IMPLEMENTASI ARSITEKTUR MICROSERVICES PADA SISTEM INFORMASI AKADEMIK STMIK JAKARTA STI&K MENGGUNAKAN MODEL ENTERPRISE JAVABEANS (EJB) DAN POLYMER JS.

Febianto Arifien[1], Rozi[1] dan Edy Sutomo[2]
(1)STMIK Jakarta STI&K
Jl. BRI No.17, Radio Dalam, Kebayoran Baru, Jakarta Selatan 12140
(2)Universitas Gunadarma
Jl. Margonda Raya No. 100, Depok, Jawa Barat 16424
{febianto, rozi}@jak-stik.ac.id, edysutomo@staff.gunadarma.ac.id

ABSTRAK

Layanan mikroservices baru-baru ini muncul sebagai gaya arsitektur, membahas cara membangun, mengelola, dan mengembangkan arsitektur dari unit kecil yang berdiri sendiri. Masalah pada STMIK Jakarta STI&K yakni kebutuhan integrasi yang mengandalkan interoperabilitas antar beberapa lingkungan heterogen dari sistem yang sedang berjalan dan kendala saat pelaporan melalui aplikasi Feeder PDDIKTI, maka perlu dilakukan pengembangan teknologi perangkat lunak yang bisa dilakukan secara dinamis agar memungkinkan untuk sistem dibangun dengan teknologi yang berbeda-beda, dan memudahkan pekerjaan integrasi data. Microservice menjadi sebuah tren yang digunakan oleh pengembang dalam beberapa tahun terakhir. Tujuan penelitian ini adalah membuat model arsitektur microservice, sebagai solusi untuk meminimalisasi pekerjaan perubahan arsitektur perangkat lunak untuk memenuhi kebutuhan pengembangan yang tinggi. Model yang dikembangkan dengan menggunakan arsitektur microservice smart secure web sevice yakni: membangun model REpresentational State Transfer – Application Programing Interface (REST-API) dan didukung oleh Enterprise Java Bean (EJB). Berdasarkan hasil pengujian, pelaporan dan aktivitas di STMIK Jakarta STI&K menjadi lebih efisien karena prosesnya sudah tersintegrasi sistem dan penambahan aplikasi atau layanan dapat langsung membuat baru atau memodifikasi yang telah ada tanpa mempengaruhi layanan lainnya. Dengan demikian dapat disimpulkan bahwa dengan menggunakan arsitektur microservice sangatlah efektif dalam pengembangan perangkat lunak untuk memenuhi lingkungan sistem yang dinamis dan heterogen.

Kata Kunci : *Interoperabilitas, Microservices, EJB (Enterprise JavaBeans), Polymer Js*

PENDAHULUAN

Pengembangan perangkat lunak berbasis web membutuhkan interoperabilitas di lingkungan yang heterogen, dilihat dari sistem operasi, perangkat lunak, bahasa pemrograman, dan basis data, sehingga mereka dapat berkomunikasi satu sama lain dan bertukar data atau informasi. Sejak awal, pengembangan perangkat lunak telah dilanda tantangan terkait dengan pengiriman yang tepat waktu, kesesuaian fitur, dan kualitas hasil. Layanan *mikroservices* baru-baru ini muncul sebagai gaya arsitektur, membahas cara membangun, mengelola, dan mengembangkan arsitektur dari unit kecil yang berdiri sendiri. Khususnya di *cloud*, pendekatan arsitektur layanan mikro tampaknya menjadi pelengkap yang ideal dari teknologi kontainer di tingkat *Platform as a Service* (PaaS), layanan *Cloud* yang disediakan dalam bentuk platform yang dapat dimanfaatkan pengguna untuk membuat aplikasi diatasnya[1].

Microservice menjadi sebuah tren yang digunakan oleh pengembang dalam beberapa tahun terakhir. *Microservice* memberikan keleluasaan bagi pengembang untuk mengembangkan perangkat lunak dalam waktu yang cepat. Berkembangnya arsitektur *microservice* juga didukung oleh kurang handalnya arsitektur monolitik dalam menangani system failure [2]. Karena dalam satu aplikasi memiliki satu *codebase* yang sama, merupakan hal yang pasti bahwa arsitektur monolitik juga menjadi *single point of failure* yang akan terjadi jika salah layanan terjadi malfungsi atau error . Keunggulan arsitektur *microservice* lainnya adalah perubahan

yang terjadi pada satu layanan dalam frekuensi yang tinggi akan tetap membuat sistem secara keseluruhan bekerja dengan baik tanpa adanya perubahan yang signifikan. Sementara itu arsitektur *microservice* memungkinkan untuk sistem dibangun dengan teknologi yang berbeda-beda. Tujuannya adalah untuk menyesuaikan teknologi dengan kebutuhan pada suatu layanan. Keunggulan arsitektur *microservice* juga ada *poin scale*. Pada arsitektur *microservice*, perubahan dapat dilakukan pada komponen yang membutuhkan saja, tidak perlu sistem secara menyeluruh[3].

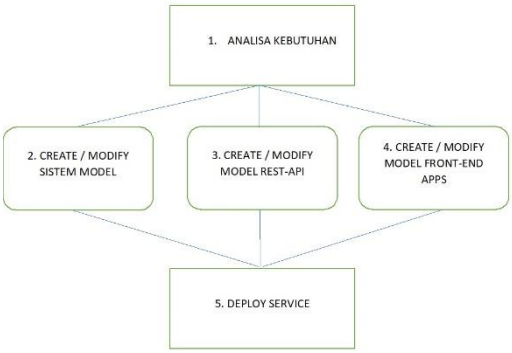
STMIK Jakarta STI&K merupakan salah satu institusi yang terus mengembangkan konsep arsitektur perangkat lunak sesuai dengan kebutuhan eksternal dan internal. Faktor kebutuhan internal didasari pada perubahan di sisi sistem yang sudah berjalan di STMIK Jakarta STI&K yang dulunya masih menggunakan sistem under DOS, sehingga diperlukan strategi perubahan yang efektif dari sistem berbasis offline menuju sistem berbasis web. Kewajiban institusi untuk melaporkan setiap kegiatan aktivitas perkuliahan dengan aplikasi Feeder PDDIKTI adalah salah satunya, permasalahan yang muncul ketika ketidaksincronnya atribut data dan sistem pengiriman data menggunakan model web service menjadi kendala. Hal ini terjadi akibat perbedaan bahasa pemrograman, sistem web service antara model SOAP dan REST-API, database yang berbeda dan lain-lain [4].

Berdasarkan kebutuhan integrasi yang mengandalkan interoperabilitas data maka perlu dilakukan pengembangan teknologi web untuk memudahkan pekerjaan integrasi data dari berbagai instansi, sebagai contoh pada instansi pendidikan di perguruan tinggi. Data akademik yang terdiri dari data mahasiswa, mata kuliah, aktivitas mengajar dosen, riwayat status mahasiswa, aktivitas kuliah mahasiswa dan nilai semester yang sangat banyak menjadi kendala saat pengisian melalui aplikasi Feeder PDDIKTI [4]. Tujuan penelitian ini adalah membuat model arsitektur *microservice*, sebagai solusi untuk meminimalisasi pekerjaan perubahan arsitektur perangkat lunak untuk memenuhi kebutuhan pengembangan yang tinggi dari lingkungan intenal dan lingkungan eksternal, seperti yang telah disebutkan sebelumnya.

METODE PENELITIAN

Implementasi arsitektur microservice yang diterapkan pada sistem informasi akademik pada STMIK Jakarta STI&K adalah mengembangkan model perangkat lunak dengan memanfaatkan teknologi arsitektur *microservices* yang menggunakan sistem *smart secure web services* dengan tiga level. Ketiga level tersebut yakni: membangun model *REpresentational State Transfer – Application Programing Interface* (REST-API) dan didukung oleh *Enterprise Java Bean* (EJB) untuk level teknis, membuat model mapping database menggunakan metode semantik ontologi yang secara otomatis menyesuaikan struktur dan prosedur dari model klien untuk level semantik, selanjutnya level terakhir yaitu mengembangkan yang disebut *model smart secure*, yakni membuat otomatisasi dari seluruh proses integrasi dan menambahkan model pengambilan keputusan dengan metode *deep learning* dari keseluruhan proses serta mengimplementasikan model enkripsi untuk pengamanan transfer data [5].

Metode penelitian yang difokuskan pada pembuatan sistem model dilanjutkan dengan pembuatan *web service* REST-API, terakhir membuat antarmuka *sebagai front-end services* menggunakan pustaka polymer js. Bagan tahapan penelitian seperti pada Gambar 1.



Gambar 1. *Bagan Tahapan Penelitian*

I. Analisa Kebutuhan

Tujuan dari tahap analisis kebutuhan adalah memahami dengan sesungguhnya kebutuhan dari sistem yang baru berdasarkan sistem yang sudah berjalan dan mengembangkan sebuah sistem yang memadai kebutuhan tersebut atau memutuskan bahwa pengembangan sistem yang lama tidak dibutuhkan.

II. Create / Modify Sistem Model

Langkah kedua pembuatan aplikasi, dibangun komponen modul INTF, modul EJB dan modul EAR. Langkah-langkah tersebut dilakukan dua kali, yang pertama membangun modul SYSTEM EAR, berfungsi sebagai data akses dan yang kedua membangun WEB API EAR, berfungsi sebagai bisnis akses. Modul EJB pada SYSTEM EAR terkoneksi *datastore* yaitu gudang penyimpan data untuk mengelola koleksi data yang terkoneksi menggunakan *JDBC Driver* dengan fungsi-fungsi SQL[4].

III. Create / Modify Model REST-API

Modul REST-API merupakan layanan berdasarkan arsitektur RESTful Java dengan JAX-RS yang kerjanya mendisain antarmuka dengan mengirim perintah HTTP di JAX-RS. Pada WEB API EAR mengimplementasi teknologi DTO yaitu kumpulan data berdasarkan dari *Model Class* yang tidak memiliki validasi, tidak ada logika bisnis, tidak ada logika apapun. Ini sangat sederhana, ringan dan hanya dijadikan wadah untuk memindahkan data. Fungsinya dijadikan sebagai wadah untuk memindahkan data dari DAL (*data access layer*) ke BAL (*bussiness access layer*) atau antara lapisan dimasing-masing dalam arsitektur *n-tier*[4].

IV. Create / Modify Model Front-End Apps

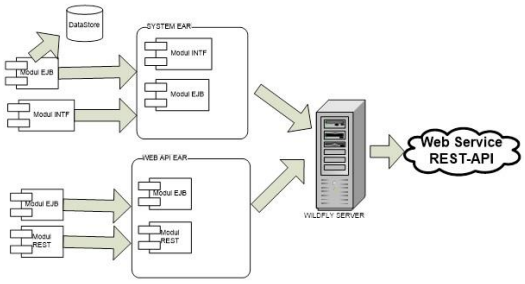
Model Front-End aplikasi berusaha mengembangkan aplikasi lintas platform untuk mengatasi permasalahan adanya beberapa platform yang berbeda.Penggunaan *Progressive Web App* (PWA) yang menggabungkan hal terbaik dari web dengan fitur yang telah dimiliki oleh aplikasi native. Polymer adalah PWA terbaik yang dikembangkan oleh Google. Polymer memberikan standar web yang berbeda (sebagai contoh: web komponen, variabel CSS) yang membantu menjalankan aplikasi di browser dengan sangat cepat [6].

V. Deploy Service

Tahapan ini merupakan proses mendeploy file .ear ke server wildfly. Proses ini dapat dilakukan dengan beberapa cara:

a. Meng*copy* file .ear ke system file dan melakukan konfigurasi pada file standalone.
b. Menggunakan antarmuka manajemen (konsol admin atau CLI)
c. Menggunakan Maven untuk mendeploy aplikasi ke server.

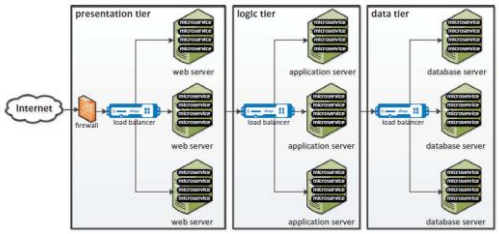
Gambar 2 merupakan skema diagram komponen aplikasi REST-API secara global[4].



Gambar 2.*komponen aplikasi REST-API secara global*

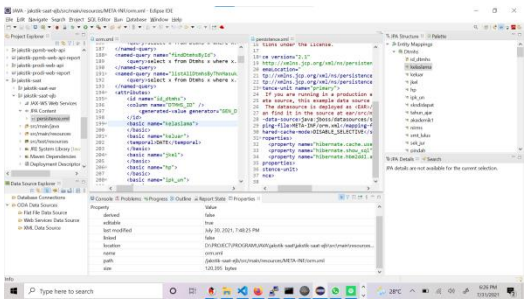
Penggambaran arsitektur *microservices*, semua permintaan klien berdasarkan analisa kebutuhan diarahkan dan kemudian diteruskan ke microservice sesuai dengan titik akhir yang ditentukan. Setelah request dari *client* sesuai dengan ketentuan, *microservice* mengirimkan respon yang diarahkan kembali ke API gateway, kemudian diteruskan ke client. Setiap *microservice* memiliki endpoint tersendiri sesuai dengan tujuannya, sehingga *API Gateway* memuat semua endpoint yang mengarah ke semua *microservice* yang ada. Selain kecepatan dalam hal pengembangan, skalabilitas operasional juga diidentifikasi sebagai manfaat potensial utama dari arsitektur *microservice*. Mengingat adopsi

infrastruktur *cloud* di mana-mana dan layanan *platform, scaling out* (menambahkan server virtual tambahan) lebih disukai untuk meningkatkan (menambahkan komponen tambahan seperti RAM dan CPU ke server yang ada). Perbedaan level *tier* (*data tier*, *logic tier* dan *presentation tier*) memungkinkan perubahan atau modifikasi pada setiap tingkatan secara terpisah, hal ini dipandang sangat fleksibel dimana setiap tingkat mungkin memiliki skala yang berbeda-beda, bisa saja satu tingakatan memuat banyak layanan seperti yang terlihat apada gambar 3[7].



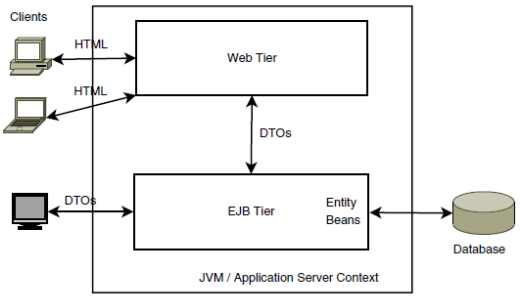
Gambar 3.*Penggambaran arsitektur microservice*

Data Tier menangani data persisten dengan mengaitkan ujung depan ke sumber data tempat konten yang akan dipublikasikan dan dimanipulasi menggunakan model JPA-ORM (Gambar 4).



Gambar 4.*JPA-ORM pada Eclipse IDE*

Gambar 4 Mengilustrasikan model JPA-ORM yang merupakan sumber data asli yang mencakup sistem apa pun yang dapat diakses melalui JDBC/ODBC. Platform penyimpanan data tambahan seperti database SQL, repositori XML, dapat ditambahkan dengan memprogram dan mengimpor serta untuk menghubungkan ke sumber data tersebut. Ketika aplikasi terikat ke beberapa persisten, aplikasi dapat digunakan untuk memeriksa keselarasan antara model domain dan database fisik untuk merekonsiliasi perubahan yang dibuat dalam model dengan database. Semua kelas dalam model domain dan elemen turunannya harus dipetakan dengan benar sebelum menjalankan aplikasi. Jika tidak, dapat menghasilkan hasil yang tidak lengkap.



Gambar 5.*Konsep DTO (Data Transfer Object)*

Gambar 5 menunjukkan konsep DTO (*Data Transfer Object*) digunakan pada *Business Logic / Logic Tier. Data Transfer Object* (DTO) adalah pola desain yang umum digunakan dalam sistem terdistribusi pada umumnya dan aplikasi *Enterprise Java* pada khususnya. Setiap pemanggilan metode yang dibuat ke objek bisnis dalam sistem menggunakan jaringan lapisan terlepas dari kedekatan klien ke server, menciptakan *overhead* jaringan [8].

Presentation layer merupakan Interface yang akan ditampilkan ke *client.* Berisi berbagai macam konten dan informasi yang dibutuhkan oleh *user* dan dapat diakses melalui software atau *web-browser*. Disinilah para *front-end programmer* bekerja, dengan membuat berbagai gamabaran konten dan Interface yang menarik serta *responsive* untuk diakses oleh semua *platform*. Dalam pembuatannya biasanya menggunakan HTML5, Javascript, CSS atau beberapa *framework* yang populer digunakan oleh *front-end programmer*[9].

Keuntungan utama dari *microservice* adalah karena keunikan setiap properti sehingga memungkinkan untuk

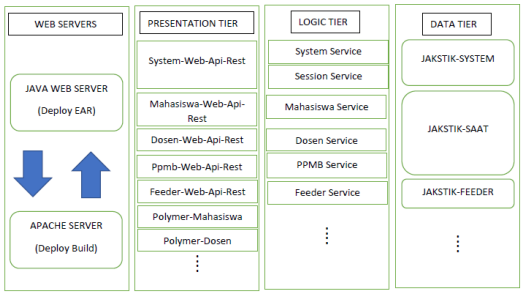
dieksploitasi dengan lebih baik, dengan kemudahan dan kemungkinan penerapan dan pengelolaan secara mandiri dalam aplikasi [10].

HASIL DAN PEMBAHASAN

Implementasi design dari konsep *microservice* yang diterapkan pada sistem informasi akademik STMIK Jakarta STI&K terdiri dari *Data Tier*, *Logic Tier*, *Presentation Tier* dan *Web Servers*, yang terlihat pada Gambar 6.

Satu layanan (single service), terdiri dari *Data Tier, Logic Tier dan Presentation Tier*, yang kemudian dideploy ke *web server*. Dengan menggunakan konsep microservices ini, memungkinkan untuk men-*deploy single service* ke lebih dari satu server sehingga apabila terjadi kerusakan pada satu web server, maka layanan lain tetap bisa diakses.

Web server yang telah dibuat di STMIK Jakarta STI&K sementara ada 2 yakni Java Web Server yang memuat layanan REST-API Gateway dan Apache Server yang memuat Interface Polymer Js.

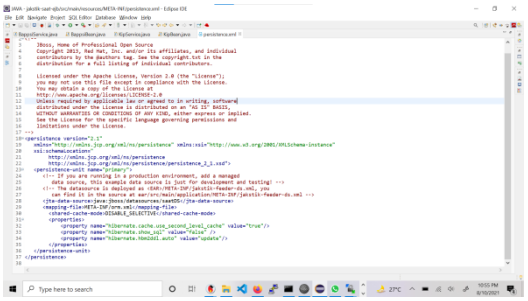


Gambar 6.*Penggambaran arsitektur microservice STMIK Jakarta STI&K*

Pada Gambar 6 ditunjukkan bahwa arsitektur microservice di STMIK Jakarta STI&K dibangun berdasarkan 3 tingkatan Tier antara lain: Data Tier, Logic Tier dan Presentation Tier. Ketiga Tier tersebut dideploy pada masing-masing web server.

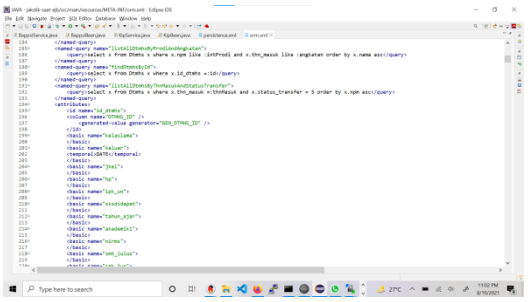
Data Tier

Pembuatan Model *JPA-ORM*, diawali dengan membuat *datasource* pada unit persistent untuk mengakses database dengan bantuan koneksi JDBC/ODBC. Konfigurasi unit persistent pada konten JPA sebagai berikut:



Gambar 7.*Persistence Unit pada JPA-ORM*

Keterangan yang penting pada konfigurasi *persistence unit* adalah nama jta-data-source yakni java:jboss/datasources/saatDS dan saatDS sebagai nama datasource yang nantinya diperlukan untuk konfigurasi di sisi server java-nya.



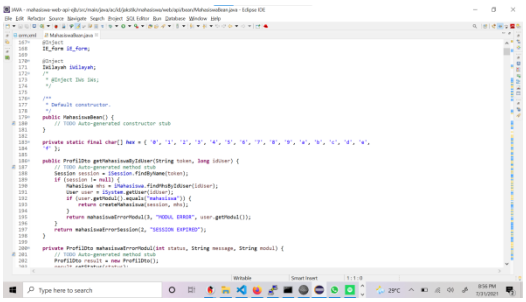
Gambar 8.*Entity Mapping pada JPA-ORM*

Hasil *entity mapping* ditunjukkan oleh Gambar 8. Dengan pembuatan entity kelas model mahasiswa terbukti dapat terkoneksi dengan database mahasiswa (Gambar 10) dimana setiap perubahan pada model domain pada kelas java mahasiswa dapat langsung merubah domain database fisik. Pada setiap perubahan yang dibuat pada aplikasi langsung dieksekusi pada database fisik. Dengan demikian dapat dinyatakan bahwa *entity mapping* sangat efektif untuk model pengembangan pada level data tier.

Logic Tier

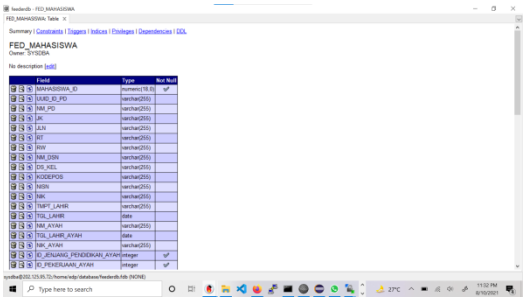
Gambar 9. merupakan hasil dari pengembangan logic tier menggunakan konsep DTO (Data Transfer Object) dimana dicontohkan pada proses bisnis layanan mahasiswa koneksi dari tabel mahasiswa di tampilkan dalam bentuk profilDTO. Pada sistem, kriteria utama perancangan dan implementasi *Business Logic Layer* biasa

menggunakan EJB. Pengembangan EJB harus mencapai definisi antarmuka *presentation tier* berdasarkan paket javax.ejb.



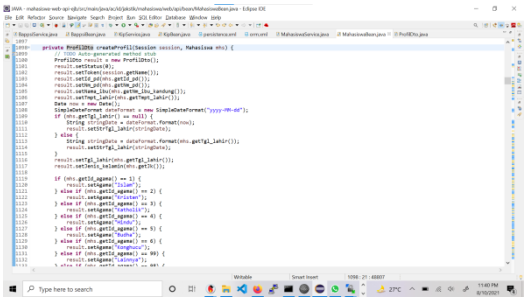
Gambar 9. Source Code*Penggambaran EJB pada Eclipse IDE*

Penerapan *Logic Tier* pada arsitektur microservice STMIK Jakarta STI&K menggunakan konsep DTO, menghasilkan / memberi bentuk penyederhanaan proses, sehingga atribut yang digunakan dapat diubah sesuai kebutuhan melalui penambahan layer DTO, sehingga *query data* yang ditampilkan pada presentation tier lebih efisien sesuai kebutuhan.



Gambar 10. *Database Mahasiswa*

Sebagai contoh pada gambar 10, untuk menampilkan data mahasiswa yang terdiri dari 55 atribut seperti nama, npm, nisn, nik dan lain-lain, diterapkan model DTO sebagai layer model untuk menampilkan atribut sesuai kebutuhan saja yakni dalam bentuk ProfilDTO seperti yang terlihat pada gambar 11.

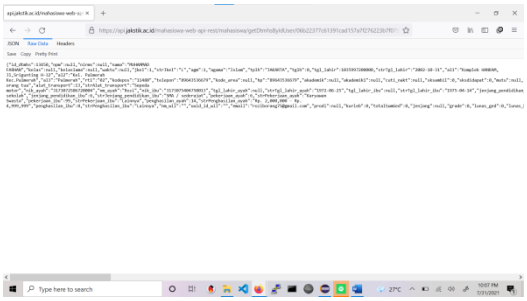


Gambar 11. *ProfilDTO pada Logic Tier*

EJB-service pada logic tier dibuat menyesuaikan kebutuhan REST-API service, sehingga satu aplikasi REST-API client pada presentation tier menghasilkan satu EJB-service pada logic-tier.

Presentation Tier

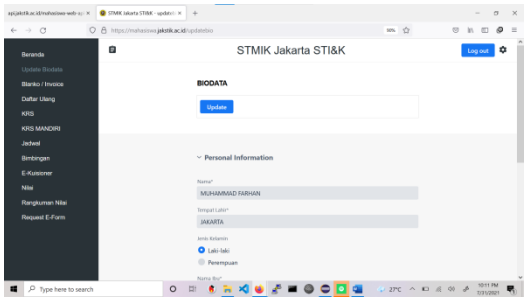
Presentation tier yang dideploy pada java server client adalah *bentuk REST-API Gateway* yang dipanggil oleh *application web* dari *Polymer application* untuk berinteraksi CRUD (*Create, Read, Update dan Delete*) dari database sistem.



Gambar 12. *Mahasiswa REST-API client*

Gambar 12 merupakan hasil uji coba *bentuk REST-API Gateway* dari Mahasiswa REST-API yang bisa menampilkan data mahasiswa sesuai dengan kebutuhan. API Gateway ini dikoneksikan dengan User Interface dalam bentuk aplikasi-aplikasi Front-End.

Bentuk hasil ujicoba apliaksi *Front-End* menggunakan model Polymer Js, seperti perlihat pada Gambar 13.

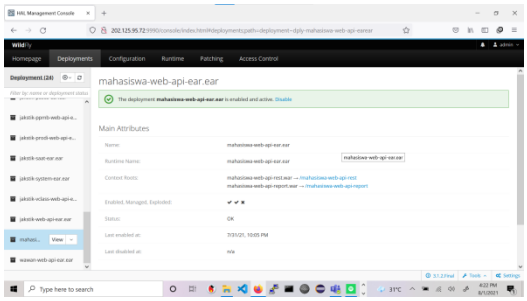


Gambar 13.*Bentuk Aplikasi Mahasiswa menggunakan Polymer Js.*

Seperti yang terlihat pada Gambar 13, bahwa API Gateway mahasiswa dapat terkoneksi dengan user interface aplikasi front-end mahasiswa menggunakan polymer js.

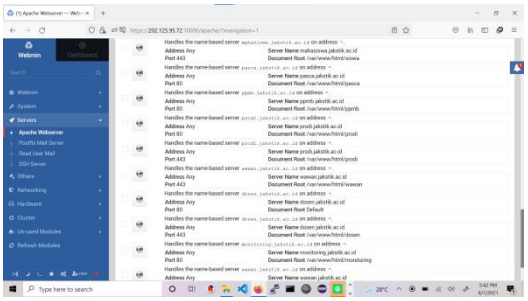
Berdasarkan hasil uji coba tersebut, maka penggunaan aplikasi mahasiswa merupakan salah satu layanan microservice yang dipakai untuk aktivitas mahasiswa STMIK Jakarta STI&K seperti daftar ulang, isi krs, lihat log bimbingan, lihat nilai dan lain-lain.

Berikut hasil *microservices* yang dideploy ke *wildfly server*, seperti yang terlihat pada Gambar 14.



Gambar 14.*Server Wildfly STMIK Jakarta STI&K*

Saat ini layanan yang telah dibuat sejumlah 24 layanan microservice seperti layanan dosen, layanan program studi, layanan pmb dan lain-lain. Untuk *front-end* menggunakan model polymer js, seperti yang terlihat pada Gambar 14.



Gambar 15.*Server Apache STMIK Jakarta STI&K dari Polymer Js*

Layanan build antarmuka menggunakan model Polymer Js, dideploy pada server *Apache service*. Bentuk layanan antarmuka ini memungkinkan pengguna mengakses setiap layanan microservice STMIK Jakarta STI&K dalam sebuah aplikasi yang lebih mudah digunakan dan sesuai kebutuhan penggunanya.

PENUTUP

Berdasarkan pada hasil pengujian terbukti dapat meningkatkan layanan, seperti contoh pada layanan mahasiswa sebelum dilakukan pengembangan tidak dapat melakukan aktivitas mahasiswa seperti daftar ulang isi krs dan lain-lain secara online, namun dengan adanya pengembangan bentuk arsitektur microservice, aktivitas mahasiswa dapat dilakukan dengan cara online dan tidak perlu merombak sistem yang sedang berjalan serta tidak memerlukan waktu yang lama untuk menambahkan dan merubah layanan yang telah ada. Dengan demikian dapat disimpulkan bahwa bentuk arsitektur microservice bisa menjadi pilihan untuk proses pengembangan sistem dalam lingkungan yang dinamis dan heterogen secara efektif karena permasalahan bisa diselesaikan, efisien penggunaan sumber daya karena tidak merombak struktur yang telah ada, dan fleksibel disebabkan setiap perubahan dapat diakomodasi dengan menambah atau memodiikasi layanan tertentu saja

Beberapa hal yang menjadi perhatian dalam penelitian ini adalah dalam setiap perubahan masih dilakukan secara manual untuk setiap perubahan layanannya, sehingga sebagai bahan pengembangan untuk penelitian selanjutnya adalah

menerapkan automatisasi setiap ada perubahan internal maupun eksternal, sehingga sistem bisa melakukan pembaharuan layanan secara mandiri.

DAFTAR PUSTAKA

[1] P. W. Widya, R. Muslim, B. Adi, and A. K. Awan, "Rancang Bangun Layanan Platform as a Service (PAAS) untuk Mendukung Sistem Multi-Tenancy Pengembangan Aplikasi Berbasis Komputasi Awan," J. Tek. Pomits, vol. 2, no. 1, 2013.

[2] X. Larrucea, I. Santamaria, R. Colomo-palacios, and C. Ebert, "Microservices," 2021.

[3] "Eduvest – Journal of Universal Studies Volume 1 Number 7 , July 2021 IMPLEMENTATION OF MICROSERVICES IN FINTECH APPLICATIONS USING EXPRESSJS FRAMEWORK Bernardus Redika Westama Putra and Evangs Mailoa Satya Wacana Christian University Received : Revised :," vol. 1, no. 7, pp. 559–567, 2021.

[4] F. Arifien and M. Riastuti, "Model Interoperabilitas Web Service Feeder PDDIKTI Menggunakan Enterprise Javabeans (EJB) dan REST-API," vol. 3, 2019.

[5] M. Maleshkova, P. Philipp, Y. Sure-Vetter, and R. Studer, "Smart Web Services (SmartWS) -- The Future of Services on the Web," 2019, [Online]. Available: http://arxiv.org/abs/1902.00910.

[6] F. Ari, K. Kunci, C. Library, U. Interface, U. Experience, and W. Components, "Penerapan Pustaka PolymerJs pada Aplikasi Presensi Dosen," J. Ilm. Komputasi, vol. 19, no. 2, pp. 285–292, 2020, doi: 10.32409/jikstik.19.2.95.

[7] S. Baškarada, V. Nguyen, and A. Koronios, "Architecting Microservices: Practical Opportunities and Challenges," J. Comput. Inf. Syst., vol. 60, no. 5, pp. 428–436, 2020, doi: 10.1080/08874417.2018.1520056.

[8] A. Pantaleev and A. Rountev, "Identifying data transfer objects in EJB applications," Proc. - ICSE 2007 Work. 5th Int. Work. Dyn. Anal. WODA 2007, 2007, doi: 10.1109/WODA.2007.6.

[9] N. Nurwanto, "Penerapan Progressive Web Application (PWA) pada E-Commerce," Techno.Com, vol. 18, no. 3, pp. 227–235, Aug. 2019, doi: 10.33633/tc.v18i3.2400.

[10] J. Soldani, D. A. Tamburri, and W. J. Van Den Heuvel, "The pains and gains of microservices: A Systematic grey literature review," J. Syst. Softw., vol. 146, 2018, doi: 10.1016/j.jss.2018.09.082.