LAPORAN HASIL AKHIR PROJECT AKSO

MICROSERVICE SEDERHANA DENGAN DOCKER

Disusun Oleh:

1. **Desminica Maharani S. (25031554190)**
2. **Nela Elvareta Destya M. (25031554056)**
3. **Ummul Lutfiah (25031554167)**
4. **Zahwa Aulia Savila (25031554085)**

UNIVERSITAS NEGERI SURABAYA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

PROGRAM STUDI S1 SAINS DATA

2025-2026

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Perkembangan sistem informasi modern menuntut arsitektur perangkat lunak yang bersifat modular, fleksibel, dan mudah dikembangkan. Salah satu pendekatan yang banyak digunakan untuk memenuhi kebutuhan tersebut adalah arsitektur microservice, yaitu pendekatan di mana sebuah sistem besar dipecah menjadi beberapa sub-layanan yang berdiri sendiri, saling terhubung melalui Application Programming Interface (API), serta dapat dikembangkan dan dikelola secara independen. Arsitektur ini banyak diterapkan pada sistem berskala besar karena mampu meningkatkan kecepatan pengembangan, skalabilitas, serta ketahanan sistem terhadap kegagalan.

Dalam penerapan microservice, teknologi containerization menjadi komponen penting, salah satunya melalui penggunaan Docker. Docker memungkinkan setiap sub-layanan dijalankan dalam lingkungan terisolasi yang ringan, konsisten, dan mudah direplikasi. Dengan bantuan Docker dan docker-compose, proses deployment, konfigurasi jaringan internal, manajemen dependensi antar layanan, serta pengelolaan sumber daya dapat dilakukan secara lebih efisien.

Pada proyek ini, dikembangkan sebuah layanan akademik sederhana berbasis microservice yang terdiri dari beberapa sub-layanan utama, yaitu API Gateway, layanan otentikasi (auth-service), dan basis data otentikasi (auth-db). Sistem ini merupakan legacy software yang telah memiliki kerangka awal, sehingga tim berperan sebagai DevOps untuk melengkapi konfigurasi, memastikan seluruh layanan dapat berjalan dengan baik, serta melakukan pemrograman tambahan pada API, khususnya untuk perhitungan Indeks Prestasi Semester (IPS) mahasiswa.

Melalui proyek ini, mahasiswa diharapkan mampu memahami konsep arsitektur microservice secara praktis, mengimplementasikan Docker dan docker-compose dalam skenario nyata, serta mengintegrasikan aspek sistem operasi, jaringan, dan pemrograman API dalam satu kesatuan sistem yang berjalan secara utuh.

1.2 Tujuan

Tujuan dari pelaksanaan proyek microservice ini adalah sebagai berikut:

1. Memahami konsep arsitektur microservice dan penerapannya dalam pengembangan sistem berbasis layanan terdistribusi.
2. Mengimplementasikan containerization menggunakan Docker, termasuk pembuatan dan konfigurasi container untuk setiap sub-layanan yang terlibat.
3. Mengonfigurasi docker-compose.yml agar seluruh sub-layanan (API Gateway, auth-service, dan auth-db) dapat berjalan secara terintegrasi, stabil, dan sesuai dengan spesifikasi yang ditentukan.
4. Menerapkan konsep internal networking dan volume management pada Docker untuk mendukung komunikasi antar layanan dan penyimpanan data yang persisten.
5. Mengembangkan API sederhana pada layanan penilaian untuk melakukan perhitungan IPS mahasiswa berdasarkan logika yang telah ditentukan.
6. Meningkatkan pemahaman peran DevOps, khususnya dalam menangani deployment, dependency management, dan konfigurasi ulang sistem legacy.
7. Melatih kerja sama tim dan tanggung jawab individu, baik dalam penyelesaian tugas teknis maupun melalui mekanisme penilaian rekan sejawat.

BAB II

IMPLEMENTASI SISTEM DAN PROGRAM

2.1 Docker-compose.yml

Docker-compose.yml adalah file yang digunakan untuk mengatur dan menjalankan seluruh layanan microservice dalam satu system secara terintegrasi. Pada konfigurasi ini, system terdiri dari beberapa service utama, yaitu API Gateway, Auth Service, Acad Service, dan masing - masing database pendukungnya. Selain itu, file ini juga mendefinisikan jaringan internal dan volume untuk mnjamin komunikasi antar layanan serta penyimpanan data yang persisten.

1. Api Gateway (api-gateway)

   Service API Gateway berfungsi sebagai pintu masuk utama (entry point) sistem. API Gateway dijalankan menggunakan image nginx:alpine dan diberi nama container api-gateway. Service ini hanya dijalankan setelah Auth Service dan Acad Service aktif, sehingga tidak terjadi kesalahan ketika meneruskan permintaan. File konfigurasi nginx.conf dimapping ke dalam container agar aturan routing dapat diatur dari luar container. Port 8081 pada komputer host dipetakan ke port 80 di dalam container. Artinya, pengguna dapat mengakses sistem melalui alamat localhost:8081. API Gateway juga dihubungkan ke jaringan internal agar dapat berkomunikasi dengan service lain secara aman.

2. Auth Service (auth-service)

   Auth Service merupakan layanan yang bertanggung jawab terhadap proses otentikasi pengguna, seperti login dan pembuatan token akses. Service ini memastikan bahwa hanya pengguna yang telah terdaftar dan memiliki akun yang valid yang dapat mengakses layanan lain di dalam sistem. Auth service dijalankan pada port 3001. Port ini digunakan untuk menerima request dari Api Gateway (api-gateway).

   Beberapa environment yang digunakan untuk mengatur konfigurasi Auth service, seperti NODE_ENV digunakan untuk menentukan metode pengembangan. PORT, menentukan port yang digunakan oleh service. MONGO_URI berisi alamat database MongoDB yang digunakan untuk menyimpan data akun pengguna. Selain itu, JWT_SECRET digunakan sebagai kunci rahasia dalam pembuatan token JWT yang berfungsi sebagai identitas pengguna setelah login. Auth

service juga dibatasi penggunaan sumber daya CPU dan memori dengan CPU (cpus) adalah 0. 5 virtual core dan memori (mem_limit) adalah 512 mb.

3. Auth Database (auth-db)

   Auth Database menggunakan MongoDB versi 6.0 sebagai media penyimpanan data akun pengguna. Database ini menyimpan informasi yang diperlukan oleh Auth Service, seperti username dan password serta database dijalankan dengan nama container auth-db. Pada konfigurasi Auth Database (auth-db) digunakan pengaturan “restart: always”. Pengaturan ini bertujuan untuk memastikan bahwa container database akan otomatis dijalankan kembali apabila terjadi gangguan.

4. Acad Service (acad-service)

   Acad Service merupakan layanan yang bertugas menangani proses akademik dalam sistem yang dibangun. Pada tugas ini, Acad Service difokuskan untuk melakukan pengolahan data akademik sederhana, yaitu perhitungan Indeks Prestasi Semester (IPS) mahasiswa. Service ini dijalankan pada port 3002 dan hanya dapat diakses melalui Api Gateway. Acad Service juga terhubung dengan database akademik yang berjalan pada container acad-db, di mana alamat dan konfigurasi database diatur melalui environment variable DB_HOST, DB_PORT, DB_NAME, DB_USER, dan DB_PASSWORD.

5. Acad Database (acad-db)

   Acad Database menggunakan PostgreSQL versi 15 sebagai media penyimpanan data akademik. Database ini digunakan oleh Acad Service untuk mendukung proses pengolahan data. Konfigurasi database meliputi nama database, username, dan password yang ditentukan melalui variabel environment. Volume digunakan untuk menyimpan data database secara permanen agar data tidak hilang ketika container dihentikan atau dibuat ulang.

6. Networks

   Pada sistem ini digunakan sebuah jaringan internal Docker dengan nama microservices-network. Jaringan ini menggunakan driver bridge, yang merupakan tipe jaringan standar pada Docker untuk menghubungkan beberapa container dalam satu sistem. Pengaturan ipam dengan subnet 172.16.0.0/28 digunakan untuk mengatur rentang alamat IP yang dipakai oleh container di

dalam jaringan tersebut. Dengan adanya pengaturan jaringan internal ini, komunikasi antar service menjadi lebih aman dan mudah dikelola.

7. Volumes

Volume digunakan untuk menyimpan data secara permanen diluar container. Dengan penggunaan volume, data pada database tidak akan hilang meskipun container dihentikan atau dijalankan ulang. Hal ini penting untuk menjaga konsistensi dan keamanan data selama sistem berjalan.

2.2 Main.py

File main.py merupakan implementasi utama dari Acad Service yang dibangun menggunakan framework FastAPI. Layanan ini berfungsi sebagai backend yang menangani pengolahan data akademik mahasiswa, khususnya perhitungan Indeks Prestasi Semester (IPS). Acad Service berperan sebagai penghubung antara database akademik yang dibuat melalui file db_kelas.sql dengan sistem aplikasi yang mengakses data tersebut melalui API.

Pada bagian awal file main.py, aplikasi FastAPI diinisialisasi dengan membuat objek app menggunakan FastAPI(). Objek app merupakan inti dari Acad Service yang berfungsi sebagai pengelola seluruh API yang dibuat. Seluruh endpoint, middleware, dan konfigurasi layanan didaftarkan melalui objek ini. Dengan demikian, app bertindak sebagai pusat pengaturan yang mengatur alur permintaan dan respon dalam sistem. Selain itu, aplikasi dikonfigurasi dengan middleware CORS (Cross-Origin Resource Sharing). Middleware ini memungkinkan Acad Service untuk diakses oleh service lain seperti API Gateway tanpa mengalami kendala perbedaan domain. Konfigurasi CORS ini mendukung penerapan arsitektur microservices yang membutuhkan komunikasi antar layanan.

Acad Service menggunakan database PostgreSQL sebagai media penyimpanan data akademik. Konfigurasi koneksi database diambil dari environment variable yang telah didefinisikan pada file docker-compose.yml. Database yang digunakan memiliki struktur tabel yang dibuat melalui file db_kelas.sql. Pendekatan ini membuat kode program menjadi lebih fleksibel karena tidak bergantung langsung pada konfigurasi tertentu dan mudah dijalankan di lingkungan Docker. Untuk merepresentasikan data mahasiswa, digunakan model data Mahasiswa yang didefinisikan

menggunakan Pydantic. Model ini merepresentasikan struktur data pada tabel mahasiswa dalam database, yang meliputi NIM, nama, jurusan, dan angkatan.

Koneksi ke database dikelola menggunakan fungsi get_db_connection. Fungsi ini bertugas membuka koneksi sebelum proses query dilakukan dan menutup kembali koneksi setelah proses selesai. Apabila terjadi kesalahan saat proses query, sistem akan melakukan rollback untuk menjaga konsistensi data. Mekanisme ini membuat pengelolaan database menjadi lebih aman dan stabil.

Endpoint API /api/acad/mahasiswa digunakan untuk mengambil seluruh data mahasiswa dari tabel mahasiswa. Data yang ditampilkan berasal langsung dari database yang telah diisi menggunakan perintah INSERT pada file db_kelas.sql. Endpoint ini dapat digunakan untuk menampilkan atau memverifikasi data mahasiswa yang tersimpan dalam sistem akademik. Selanjutnya, endpoint API /api/acad/ips digunakan untuk menghitung Indeks Prestasi Semester (IPS) mahasiswa berdasarkan NIM. Pada proses ini, sistem mengambil data dari tabel mahasiswa, krs, dan mata_kuliah yang seluruhnya dibuat melalui file db_kelas.sql. Data nilai huruf diambil dari tabel krs, sedangkan data jumlah SKS diambil dari tabel mata_kuliah.

Nilai huruf yang diperoleh mahasiswa kemudian dikonversi menjadi bobot angka sesuai dengan aturan akademik. Bobot nilai tersebut dikalikan dengan jumlah SKS setiap mata kuliah, kemudian seluruh hasilnya dijumlahkan dan dibagi dengan total SKS untuk menghasilkan nilai IPS. Proses ini merupakan penerapan perhitungan IPS secara manual yang diotomatisasi melalui sistem.

BAB 3
HASIL DAN PEMBAHASAN

3.1 Hasil Dan Pembahasan Project

Alur kerja sistem pada proyek ini berjalan seperti berikut: Pertama, Docker Compose digunakan untuk menjalankan semua layanan yang dibutuhkan. Kemudian, container acad-service dijalankan sebagai layanan API akademik yang menggunakan FastAPI. Aplikasi pada acad-service dijalankan lewat file main.py, yang menyediakan endpoint /api/acad/ips. Endpoint ini terhubung ke database PostgreSQL (acad-db) untuk mengambil data akademik mahasiswa. Sistem kemudian melakukan perhitungan Indeks Prestasi Semester (IPS) dan menghasilkan respons berupa format JSON, lalu menampilkan hasilnya melalui Thunder Client. Proyek ini dibangun dengan konsep microservice yang dijalankan dalam lingkungan Docker menggunakan Docker Compose. Docker Compose berfungsi untuk menyiapkan dan menjalankan beberapa layanan secara bersamaan, yaitu layanan API akademik (acad-service) dan layanan basis data (acad-db). Dengan demikian, kedua layanan tersebut dapat saling terhubung dan berjalan secara terkoordinasi.

Layanan acad-service berisi aplikasi backend yang dibangun menggunakan framework FastAPI dan dijalankan dari file main.py. Aplikasi ini menyediakan endpoint API untuk mengambil data mahasiswa dan menghitung IPS. Konfigurasi koneksi database, seperti host, port, nama database, dan kredensial, didefinisikan dalam file docker-compose.yml. Konfigurasi ini dikirimkan ke aplikasi melalui environment variable, sehingga aplikasi bisa terhubung dengan database PostgreSQL tanpa perlu pengaturan tambahan. Ketika pengguna mengakses endpoint http://localhost:port/api/acad/ips?nim=xxxxx, aplikasi akan mengambil data nilai dan jumlah SKS mahasiswa dari database PostgreSQL.

Selanjutnya, sistem mengonversi nilai huruf (A, B+, B, dan seterusnya) menjadi bobot angka, kemudian menghitung IPS dengan membagi total bobot dengan total SKS. Proses perhitungan dilakukan sepenuhnya di dalam file main.py. Hasil perhitungan IPS diberikan kembali kepada pengguna dalam bentuk response JSON. Untuk menguji sistem, digunakan Thunder Client yang terdapat di Visual Studio Code. Thunder Client berperan sebagai alat untuk mengirimkan request ke endpoint API dan menampilkan response yang dihasilkan sistem tanpa terlibat langsung dalam perhitungan data.

Berdasarkan hasil pengujian, Thunder Client berhasil menampilkan output perhitungan IPS mahasiswa sesuai dengan data yang tersimpan di database, seperti yang terlihat pada gambar hasil pengujian yang telah dilampirkan. Dengan alur kerja tersebut, sistem mampu menghitung dan menampilkan nilai IPS mahasiswa secara otomatis melalui layanan API yang terintegrasi dengan database dan dijalankan dalam lingkungan Docker.

BAB 4
KESIMPULAN

Berdasarkan hasil implementasi dan pengujian yang dilakukan, bisa disimpulkan bahwa sistemmicroservice layanan akademik yang dibuat dengan Docker Compose berjalan lancar. Layanan API akademik (acad-service) berhasil dijalankan di dalam container Docker dan terhubung dengan database PostgreSQL (acad-db) sesuai dengan pengaturan yang sudah ditentukan.

Pengujian menggunakan Thunder Client menunjukkan bahwa endpoint /api/acad/ips mampu menerima input berupa NIM mahasiswa, mengambil data nilai dan SKS dari database, serta menghitung Indeks Prestasi Semester (IPS) secara otomatis berdasarkan logika yang sudah diimplementasikan di dalam file main.py. Hasil perhitungan IPS ditampilkan dalam bentuk respons JSON dan bisa diterima oleh client dengan baik.

Dengan demikian, sistem yang dikembangkan telah mencapai tujuan utama proyek, yaitu membuat layanan akademik berbasis API yang bisa menghitung dan menampilkan nilai IPS mahasiswa secara terintegrasi, serta berjalan stabil dalam lingkungan Docker.

.