Laporan Project Docker: Perancangan, Implementasi, dan Modifikasi Layanan Microservices untuk Sistem Akademik.

Disusun:

Sonnya Gratia Plena Leo	25031554138
Istu Maya Adelliya	25031554005
Alfatkhul Khakim	25031554103

SAINS DATA 2025D FAKULTAS

MATEMATIKA DAN ILMU PENGETAHUAN

ALAM 2025/2026

DAFTAR ISI

JUDUL. 1
DAFTAR ISI 2
BAB I 3
PENDAHULUAN 3
1.1 Latar Belakang 3
1.2 Tujuan 4
BAB II 5
GAMBARAN PROYEK DOCKER 5
2.1 Struktur File dan Folder Proyek 5
2.2 Deskripsi Singkat Tiap Layanan 7
BAB III 10
PERUBAHAN DAN MODIFIKASI 10
3.1 Rangkuman Perubahan yang Dilakukan 10
3.2 Alasan dan Detail Setiap Perubahan 10
3.3 Modifikasi Lain pada File atau Folder 11
3.4 Dampak Perubahan terhadap Cara Kerja Sistem 12
BAB IV 14
PENUTUP 14
4.1 Kesimpulan 14
DAFTAR PUSTAKA 15

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi dan informasi mendorong Perguruan Tinggi dalam mengelola data akademik secara efisien waktu, akurat, dan terintegrasi. Data Akademik di antaranya data Mahasiswa, mata kuliah hingga perhitungan indeks prestasi studi (IPS). Pada awalnya sistem akademik dibangun secara monolitik yang menyebabkan kesulitan dalam pengembangannya, karena setiap perubahan fitur dapat berdampak pada seluruh sistem aplikasi dan berisiko meningkatkan gangguan layanan. Dengan ini, arsitektur sistem menjadi peluang untuk menciptakan sistem yang lebih modular, mudah dikembangkan, dan dapat beradaptasi terhadap kebutuhan akademik hingga bisnis.

Salah satu pendekatan arsitektur sistem yang berkembang untuk menghadapi tantangan tersebut adalah microservices. Microservices adalah gaya arsitektur yang mengelola aplikasi menjadi layanan-layanan kecil yang dapat dikembangkan, diuji, dan dioperasikan secara terpisah. Hal ini dijelaskan oleh Thones sebagai *"small application that can be deployed independently, scaled independently, and tested independently and that has a single responsibility".* Pendekatan ini memungkinkan setiap layanan fokus dalam satu tujuan, sehingga perubahan pasa satu layanan, misalnya layanan autentikasi atau layanan akademik tidak akan mengganggu layanan yang lain.

Di lain sisi, Teknologi container seperti Docker hadir sehingga memberikan cara yang lebih ringan dan fleksibel untuk melakukan deployment aplikasi microservisces di lingkungan komputasi di era saat ini. Docker dan Docker Compose dapat menyusun setiap layanan independen beserta dependensinya di dalam container independen, sehingga proses penyebaran, skala, dan menjamin layanan lebih konsisten di berbagai lingkungan.

Dalam konteks sistem akademik, arsitektur microservices memberikan peluang untuk memisahkan layanan autentikasi, layanan pengelolaan, data akademik, dan layanan perhitungan IPS ke dalam service yang saling terhubung satu satu lain dan berdiri secara independen. Dengan hal ini akan memudahkan suatu institusi atau perguruan tinggi dalam menambahkan fitur baru atau menyesuaikan aturan akademik tanpa perlu merombak keseluruhan sistem. Oleh Karena itu, Perancangan, implementasi, dan modifikasi layanan microservces untuk sistem akademik berbasis

Docker menjadi topik yang tim kami pilih dan juga sebagai Project tugas akhir kami.

1.2 Tujuan

1. Memodifikasi layanan yang diberikan agar bisa menjalankan sistem dan mendapat output
2. Menjelaskan tentang perubahan-perubahan yang telah tim kami modifikasi dalam file docker-compose.yml
3. Menjelaskan konsep-konsep mendasar tentang microservices

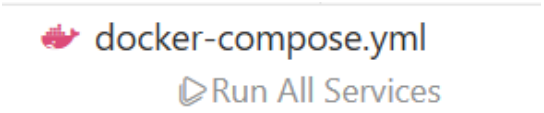
BAB II
GAMBARAN PROYEK DOCKER

2.1 Struktur File dan Folder Proyek

Project docker yang dirancang dan yang telah dimodifikasi, telah tersusun dalam beberapa file dan folder yang masing-masing memiliki fungsi spesifik untuk mendukung seluruh jalannya layanan microservices. Dengan file yang paling utama adalah docker-compose.yml, yang didalamnya sudah mencangkup tentang seluruh layanan, jaringan, dan volume untuk database yang digunakan. Setiap layanan memiliki folder masing-masing yang berisi kode sumber dan konfigurasi docker, sehingga mudah untuk proses build, pengujian, dan deploy. Hal ini mengikuti konsep arsitektur multi-container untuk aplikasi microservices yang di mana setiap layanan dapat berjalan terisolasi namun saling terhubung melalui jaringan internal yang diatur oleh Docker Compose.

Struktur file secara ringkas sebagai berikut.

1. Docker-compose.yml



File konfigurasi utama untuk menjalankan semua proses layanan sekaligus.

2. Auth-service/

```
Run Service
auth-service:
  build: ./auth-service
  container_name: auth-service
  depends_on:
    - auth-db
  environment:
    NODE_ENV: development
    PORT: "3001"
    MONGO_URI: mongodb://auth-db:27017/auth
    JWT_SECRET: "ISI_DARI_JWTSECRETS_COM"
  ports:
    - "3001:3001"
  volumes:
    - ./auth-service:/app
    - /app/node_modules
  networks:
    - microservices-network
  deploy:
    resources:
      limits:
        cpus: "0.5"
        memory: 512M
```

Kode dan dockerfile untuk layanan **autentikasi.**

3. Acad-service/

```
Run Service
acad-service:
  build: ./acad-service
  container_name: acad-service
  depends_on:
    - acad-db
  environment:
    PORT: "3002"
    DB_HOST: acad-db
    DB_PORT: "5432"
    DB_NAME: products
    DB_USER: productuser
    DB_PASSWORD: productpass
  ports:
    - "3002:3002"
  networks:
    - microservices-network

networks:
  microservices-network:
    driver: bridge
    ipam:
      config:
        - subnet: 172.16.0.0/28

volumes:
  auth-data:
  acad-data:
```

Kode dan dockerfile untuk layanan **Akademik.**

4. Nginx/

```
Run Service
api-gateway:
  image: nginx:alpine
  container_name: api-gateway
  depends_on:
    - auth-service
    - acad-service
  ports:
    - "8081:80"

  volumes:
    - ./nginx/nginx.conf:/etc/nginx/nginx.conf:ro
  networks:
    - microservices-network
```

Konfigurasi API Gateway menggunakan Nginx.

5. Db/

Skrip inisialisasi database PostgreSQL.

6. Volumes/

Direktori untuk penyimpanan data persistem.

Rancangan struktur ini menempatkan setiap layanan ke dalam area atau domain tersendiri sehingga ketika mengembangkan file ini maka dapat mengerjakan satu layanan tanpa mengganggu layanan yang lain.

2.2 Deskripsi Singkat Tiap Layanan

Project ini mengadopsi arsitektur microservices yang berjalan dalam kontainer terpisah dan setiap layanan memiliki tanggung jawabnya tersendiri. Tiap layanan salingn memiliki koneksi dan saling terhubung melalui jaringan internal yang dikelola oleh Docker-compose.

1. API Gateway

API Gateway dalam arsitektur microservice merupakan komponen terpusat yang berfungsi sebagai reverse proxy serta menjadi titik akses tunggal (single entry point) bagi seluruh permintaan user menuju berbagai microservice pada sisi backend. Pada docker-compose.yml kami API-Gateway berbasis Nginx yang menerima permintaan dari user melalui port yang diekspos (misalnya 8081:80) lalu meneruskanya ke auth-service dan acad-service di dalam jaringan Docker internal. Dengan ini user akan dihadapkan dengan satu endpoint (API Gateway), sementara gateway akan mengatur routing URL, menerapkan keamanan dasar, dan mengerahkan trafik ke container auth-service dan acad-service sesuai dengan path yang ditentukan pada nginx.conf.

Sehingga logika bisa dijalankan tanpa perlu ekspos langsung ke luar.

2. Auth-Service

Auth-Service pada project ini merupakan microservice yang dikhususkan dalam menangani proses autentikasi dan otoritas user, seperti registrasi, logi, maupun penerbitan token akses yang digunakan oleh layanan lain dalam sistem akademik. Layanan ini dijalankan sebagai container terpisah melalui docker-compose.yml, tehubung dengan basis data autentikasi, dan diakses user secara tidak langsung melalui API gateway, dengan begitu setiap permintaan yang dikirimkan ke layanan lain akan diverifikasi identitasnya terlebih dahulu.

3. Acad-Service

Acas-service pada project ini merupakan microservice yang menangani fungsionalitas inti pada sistem akademik, yaitu penyedian data akademik dan perhitungan indeks prestasi studi (IPS) melalui endpoint REST API. Di dalam docker-compose.yml, layanan ini dijalankan sebagai container terpisah, dan tehubung dengan basis data PostgreSQL akademik, dan diakses melalui API gateway. Dengan begitu pola microservice yang membagi aplikasi menjadi serangkaian layanan kecil berbasis kapabilitas yang dapat dideploy secara independen dengan mekanisme komunikasi ringan.

4. Auth-Database

Auth-db pada project ini berfungsi sebagai microservice basis data yang menyipan seluruh informasi terkait auntentikasi dan izin akses pengguna, seperti data akun dan kredensial yang dibutuhkan oleh auth-service. Layanan ini didefinisikan oleh docker-compose.yml yang dijalan di container terpisah dan hanya dapat diakses oleh auth service melalui jaringan internal docker, sesuai praktik pemisahan penyimpanan data setiap layanan pada arsitektur microservices untuk meningkatkan keamanan dan isolasi.

5. Acad-database

Acad-db pada project ini berfungsi sebagai basis data yang dikhususkan untuk menyimpan seluruh informasi yang dibutuhkan oleh acad-service, seperti data mahasiswa, mata kuliah, SKS, maupun komponen lainnya yang diperlukan utuk

perhitungan indeks prestasi strudi (IPS). Layanan ini dijalankan sebagai container database PotsgreSQL terpisah yang didefinisikan di dalam docker-compose.yml dan hanya dapat diakses oleh acas-service melalui jaringan internal docker, sehingga sejalan dengan pola microservice yang memisahkan penyimpanan data setiap layanan untuk menjaga isolasi, skalabilitas, dan keamanan.

BAB III
PERUBAHAN DAN MODIFIKASI

3.1 Rangkuman Perubahan yang Dilakukan

Pada berkas *docker-compose.yml*, dilakukan beberapa penyesuaian, antara lain dengan menambahkan *api-gateway* berbasis Nginx sebagai gerbang utama aplikasi. Aplikasi juga dibagi ke dalam beberapa layanan terpisah, yaitu *auth-service* dan *acad-service*, guna memisahkan tanggung jawab masing-masing modul. Selain itu, digunakan dua database yang berbeda, yakni *auth-db* dan *acad-db*, untuk mendukung kebutuhan tiap layanan.

Seluruh container dihubungkan melalui sebuah network khusus agar dapat saling berkomunikasi dengan baik. Volume juga ditambahkan untuk memastikan data database tetap tersimpan secara persisten. Urutan eksekusi antar service diatur menggunakan *depends_on*, sehingga service dapat berjalan sesuai dependensinya. Terakhir, file **SQL** disertakan agar proses pembuatan database dapat dilakukan secara otomatis saat container dijalankan.

3.2 Alasan dan Detail Setiap Perubahan

- API Gateway

  API Gateway berfungsi sebagai satu titik akses bagi pengguna, sehingga seluruh permintaan hanya perlu dikirim ke satu alamat. Setiap request yang masuk akan diproses terlebih dahulu oleh api-gateway, kemudian diteruskan ke service yang sesuai. Pendekatan ini membuat arsitektur sistem lebih terstruktur serta mempermudah pengelolaan akses antar service

- Pemisahan Service (auth-service dan acad-service)

  Masing-masing service dipisahkan karena memiliki peran yang berbeda. Auth-service bertanggung jawab atas proses autentikasi dan login pengguna, sementara acad-service menangani pengelolaan data akademik. Dengan pemisahan ini, pengembangan aplikasi menjadi lebih fleksibel, dan gangguan pada satu service tidak langsung berdampak pada service lainnya.

- Pemisahan Database (auth-db dan acad-db)

Database dibuat terpisah agar data autentikasi pengguna tidak tercampur dengan data akademik. Hal ini meningkatkan keamanan data serta memudahkan pengelolaan database sesuai dengan kebutuhan masing-masing service.

- Penggunaan Network Docker

  Network Docker digunakan untuk memungkinkan seluruh container saling berkomunikasi dalam satu lingkungan yang sama. Dengan adanya network ini, koneksi antar service dapat dilakukan secara internal tanpa harus membuka akses ke luar, sehingga keamanan sistem lebih terjaga.

- Penggunaan Volume untuk Database

  Volume diterapkan untuk menjaga data database tetap tersimpan meskipun container dihentikan, dijalankan ulang, atau dihapus. Dengan demikian, data tidak akan hilang selama volume masih tersedia.

- Penggunaan depends_on

  Fitur depends_on dimanfaatkan untuk mengatur urutan eksekusi service, khususnya memastikan service penting seperti database berjalan terlebih dahulu sebelum service lain dijalankan. Cara ini membantu menghindari error akibat service yang belum siap digunakan.

- Inisialisasi Database Otomatis

  File SQL disertakan untuk melakukan pembuatan dan pengisian database secara otomatis saat container pertama kali dijalankan. Pendekatan ini mempercepat proses setup dan mengurangi kebutuhan konfigurasi manual.

3.3 Modifikasi Lain pada File atau Folder

Selain pada file docker-compose.yml, perubahan juga dilakukan pada file main.py yaitu pada bagian endpoint @app.get("/api/acad/ips"). Bagian tersebut adalah inti dari penugasan yang harus dikerjakan dengan membuat logika untuk menghitung IPS (Indeks Prestasi Mahasiswa) berdasarkan data yang sudah tersedia pada db_kelas.sql.

Bentuk modifikasi pada endpoint @app.get("/api/acad/ips") meliputi:

1. Penambahan parameter input NIM

   Endpoint akan menerima parameter nim melalui query string untuk menentukan

IPS dari mahasiswa yang diinginkan.

2. Pengambilan data akademik dari database

   Sistem akan menjalankan query SQL yang merupakan penggabungan dari table mahasiswa, krs, dan mata_kuliaj untuk mendapatkan data identitas mahasiswa, bobot numerik dan jumlah SKS.

3. Konversi nilai huruf ke nilai angka

   Pemataan nilai huruf ke bobot numerik agar nilai dapat di proses secara sistematis.

4. Perhitungan total SKS dan total bobot nilai

   Total SKS diperoleh dari seluruh mata kliah yang diambil, sedangkan total bobot nilai diperoleh dari hasil perkalian bobot nilai dengan SKS masing masing mata kuliah.

5. Perhitungan IPS mahasiswa

   IPS dihitung menggunakan rumus perbandingan antara total bobot nilai dan total SKS lalu dibulatkan.

6. Penyajian detail perhitungan

   Sistem akan menampilkan rincian perhitungan tiap mata kuliah berupa nilai huruf, SKS, bobot, dan hasil kali SKS dengan hasil bobot untuk memudahkan proses verifikasi.

7. Penanganan kesalahan (error handling)

   Sistem memberikan respons yang sesuai jika data KRS tidak ditemukan, nilai tidak dikenali, atau terjadi kesalahan pada database maupun sistem.

Selain modifikasi pada file main.py, terdapat beberapa berkas dan folder pendukung yang berperan dalam menjalankan sistem, yaitu:

- File SQL (db_kelas.sql)

  Digunakan untuk pembuatan struktur tabel dan pengisian data awal mahasiswa, mata kuliah, nilai, dan bobot.

- File nginx.conf

  Digunakan untuk mengatur routing request pada API Gateway agar endpoint /api/acad/ips dapat diakses melalui satu pintu.

3.4 Dampak Perubahan terhadap Cara Kerja Sistem

Perubahan pada endpoint /api/acad/ips memberikan dampak langsung pada cara kerja sistem, karena endpoint merupakan fitur utama pada Acad Service yang nantinya akan

menjalankan perhitungan IPS mahasiswa. Dampaknya bisa dijelaskan dalam alur kerja berikut:

3.4.1 Dampak terhadap Proses Perhitungan IPS

Dengan adanya logika perhitungan pada endpoint /api/acad/ips, sistem kini mampu menerima input NIM dari client, mengambil data nilai dan SKS dari database, mengonversi nilai huruf menjadi bobot numerik dan menghitung IPS (Indeks Prestasi Semester). Setelah modifikasi, sistem berjalan sebagai berikut:

1. Client mengirimkan request ke API Gateway
2. API Gateway bertindak sebagi penghubung dan meneruskan request ke Acad Service sehingga client tidak perlu akses port internal acad-service secara langsung.
3. Acad Service membaca input NIM diambil dari query parameter.
4. Acad Service mengambil data dari database PostgreSQL.
5. Acad Service menghitung IPS
6. Acad Service menampilkan output kepada client

3.4.2 Dampak terhadap integrasi microservice

Dengan endpoint IPS yang sudah berfungsi, sistem microservice menjadi lengkap karena Auth Service menangani autentikasi, Acad Service menangani logika akademik (IPS), API Gateway menyatukan akses keduanya sehingga sistem menjalankan fungsionalitas sesuai kebutuhan, tidak sekedar “running container”.

3.4.3 Dampak terhadap pengujian dan validasi

Karena endpoint IPS menghasilkan output yang jelas dan terstruktur, proses pengujian menjadi lebih mudah. Client dapat melihat hasil IPS dari NIM yang berbeda dan juga error handling memastikan sistem memberikan respons yang sesuai jika data tidak ada atau input tidak valid.

3.4.4 Dampak terhadap stabilitas sistem

Dengan penambahan validasi dan error handling membuat sistem lebih stabil, karena sistem tidak langsung crash ketika data tidak ditemukan, kesalahan input diinformasikan dengan jelas dan gangguan koneksi database atau error internal ditangani dengan pesan error.

BAB IV
PENUTUP

4.1 Kesimpulan

Berdasarkan perancangan, implementasi, dan modifikasi yang telah dilakukan, dapat disimpulkan bahwa penerapan arsitektur microservices berbasis Docker pada sistem akademik mampu meningkatkan modularitas, fleksibilitas, dan kemudahan pengelolaan aplikasi. Pemisahan layanan ke dalam auth-service dan acad-service, penggunaan API Gateway sebagai single entry point, serta pemisahan database untuk masing-masing layanan berhasil menciptakan sistem yang lebih terstruktur, aman, dan mudah dikembangkan. Modifikasi pada endpoint perhitungan IPS di Acad Service menjadi inti fungsional sistem, karena memungkinkan proses pengolahan data akademik berjalan secara otomatis, terintegrasi, dan tervalidasi dengan baik. Selain itu, penggunaan Docker Network, volume, depends_on, serta inisialisasi database otomatis turut mendukung stabilitas, konsistensi, dan kemudahan deployment sistem. Secara keseluruhan, proyek ini menunjukkan bahwa pendekatan microservices dengan Docker merupakan solusi yang efektif dan relevan untuk pengembangan sistem akademik yang skalabel, terintegrasi, dan siap dikembangkan lebih lanjut sesuai kebutuhan institusi.

DAFTAR PUSTAKA

Dockerdocs. (2024). Docker Compose. Diambil kembali dari https://docs.docker.com/compose/

Dockerdocs. (2024). Networking overview. Diambil kembali dari https://docs.docker.com/engine/network/

Dockerdocs. (2024). Volumes. Diambil kembali dari https://docs.docker.com/engine/storage/volumes/

Felipe Osses, G. M. (2018). An Exploratory Study of Academic Architectural Tactics and Patterns in Microservices: A systematic literature review. Diambil kembali dari https://www.researchgate.net/profile/Gaston-Marquez-2/publication/331022065_An_Exploratory_Study_of_Academic_Architectural_Tactics_and_Patterns_in_Microservices_A_systematic_literature_review/links/5c619bdb92851c48a9cd3334/An-Exploratory-Study-of-Academic

Murilo Góes de Almeida, E. D. (2022, Maret 16). Authentication and Authorization in Microservices Architecture: A Systematic Literature Review. Diambil kembali dari https://www.mdpi.com/2076-3417/12/6/3023

Sanjay Gadge, V. K. (2017). Microservice Architecture: API Gateway Considerations. Diambil kembali dari https://mainlab.cs.ccu.edu.tw/presentation/pdf/(2017)Microservice-Architecture-API-Gateway-Considerations.pdf

Thönes, J. (2015, Februari). Microservices. Diambil kembali dari https://ieeexplore.ieee.org/abstract/document/7030212