selama instalasi, yang membuat ukuran image lebih kecil.</p> |
| Expose 17787 | <p>Instruksi untuk memberi tahu Docker bahwa container ini akan mendengarkan pada port 17787. Ini tidak membuka port secara otomatis, tetapi memberi informasi kepada Docker dan pengguna lainnya tentang port yang akan digunakan untuk berkomunikasi dengan aplikasi di dalam container</p> |
| <pre> CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "17787"] </pre> | <p>Instruksi yang dieksekusi saat container dijalankan. Di sini, perintah ini menjalankan server Uvicorn dengan aplikasi FastAPI yang ada di file <code>main.py</code> (di dalam objek <code>app</code>)</p> <ul style="list-style-type: none"> - <code>main:app</code> menunjuk ke objek <code>app</code> yang ada di file <code>main.py</code> - <code>--host 0.0.0.0</code> berarti aplikasi akan mendengarkan di semua alamat IP, agar bisa diakses dari luar container - <code>--port 17787</code> menetapkan port yang digunakan oleh server untuk menerima koneksi, sesuai dengan yang dibuka dengan EXPOSE |

Setelah Dockerfile dibuat, selanjutnya kita akan mem-*build* Docker image dengan tujuan membungkus aplikasi beserta dependensinya, agar dapat dijalankan dengan mudah dan konsisten di berbagai lingkungan.

```

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker build -t
<Nama Image> .

```

Untuk memastikan bahwa Docker Image telah dibuat, kita bisa mengeceknya dengan command dibawah ini

```

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker images

```

```

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
tugaskecil-teknologiplatform-ii2210_app    latest         8d35516c227f   About an hour ago   189MB
new-docker-ii2210      latest         7632c74d87d5   2 hours ago       189MB

```

Gambar 2.15 Tampilan Docker Image yang Tersedia pada VM

Dalam memastikan bahwa kontainer dapat berjalan sesuai dengan port yang diinginkan, maka kita bisa menjalankannya dengan command dibawah ini

```

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker run -p
<port>:<port> <Nama Image>

```

Perlu diperhatikan bahwa port yang digunakan yaitu **17787** dan akan terjadi perubahan output ketika kita mengakses **http://<IP Publik VM>:17787** , hal tersebut menunjukkan bahwa kontainer sudah berjalan dengan baik pada port yang sesuai

```

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker run -p 17787:17787 new-docker-ii2210
INFO:      Started server process [1]
INFO:      Waiting for application startup.
INFO:      Application startup complete.
INFO:      Uvicorn running on http://0.0.0.0:17787 (Press CTRL+C to quit)

```

Gambar 2.16 Tampilan Output Kontainer Berjalan Sebelum Mengakses Web Browser

```
nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker run -p 17787:17787 new-docker-ii2210
INFO: Started server process [1]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:17787 (Press CTRL+C to quit)
INFO: 36.72.204.140:4532 - "GET / HTTP/1.1" 200 OK
```

Gambar 2.17 Tampilan Output Kontainer Berjalan Setelah Mengakses Web Browser

2.3.3. Commit dan Push Terhadap Repository

Setelah seluruh perubahan dilakukan pada direktori (menambah dan mengubah file) di VM, perubahan tersebut tidak akan otomatis berubah pada GitHub. Untuk memperbarui GitHub, langkah selanjutnya adalah melakukan Commit dan Push

```
nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ git add .

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ git commit -m
"<konvensi> : <Pesan>"

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ git push origin
main
```

Setelah menggunakan git add . untuk menambahkan perubahan, commit mencatat perubahan tersebut di repository lokal, sementara push akan mengirimkan perubahan ke GitHub sehingga file pada repository GitHub akan ter-*update*

```
nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ git add .
nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ git commit -m "feat : Menambahkan file index.html sebaga
1 file yang akan dipanggil di main.py"
[main f146afe] feat : Menambahkan file index.html sebagai file yang akan dipanggil di main.py
Committer: Ubuntu <nakei@UbuntuVM-II2210.xtg0bi51vgweibvniakxs3gah.jktx.internal.cloudapp.net>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 187 insertions(+)
Create mode 100644 index.html
nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 2.07 KiB | 2.07 MiB/s, done.
Total 3 (delta 1), reused 1 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:NakeishaValya/TugasKecil-TeknologiPlatform-II2210.git
 1b0b8b9..f146afe  main -> main
```

Gambar 2.18 Tampilan Commit dan Push pada VM

2.4. Deployment Web Server

2.4.1. Pull Terhadap Repository

Setelah perubahan dilakukan, tentunya kita perlu melakukan pull untuk menarik dan menggabungkan perubahan terbaru dari repository GitHub ke dalam repository/direktori VM.

```
nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ git pull origin
main

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ git pull origin main
From github.com:NakeishaValya/TugasKecil-TeknologiPlatform-II2210
* branch      main      -> FETCH_HEAD
Already up to date.
```

Gambar 2.19 Tampilan Pull pada VM

2.4.2. Deployment Web Server

Deployment web merupakan suatu proses untuk menjalankan aplikasi web di server supaya bisa diakses lewat internet, misalnya browser. Di dalam server pada VM, kita menggunakan Docker untuk membungkus aplikasi agar mudah dijalankan. Kemudian, **Docker Compose** dibuat untuk membantu kita menjalankan beberapa bagian aplikasi sekaligus (web dan database) hanya dengan satu file konfigurasi, berikut file docker-compose.yml

```
version: '3.9'
```

```

services:
  fastapi:
    build: .
    ports:
      - "17787:17787"
    container_name: fastapi_app
    command: >
      sh -c "python3 -c 'from main import create_db_and_tables;
      create_db_and_tables()' &&
      uvicorn main:app --host 0.0.0.0 --port 17787"
    volumes:
      - ../app
    restart: always

```

Setelah file tersebut berhasil dibuat, selanjutnya kita bisa menjalankan Docker Compose

```

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker-compose up
-d --build

```

Namun, jika terjadi error karena port yang digunakan dalam file konfigurasi sudah dipakai oleh proses lain, kita perlu menghentikan proses yang menggunakan port tersebut agar Docker Compose dapat dijalankan. Error ini biasanya terjadi karena port tersebut sudah digunakan saat menjalankan container yang sedang berjalan.

```

# Command untuk menampilkan detail Docker termasuk Docker ID
nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker ps

```

```

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker stop
<Container ID>

```

```

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker rm
<Container ID>

```

```

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker-compose up -d --build
Building fastapi
[+] Building 1.7s (9/9) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 589B                             0.0s
=> [internal] load metadata for docker.io/library/python:3.10.12-slim 1.4s
=> [internal] load .dockerignore                                 0.0s
=> => transferring context: 2B                                    0.0s
=> [1/4] FROM docker.io/library/python:3.10.12-slim@sha256:bd440b214e4d7deddc0a94de23a3d97d28df 0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 4.72kB                                0.0s
=> CACHED [2/4] WORKDIR /app                                    0.0s
=> CACHED [3/4] COPY . .                                        0.0s
=> CACHED [4/4] RUN pip install --no-cache-dir -r requirements.txt 0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> writing image sha256:3d55ac489e62cde97bd62bf491e24fae9869ba243ce12f1a10be58dca9876350 0.0s
=> naming to docker.io/library/tugaskecil-teknologiplatform-ii2210-fastapi 0.0s
Starting fastapi_app ... error
ERROR: for fastapi_app Cannot start service fastapi: failed to set up container networking: driver failed programming external connectivity on endpoint fastapi_app (2118fee4f1e1142c27580df1f8869eca6863d28c97d2bbe2b8373a): Bind for 0.0.0.0:17787 failed: port is already allocated
ERROR: for fastapi Cannot start service fastapi: failed to set up container networking: driver failed programming external connectivity on endpoint fastapi_app (2118fee4f1e1142c27580df1f8869eca68a9cc28c97d2bbe2b8373a): Bind for 0.0.0.0:17787 failed: port is already allocated
ERROR: Encountered errors while bringing up the project.

```

Gambar 2.20 Tampilan Docker Compose Gagal Dijalankan

```

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker-compose up -d --build
Building fastapi
[+] Building 1.1s (9/9) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 589B                             0.0s
=> [internal] load metadata for docker.io/library/python:3.10.12-slim 0.9s
=> [internal] load .dockerignore                                 0.0s
=> => transferring context: 2B                                    0.0s
=> [1/4] FROM docker.io/library/python:3.10.12-slim@sha256:bd440b214e4d7deddc0a94de23a3d97d28df 0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 4.72kB                                0.0s
=> CACHED [2/4] WORKDIR /app                                    0.0s
=> CACHED [3/4] COPY . .                                        0.0s
=> CACHED [4/4] RUN pip install --no-cache-dir -r requirements.txt 0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> writing image sha256:3d55ac489e62cde97bd62bf491e24fae9869ba243ce12f1a10be58dca9876350 0.0s
=> naming to docker.io/library/tugaskecil-teknologiplatform-ii2210-fastapi 0.0s
Starting fastapi_app ... done

```

Gambar 2.21 Tampilan Docker Compose Berhasil Dijalankan

Jika terjadi perubahan pada file-file dalam directory selama proses pembuatan web server, maka perlu dilakukan *rebuild* dengan menjalankan proses deployment ulang

```

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker-compose up
--build

```


Sebagai alternatif, bisa melakukan rebuild langsung pada container-nya hanya saja memerlukan beberapa command yang memakan waktu cukup lama

```
nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker build -t <Nama Image> .

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker stop <Nama Container>

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ docker rm <Nama Container>
```

2.5. Uji Coba

Sebelum melakukan uji coba di web browser, perlu diketahui bahwa file **tester.py** harus dijalankan terlebih dahulu agar web server dapat menampilkan pesan yang ditulis di dalamnya. Setiap kali **tester.py** dijalankan, pesan tersebut akan disimpan ke dalam database. Dengan demikian, terdapat baris dalam **tester.py** ang, ketika dijalankan, akan menyimpan pesan tersebut, dan output yang dihasilkan dapat dilihat pada *Gambar 2.23*

```
nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ messages = [
    "<Pesan 1>", "<Pesan 2>",
]
```

Pesan yang sudah berhasil di-run maka pesan tersebut akan masuk kedalam database dan muncul secara *random* pada web browser. Semua data yang masuk ke dalam database dapat dilihat dengan membuka sqlite3 dan menjalankan perintah dibawah ini

```
nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ sqlite3 motd.db
```

Setelah itu, terminal akan otomatis masuk ke dalam SQLite. Kali ini, saya hanya akan menampilkan seluruh tabel dan melihat isi dari tabel tersebut

```
# Menampilkan seluruh tabel pada database
sqlite> .tables

# Menampilkan isi data pada tabel yang dipilih
sqlite> SELECT * FROM <nama tabel>
```

Database tersebut sebenarnya masih bisa dijalankan ataupun diubah dengan perintah lainnya, selengkapnya dapat dilihat di situs resmi SQLite [disini](#)

```
sqlite> .tables
motd
sqlite> SELECT * FROM motd;
Ada Gula Ada Semut, Udah la Ga Mut|6|vaelya|2025-04-08 05:56:09.392340
Mau Tau Fakta Menarik? Ya Faktanya Kamu Menarik|7|vaelya|2025-04-08 05:56:09.412305
Padahal Orang Kek Aku Gaada Di Keranjang Kuning|8|vaelya|2025-04-08 05:56:09.428267
Hidup Selalu Kegocek Sama Ekspetasi Sendiri|9|vaelya|2025-04-08 05:56:09.444070
HTS - Harus Tetap Shalat|15|vaelya|2025-04-08 05:57:30.348823
Idup Cuma Sekali, Kacaunya Tiap Hari|16|vaelya|2025-04-08 05:57:30.368169
Di Read Doang Emangnyh Ak Novel|17|vaelya|2025-04-08 05:57:30.386708
Uhhina Tidak Tumbang, Dipuji Tidak Terbang, DiTransfer Duit Makasih Bang|18|vaelya|2025-04-08 05:57:30.402230
Semangat UTS-Nya YaaaaaD|< - < )DD|19|vaelya|2025-04-08 19:19:51.841043
```

Gambar 2.22 Tampilan Tabel dan Isi Tabel pada Database motd.db

2.5.1. Melakukan Run pada File tester.py

Untuk melakukan run di komputer lokal, kali ini saya menggunakan terminal di Ubuntu yang sudah terinstall python

```
nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ python3 tester.py

nakei@UbuntuVM-II2210:~/TugasKecil-TeknologiPlatform-II2210$ python3 tester.py
Status: 200
Response: {"message":"MOTD created successfully."}
```

Gambar 2.23 Tampilan File Berhasil Dijalankan

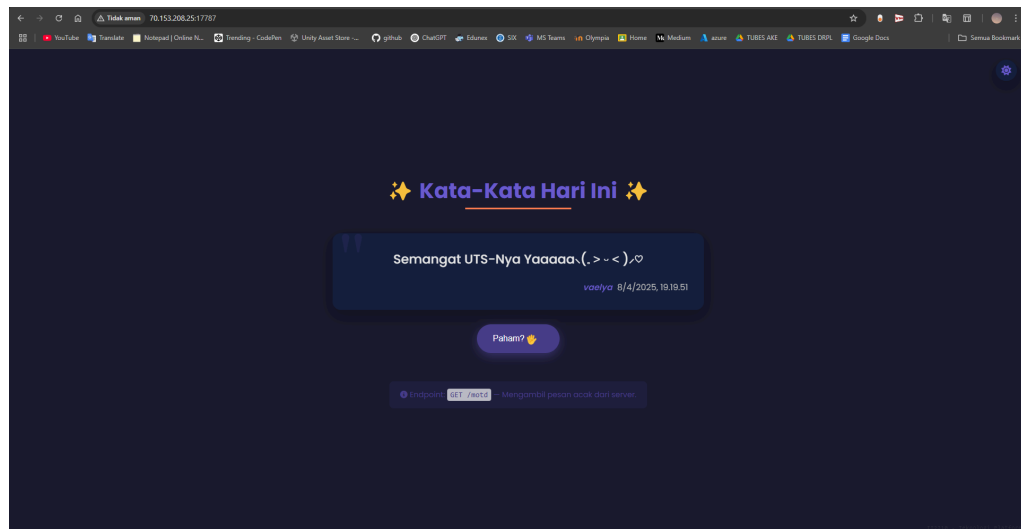
2.5.2. Tampilan Web Browser

Hasil file HTML yang telah dibuat dapat diakses dan ditampilkan melalui prompt di bawah ini yang dapat langsung di-search pada web browser

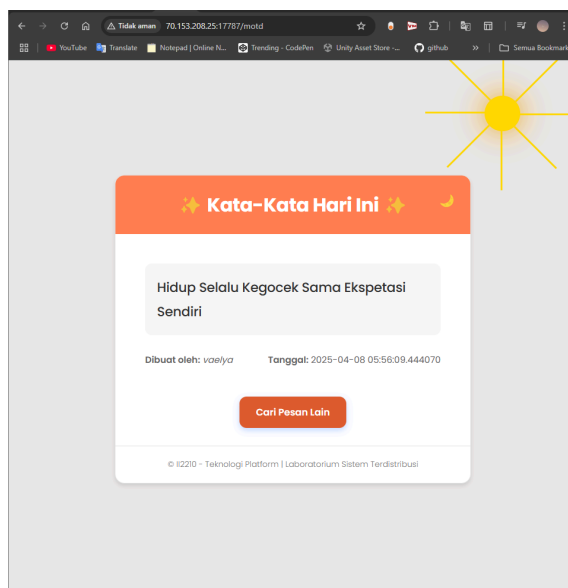
http://<IP Public VM>:17787/

http://<IP Public VM>:17787/motd

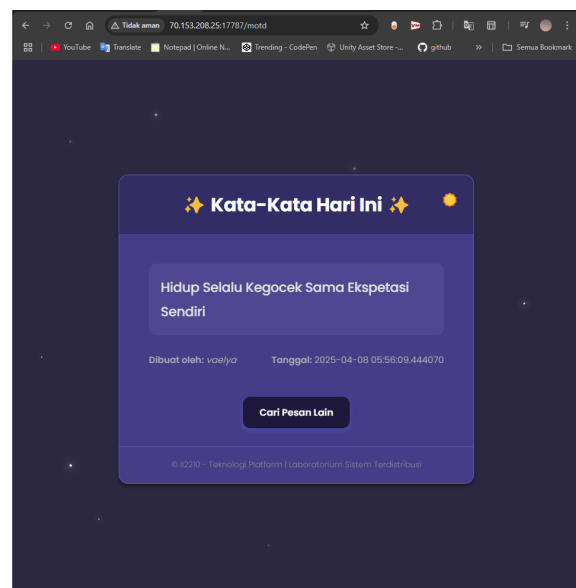
Browser yang saya gunakan pada uji coba kali ini yaitu Chrome, dan perlu diperhatikan bahwa saya membuat dua file HTML yaitu index.html yang dapat diakses melalui <http://70.153.208.25:17787/> dan motd.html yang dapat diakses melalui <http://70.153.208.25:17787/motd>



Gambar 2.24 Tampilan index.html pada <http://70.153.208.25:17787/>



Gambar 2.24 Tampilan motd.html pada <http://70.153.208.25:17787/motd>



*Gambar 2.25 Tampilan motd.html pada
http://70.153.208.25:17787/motd*

Pada HTML endpoint GET /motd, terdapat fitur *switch* tampilan/tema, apabila mengklik icon "☀️" maka akan menampilkan *Gambar 2.24*, sedangkan apabila mengklik icon "🌙" maka akan menampilkan *Gambar 2.25*

BAB III

PENUTUP

3.1. Kesimpulan

Penggunaan **Docker** dalam pengembangan aplikasi web memberikan kemudahan dalam mengelola lingkungan pengembangan yang konsisten. Dengan menggunakan **Dockerfile** dan **docker-compose.yml**, aplikasi dapat dijalankan dalam container yang terisolasi, memudahkan pengelolaan dan deployment tanpa bergantung pada konfigurasi sistem tertentu. **FastAPI** yang digunakan dalam aplikasi ini dapat diintegrasikan dengan **SQLite** di dalam container, yang memberikan solusi penyimpanan data ringan dan efisien. Hal ini memastikan bahwa aplikasi dapat berjalan dengan lancar dalam berbagai platform, terlepas dari perbedaan lingkungan.

Selain itu, untuk meningkatkan keamanan, aplikasi ini menggunakan **TOTP (Time-based One-Time Password)** untuk otentikasi dua faktor, yang memastikan bahwa hanya pengguna yang terotorisasi yang dapat menambah data melalui endpoint **POST /motd**. **GitHub** digunakan untuk mengelola kode sumber aplikasi, memfasilitasi proses **commit**, **push**, dan **pull** yang memastikan kolaborasi yang efisien antar pengembang dan pengelolaan versi kode yang terstruktur. Dengan pendekatan ini, aplikasi web yang dibangun menjadi lebih aman, mudah dikelola, dan dapat di-deploy dengan efisien menggunakan Docker dan GitHub.

DAFTAR PUSTAKA

- FastAPI in Containers - Docker - FastAPI. (n.d.-b).*
<https://fastapi.tiangolo.com/deployment/docker/>
- Tagliaferri, L. (2024, July 1). How to install Git on Ubuntu. DigitalOcean.*
<https://www.digitalocean.com/community/tutorials/how-to-install-git-on-ubuntu>
- Abid, E. (2024, September 2). How to Build Fast API Application using Docker Compose. DigitalOcean.*
<https://www.digitalocean.com/community/tutorials/create-fastapi-app-using-docker-compose>
- How to most correctly restart FastAPI inside a Docker container. (n.d.). Stack Overflow.*
<https://stackoverflow.com/questions/72992017/how-to-most-correctly-restart-fastapi-inside-a-docker-container>
- “Ubuntu.” (2025, February 10). Docker Documentation.*
<https://docs.docker.com/engine/install/ubuntu/>
- Docker engine on Windows • FlowFuse. (n.d.). FlowFuse.*
<https://flowfuse.com/docs/install/docker/windows-docker-ce/>
- Docker: Error: Cannot start container: port has already been allocated. (n.d.). Stack Overflow.*
<https://stackoverflow.com/questions/24993704/docker-error-cannot-start-container-port-has-already-been-allocated>
- “Build, tag, and publish an image.” (2024, August 27). Docker Documentation.*
<https://docs.docker.com/get-started/docker-concepts/building-images/build-tag-and-publish-an-image/>
- Yulian, N. (2021, December 14). Berkenalan Dengan Docker Compose (Docker) Contoh kasus Dockerize PHP , Apache, dan PostgreSQL. Medium.*
<https://medium.com/@nanoyulian/berkenalan-dengan-docker-compose-docker-63208f45ca4c>