Laporan Cloud Computing Tugas 3

Ricardo Supriyanto / 5025221218

Link Repository: https://github.com/Ricardo08S/Task-CloudComputing/tree/main/Task-3

I. Pendahuluan

Tugas ini bertujuan untuk mempraktikkan penggunaan Docker dalam mengelola layanan berbasis container dengan fokus pada konfigurasi dan pengelolaan berbagai skenario layanan. Dalam tugas ini, diminta untuk menjalankan empat kasus berbeda menggunakan Docker Compose, mendokumentasikan langkah-langkah implementasi, dan memberikan penjelasan detail terkait proses dari repositori ini https://github.com/rm77/cloud2023/tree/master/containers/compose/compose. Selain itu, juga ditugaskan untuk mengembangkan satu kasus tambahan (Case 5) berdasarkan kreativitas masing-masing, yang mencakup desain arsitektur, skrip pendukung, dan dokumentasi hasil implementasi.

II. Deskripsi Tugas

Tugas ini terdiri dari empat kasus utama yang harus dijalankan menggunakan Docker Compose. Setiap kasus berhubungan dengan konfigurasi dan pengelolaan layanan yang berbeda. Berikut adalah deskripsi setiap kasus:

Case 1: Menjalankan web server Nginx dengan protokol HTTP.

Case 2: Menjalankan web server Nginx dengan protokol HTTPS menggunakan sertifikat.

Case 3: Menjalankan WordPress secara lokal menggunakan Nginx sebagai web server dengan protokol HTTPS menggunakan sertifikat.

Case 4: Menjalankan aplikasi PHP dan phpMyAdmin dalam container terpisah dan menghubungkannya untuk pengelolaan database.

Case 5 (Case Tambahan): Menjalankan Layanan Flask untuk Generator QR Code Menggunakan Docker Compose

III. Arsitektur dan Implementasi Script

Case 1: Menjalankan Web Server Nginx dengan Protokol HTTP

1. **Deskripsi**
   - Menjalankan layanan web server Nginx menggunakan konfigurasi pada file *docker-compose.yml*. Layanan ini menyajikan berbagai template Bootstrap yang dapat diakses melalui protokol HTTP.
2. **Langkah-langkah**
   - Jalankan perintah untuk memulai container dengan menggunakan konfigurasi pada file *docker-compose.yml*.

3. **Penjelasan**
    - Template Bootstrap yang disajikan oleh layanan dapat diakses melalui **http://localhost:9999** dengan menggunakan protokol HTTP.

Case 2: Menjalankan Web Server Nginx dengan Protokol HTTPS

1. **Deskripsi**
    - Menjalankan layanan web server Nginx dengan dukungan protokol HTTPS untuk menyajikan berbagai template Bootstrap melalui koneksi yang lebih aman.
2. **Langkah-langkah**
    - Gunakan perintah untuk memulai container menggunakan konfigurasi pada file *docker-compose.yml*.
3. **Penjelasan**
    - Layanan web server dapat diakses melalui **https://localhost**, yang menampilkan template Bootstrap menggunakan protokol HTTPS.

Case 3: Menjalankan WordPress dan phpMyAdmin dengan Protokol HTTPS

1. **Deskripsi**
    - Menyediakan layanan WordPress dan phpMyAdmin yang berjalan di dalam container terpisah. Layanan ini diakses menggunakan protokol HTTPS untuk mendukung koneksi yang aman.
2. **Langkah-langkah**
    - Jalankan container menggunakan konfigurasi yang terdapat dalam file *docker-compose.yml*.
3. **Penjelasan**
    - phpMyAdmin dapat diakses melalui **https://localhost:30081** untuk mengelola basis data MySQL.
    - WordPress tersedia di **https://localhost** atau domain yang telah ditentukan.

Case 4: Menjalankan Layanan Aplikasi PHP dan phpMyAdmin

1. **Deskripsi**
    - Menyediakan aplikasi berbasis PHP yang berjalan bersama dengan phpMyAdmin di dalam container terpisah. Semua dependensi aplikasi PHP dikelola melalui Docker.
2. **Langkah-langkah**
    - Gunakan file *docker-compose.yml* untuk memulai container.
3. **Penjelasan**
    - Aplikasi PHP dapat diakses melalui **http://localhost:34001**.
    - phpMyAdmin tersedia pada **http://localhost:10000** untuk pengelolaan database MySQL.

Case 5: Menjalankan Layanan Flask untuk Generator QR Code Menggunakan Docker Compose

1. **Deskripsi**

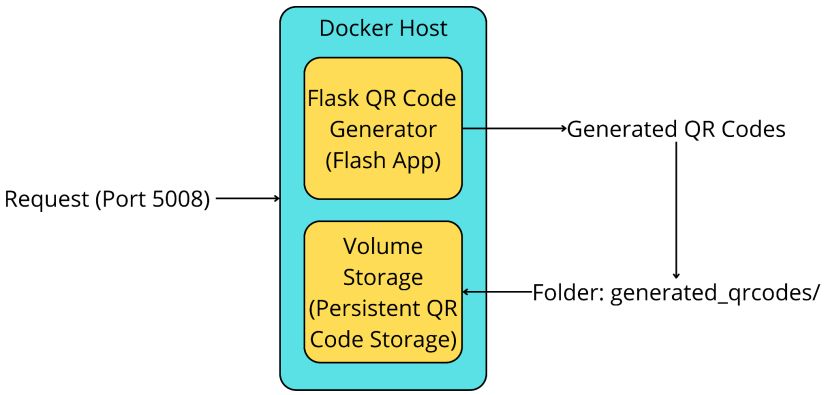
- Menjalankan aplikasi Flask yang berfungsi untuk menghasilkan QR Code secara dinamis berdasarkan data yang diterima melalui API. Layanan ini berjalan di dalam container Docker dan menggunakan volume untuk menyimpan hasil QR Code secara persisten.

2. **Langkah-langkah**
   - Jalankan container dengan perintah *docker-compose up --build*. Dengan isi dari *docker-compose.yml* seperti ini:

```yaml
version: '3.8'
services:
  flask_app:
    image: python:3.9-slim
    container_name: flask_qrcode_generator
    working_dir: /app
    ports:
      - "5008:5008"
    volumes:
      - ./app:/app
      - ./generated_qrcodes:/app/generated_qrcodes
    command: >
      sh -c "pip install -r requirements.txt && python app.py"
```

3. **Penjelasan**
   - Endpoint API untuk menghasilkan QR Code tersedia di **http://localhost:5008/generate**. Data JSON dikirim melalui POST request, dan respons berupa gambar QR Code dalam format PNG.
   - Folder **generated_qrcodes/** yang dipetakan sebagai volume akan digunakan untuk menyimpan hasil QR Code yang dihasilkan oleh layanan.

Arsitektur **Case 5**:



- **Docker Host**
  - **Deskripsi:**
    Mesin fisik atau virtual tempat container Docker dijalankan. Ini berfungsi sebagai lingkungan untuk menjalankan aplikasi Flask yang menghasilkan QR Code dan menyimpan file QR Code yang dihasilkan secara persisten.
  - **Fungsi:**
    Menyediakan sumber daya untuk container, termasuk jaringan, penyimpanan, dan pengolahan data.

- **Flask QR Code Generator Container**
  - **Deskripsi:**
    Container ini menjalankan aplikasi Flask yang bertugas menghasilkan QR Code berdasarkan data yang diterima melalui API. Aplikasi ini mendengarkan pada endpoint **/generate** yang menerima data dalam format JSON dan mengembalikan gambar QR Code dalam format PNG sebagai respons.
  - **Detail Konfigurasi:**
    - **Port:** Akses dilakukan melalui port **5008** di host.
    - **Fungsi:** Menerima request POST berisi data JSON untuk menghasilkan QR Code.
    - **Akses:** Pengguna mengirimkan request ke **http://localhost:5008/generate** dengan payload JSON, dan responsnya berupa gambar QR Code.
- **Volume Penyimpanan QR Code**
  - **Deskripsi:**
    Folder **generated_qrcodes/** di host dipetakan sebagai volume yang digunakan untuk menyimpan file QR Code yang dihasilkan. Volume ini memastikan bahwa file QR Code disimpan secara persisten meskipun container dimulai ulang.
  - **Detail Konfigurasi:**
    - **Volume: generated_qrcodes/**
    - **Fungsi:** Menyimpan file QR Code yang dihasilkan oleh layanan Flask.
- **File Konfigurasi Docker Compose (docker-compose.yml)**
  - **Deskripsi:**
    File konfigurasi Docker Compose yang mendefinisikan layanan aplikasi Flask dan volume penyimpanan QR Code. Ini memastikan bahwa semua container yang diperlukan dijalankan dengan konfigurasi yang benar.
  - **Detail Konfigurasi:**
    - **Layanan:** Satu layanan yang menjalankan aplikasi Flask.
    - **Volume:** Volume yang digunakan untuk menyimpan hasil QR Code secara persisten.
    - **Port:** Container Flask mendengarkan pada port **5008** di host.

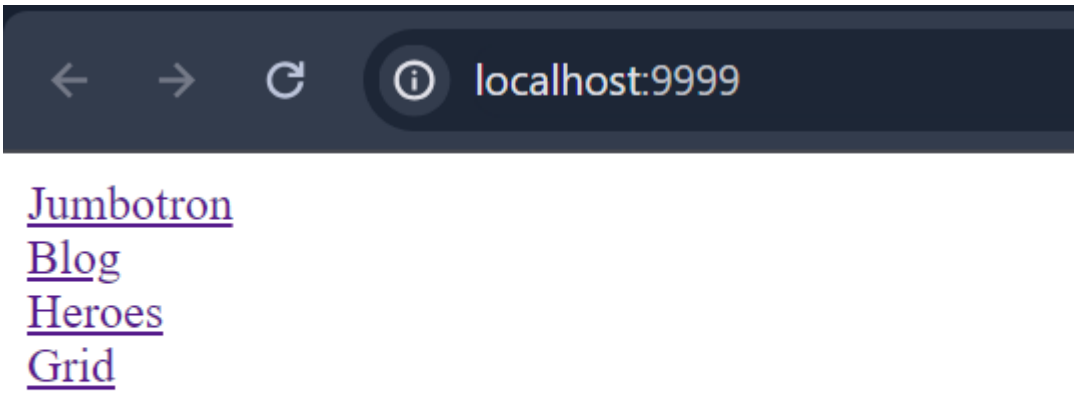
Kapan Skenario Ini Cocok Digunakan?

Skenario ini cocok digunakan untuk aplikasi yang membutuhkan kemampuan untuk menghasilkan QR Code secara dinamis berdasarkan data yang diterima melalui API, seperti:

- Sistem aplikasi yang memungkinkan pengguna untuk membuat QR Code untuk berbagai kebutuhan (misalnya, pembayaran, link web, dll.).
- Aplikasi berbasis web yang memungkinkan pengguna mengirimkan data dan menerima QR Code secara langsung melalui antarmuka pengguna.
- Aplikasi yang memerlukan penyimpanan QR Code yang dihasilkan secara persisten, misalnya untuk digunakan atau diunduh di masa mendatang.

IV. Screenshot Implementasi dan Penjelasannya

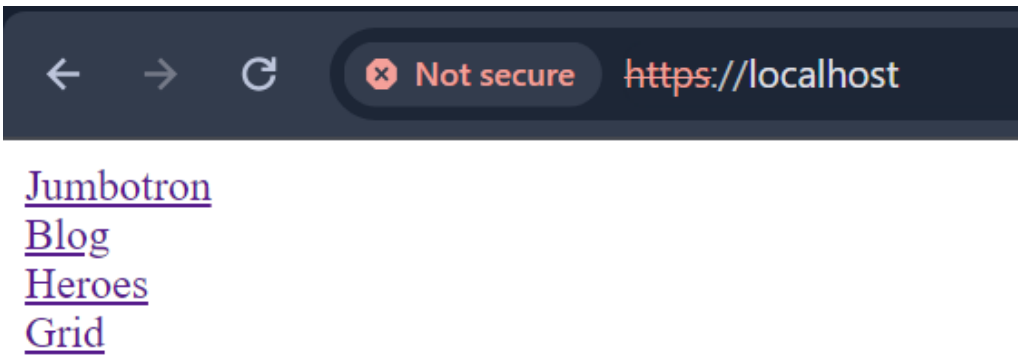
Case 1: Web Server Nginx dengan Protokol HTTP



Tampilan browser yang menunjukkan halaman template Bootstrap di **http://localhost:9999**.

Penjelasan:
Gambar di atas menunjukkan halaman web yang ditampilkan pada **http://localhost:9999** dengan protokol HTTP setelah menjalankan Case 1. Halaman ini menampilkan template Bootstrap yang dapat diakses melalui server Nginx yang dikonfigurasi untuk menggunakan protokol HTTP.

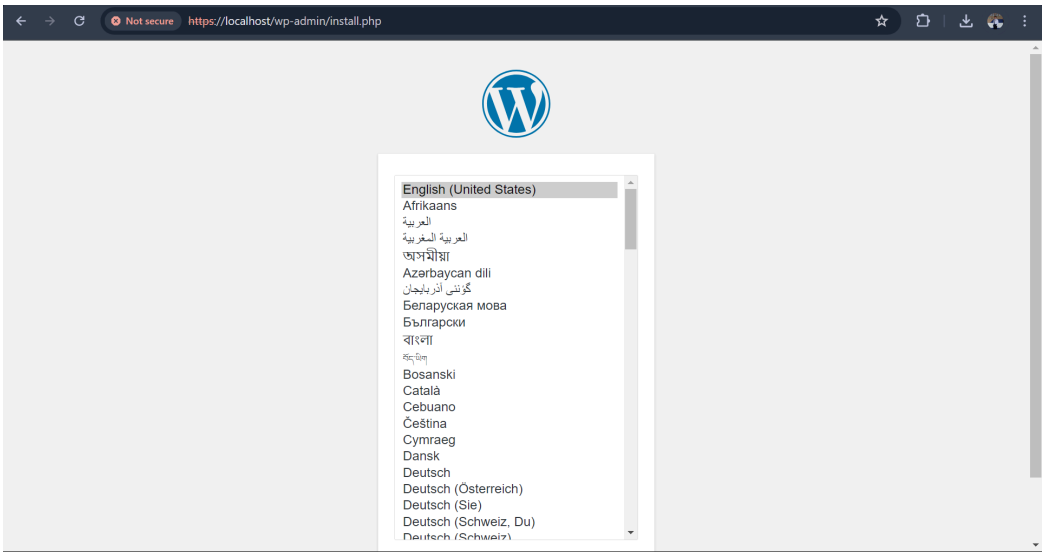
Case 2: Menjalankan Web Server Nginx dengan Protokol HTTPS



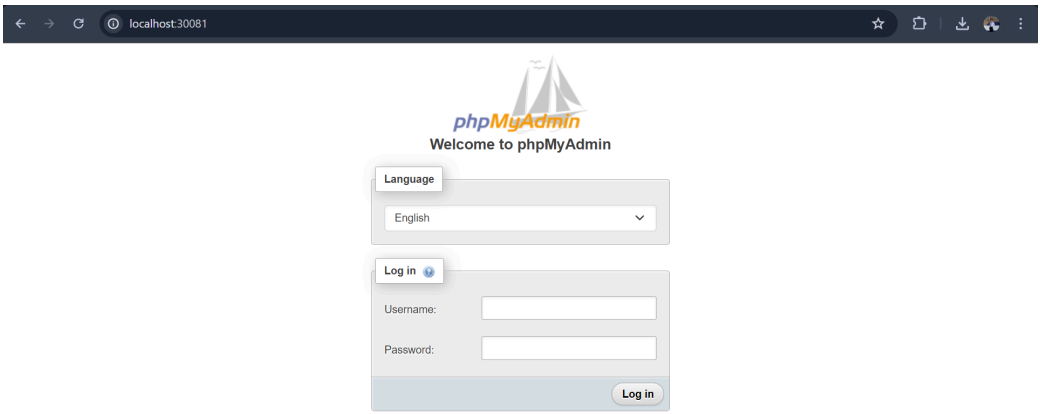
Tampilan browser yang menunjukkan halaman template Bootstrap di **https://localhost** (dengan indikator gembok untuk HTTPS).

Penjelasan:
Gambar di atas menunjukkan halaman web yang ditampilkan pada **https://localhost** dengan protokol HTTPS setelah menjalankan Case 2. Halaman ini menampilkan template Bootstrap yang dapat diakses melalui server Nginx yang dikonfigurasi untuk menggunakan protokol HTTPS dan sertifikat yang valid.

Case 3: WordPress dan phpMyAdmin dengan Protokol HTTPS



Tampilan browser yang menunjukkan halaman **WordPress** di **https://localhost/wp-admin/install.php**.

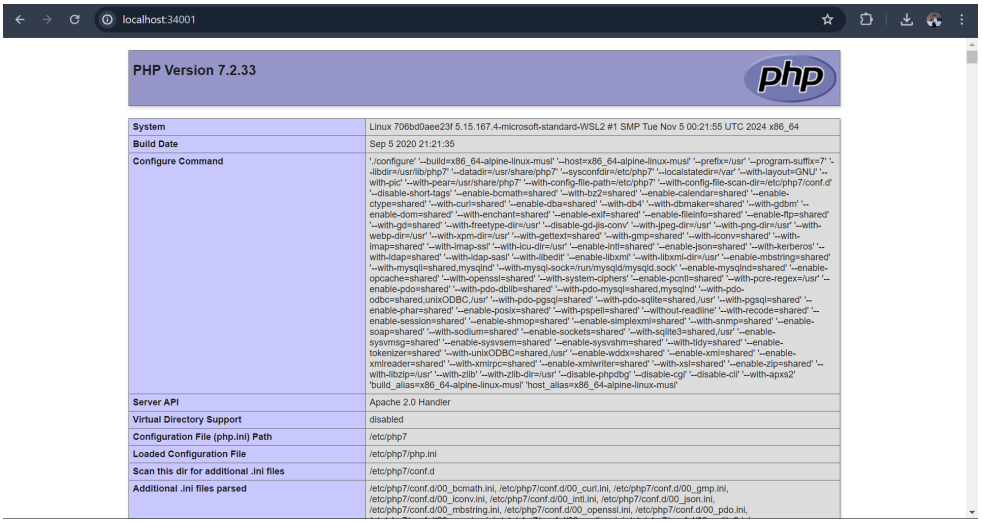


Tampilan browser yang menunjukkan **phpMyAdmin** di **https://localhost:30081**.

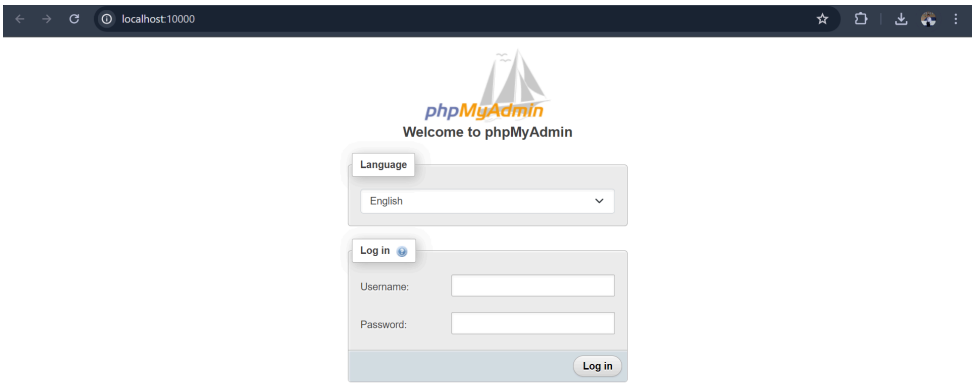
Penjelasan:
Gambar di atas menunjukkan interface **WordPress** yang diakses pada **https://localhost/wp-admin/install.php** setelah menjalankan Case 3, menunjukkan bahwa web server Nginx berhasil dikonfigurasi dengan benar untuk mengakses dan menampilkan

halaman WordPress dengan protokol HTTPS.
Gambar di atas juga menunjukkan interface **phpMyAdmin** yang diakses pada **https://localhost:30081**, yang menunjukkan bahwa MySQL database berhasil terkoneksi dan dapat dikelola.

Case 4: Layanan Aplikasi PHP dan phpMyAdmin



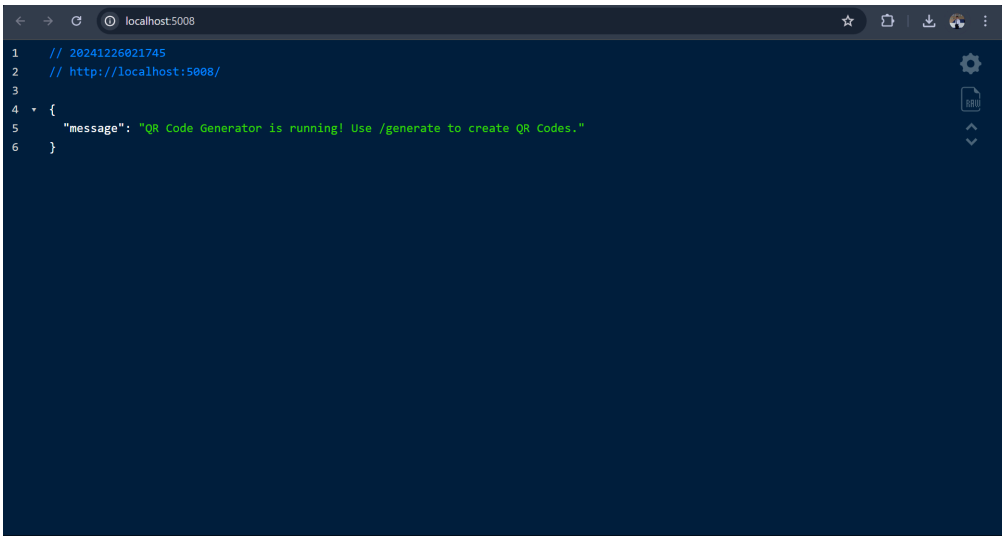
Tampilan browser yang menampilkan aplikasi PHP di **http://localhost:34001**.



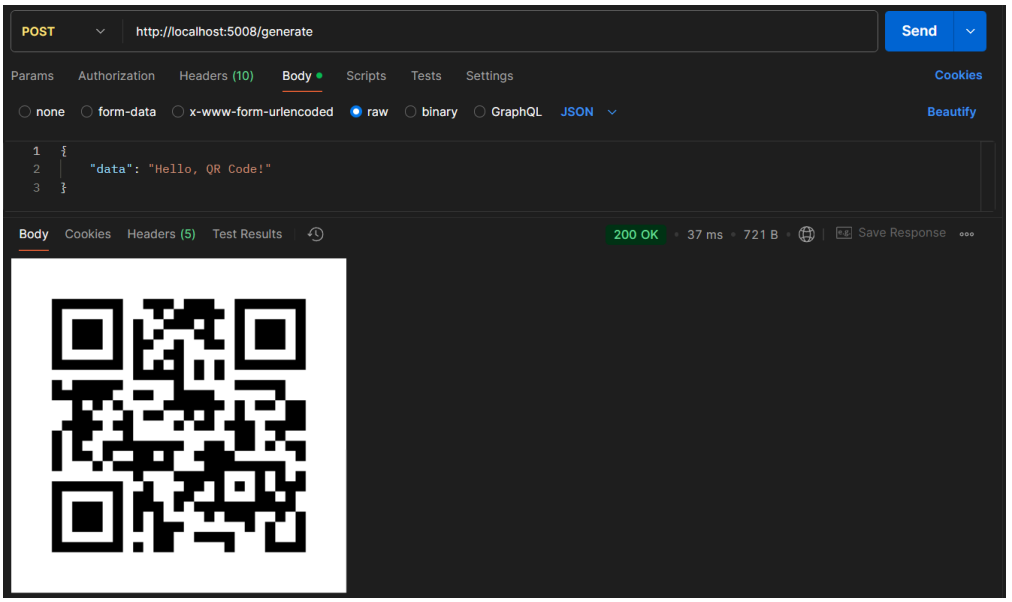
Tampilan browser yang menunjukkan **phpMyAdmin** di **http://localhost:10000**.

Penjelasan:
Gambar di atas menunjukkan halaman aplikasi PHP yang diakses pada **http://localhost:34001** setelah menjalankan Case 4. Halaman ini menampilkan layanan aplikasi PHP lengkap beserta dependensinya.
Gambar di atas juga menunjukkan interface **phpMyAdmin** yang diakses pada **http://localhost:10000**, yang menunjukkan bahwa MySQL database berhasil terkoneksi dan dapat dikelola.

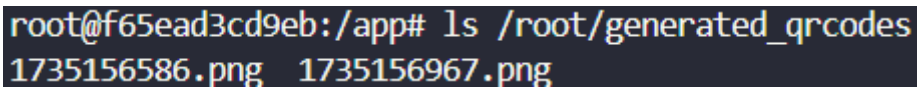
Case 5: Menjalankan Layanan Flask untuk Generator QR Code Menggunakan Docker Compose



Tampilan browser yang menunjukkan bahwa layanan Flask berjalan dengan benar dan dapat diakses di **http://localhost:5008**.



Tampilan Postman yang menunjukkan pengiriman data JSON ke endpoint **/generate** untuk menghasilkan QR Code.



Tampilan folder generated_qrcodes/ di host yang menunjukkan file QR Code yang telah dihasilkan

Penjelasan:
Gambar di atas menunjukkan bahwa layanan Flask berjalan dengan benar dan dapat diakses di **http://localhost:5008** setelah menjalankan Case 5. Layanan ini mengharuskan pengiriman data JSON ke endpoint **/generate** menggunakan metode POST. Setelah data diterima, layanan akan menghasilkan QR Code dalam format PNG sebagai respons.
Gambar di atas juga menunjukkan penggunaan Postman untuk mengirimkan data JSON ke endpoint **/generate** dan mendapatkan respons berupa QR Code. Selain itu, folder **generated_qrcodes/** di docker host juga menunjukkan file QR Code yang telah dihasilkan oleh aplikasi Flask dan disimpan secara persisten.

V. Kesimpulan

Dalam laporan ini, saya berhasil menjalankan 4 kasus dari repository yang diberikan dan mengembangkan **Case 5: Menjalankan Layanan Flask untuk Generator QR Code Menggunakan Docker Compose**. Saya juga menyediakan penjelasan terkait *script* yang digunakan, gambar arsitektur, serta screenshot yang menunjukkan konfigurasi dan akses ke aplikasi yang dijalankan. Penggunaan Docker Compose sangat memudahkan pengelolaan kontainer dan layanan-layanan yang terintegrasi, serta memungkinkan pengembangan lebih lanjut seperti pada **Case 5** yang mengimplementasikan layanan Flask untuk menghasilkan QR Code secara dinamis.