PENGORGANISASIAN KERJA SISTEM PARKIR MENGGUNAKAN ARSITEKTUR *MICROSERVICE*

Uminingsih[1], Surya Dharma Handayani[2]
[1,2]Program studi Rekayasa Sistem Komputer, Institut Sains & Teknologi AKPRIND Yogyakarta
Email: umy_bin@yahoo.com

ABSTRACT

In the past many complex systems (tools) were constructed using monolithic architecture. In this architecture it is described as an application that runs all logic programs in one application server. The problem that arises is that if there is a work disruption from one of the modules, the whole system must be turned off when there is a repair. This results in material and time losses. Microservice architecture is used to overcome this. The advantage is that it can divide a complex system into several small independent service works, so that if there is damage to one of the services, then when repairing it, there is no need to dismantle the entire system. The architecture microservice can connect several services even with different languages. The object in this research was an automatic parking machine which was designed based on microservice architecture. Testing is done when: a) after all services are connected, then each service is checked its work, and it turns out that the results are in accordance with all of them, b) testing is done by interfering with one of its services, after being checked for other services that are not directly related to the service but it is connected into one system, it can still run as usual. It has been proven that the superiority of microservice architecture is a type of the practical and efficient architecture.

Keywords: *Microservice, RFID, work organization, parking system, automatic*

INTISARI

Dimasa sebelumnya banyak konstruksi suatu sistem (sebuah alat) yang komplek menggunakan arsitektur monolitik. Pada arsitektur ini digambarkan sebagai sebuah aplikasi yang menjalankan semua program logika dalam satu server aplikasi. Permasalahan yang timbul adalah bila terjadi gangguan kerjanya dari salah satu modulnya maka sistem secara keseluruhan harus di *off* kan saat terjadi perbaikan. Hal ini menimbulkan kerugian material dan waktu . Untuk mengatasinya digunakan arsitektur *microservice*. Kelebihannya dapat membagi sistem yang komplek menjadi beberapa buah *service* kecil-kecil yang *independent* kerjanya, sehingga bila terjadi kerusakan pada salah satu *service*-nya maka saat memperbaikinya tidak perlu membongkar seluruh sistem yang ada. Dengan arsitektur *microservice* dapat menghubungkan beberapa *service* walaupun dengan bahasa yang berbeda beda. Dalam penelitian ini mengambil obyek mesin parkir otomatis yang di disain kerjanya berdasarkan arsitektur *microservice.* Pengujian dilakukan saat : a) Setelah semua *service* terkoneksi, selanjutnya masing-masing *service* di cek kerjanya, dan ternyata hasilnya sesuai semua. b) Pengujian dilakukan dengan mengganggu pada salah satu *service* nya, setelah di cek untuk *service-service* yang lain yang tidak berkaitan langsung dengan *service* tersebut tapi terkoneksi menjadi satu sistem tetap bisa jalan seperti biasanya.. Dengan ini telah dibuktikan bahwa keunggulan arsitektur *microservice* adalah jenis arsitektur yang praktis dan effisien.

Kata kunci: *Microservice,* RFID, organisasi kerja, sistem parkir

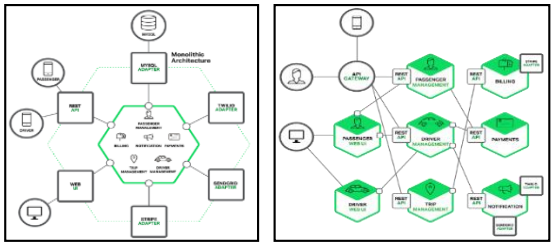
PENDAHULUAN

Perkembangan dan kemajuan pesat dalam bidang mekatronika yang meliputi (bidang *engineering* dan elektronika serta informatika) menyebabkan perubahan besar di bidang teknologi *engineering.* Salah satu contoh adalah pembuatan alat smart yang bersifat otomatis. Pada masa-masa sebelumnya arsitektur dalam perancangan alat yang bersifat komplek (terdiri dari banyak bagian-bagian dengan sistem yang berbeda) menggunakan sistem Arsitektur Monolitik. Prinsipnya arsitektur ini menggambarkan sebuah aplikasi yang menjalankan semua program logika dalam satu server aplikasi [1]. Permasalahan yang timbul adalah bila ada salah satu bagian yang rusak maka sistem harus dihentikan secara keseluruhan saat dilakukan perbaikan. Selain itu bila ingin menambah fitur juga menjadi riskan dapat

menyebabkan *error* pada *service* yang lainnya. Solusinya adalah beralih ke arsitektur *Microservice.* .

Arsitektur *Microservice* adalah sebuah arsitektur dimana pengembangan aplikasi dilakukan dalam bentuk *web service* yang lebih kecil, yang saling berkomunikasi satu sama lain. Arsitektur *microservice* merupakan alternatip arsitektur yang lebih terukur dan *flexible.* Dengan kata lain arsitektur *microservice* merupakan sebuah arsitektur yang memecah aplikasi berdasarkan fungsinya secara spesifik menjadi beberapa *service*. Setiap *service* tersebut dirancang agar bekerja secara independent dan dapat menggunakan teknologi yang berbeda sesuai kebutuhan. Perbedaan arsitektur monolitik dan arsitektur *Microservice* dilukiskan pada Gambar1.

MONOLITIK ***MICROSERVICE***



Gambar 1: Perbedaan arsitektur monolitik dan *microservice*

Alasan kenapa orang beralih memilih arsitektur *Microservice* menurut Newman (2015) menjelaskan bahwa terdapat banyak manfaat dan variasi dari *microservice*. Manfaat tersebut dijabarkan sebagai berikut:

a.*Technology Heterogenity*

Dengan sistem yang terbentuk dari beberapa *service*, developer dapat menentukan penggunaan teknologi untuk setiap *service*-nya. Ini membuat *developer* bisa memilih kumpulan teknologi yang cocok untuk tiap *service* daripada menggunakan teknologi yang bisa digunakan untuk semua namun mempunyai performa rendah. Penggunaan *microservice* juga membuat developer dapat mengadopsi teknologi baru dengan cepat dengan resiko yang minimal.

b. Resiliensi

Penggunaan arsitektur *microservice* membuat aplikasi menjadi lebih resilien terhadap kerusakan/*error* yang terjadi. Kerusakan menjadi mudah dideteksi dan terlokalisasi hanya pada *service* tertentu. Berbeda dengan aplikasi *monolithic* pada umumnya, kerusakan yang terjadi menyebabkan keseluruhan aplikasi akan berhenti bekerja.

c. Scaling

Saat beban server melebihi kapasitas, diperlukan proses *scaling* agar performa server tetap terjaga. Pada aplikasi *monolithic*, *scaling* dilakukan dengan men-*deploy* keseluruhan aplikasi. Sedangkan *scaling* pada *microservice* dapat dilakukan dengan hanya men-*deploy* beberapa *service* yang membutuhkan *scaling* saja. Hal ini membuat proses *scaling* menjadi lebih efisien karena dapat dijalankan pada perangkat yang lebih kecil.

Dengan karakteristik seperti itu maka akibatnya adalah:

a. Membuat kode aplikasi lebih sedikit dan bersifat independen sehingga dapat dilakukan pengujian aplikasi secara independen.
b. Memudahkan melakukan pemeliharaan perangkat lunak sistem aplikasi.
c. Dapat melakukan proses distribusi perangkat lunakn secara independen.
d. Lebih mudah dalam melakukan *scalability*.
e. Desainer/developer dapat bebas dalam memngembangkan aplikasi dengan berbagai bahasa pemrograman dan *framework.*

Hal itu semua tentunya sangat menguntungkan.

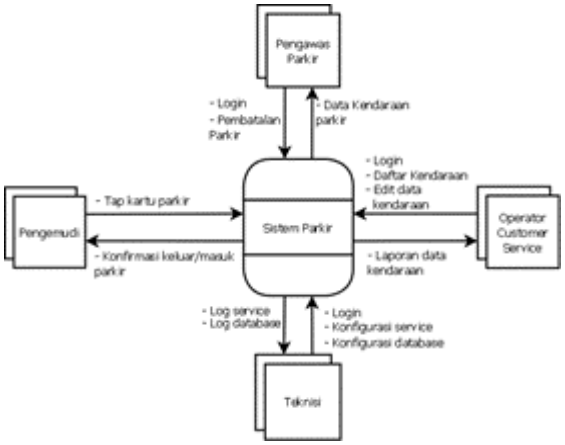
METODE PENELITIAN

Dalam penelitian ini sebagai obyek adalah sistem parkir otomatis yang menggunakan arsitektur *Microservice.* Ada beberapa langkah yang harus di persiapkan yaitu:

a. Pengumpulan data *service*
b. Mengaktipkan kartu parker
c. Mengkoneksikan dengan modul reader
d. Mengkoneksikan dengan modul kontroler,
e. Mengkoneksikan dengan power suplay
f. Pengaturan output.

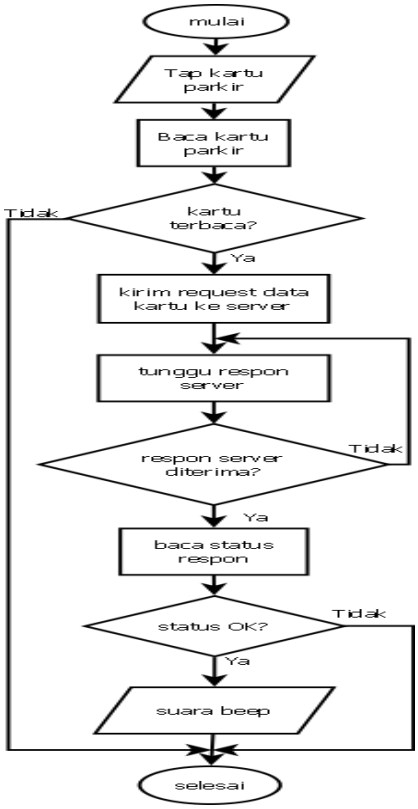
Entitas Sistem Parkir

Sebagai gambaran tentang siapa saja yang meng akses sistem ini dan datanya apa saja serta alurnya kemana saja dari sistem yang di organisir oleh *microservice* dapat dilukiskan dalam Diagram Entitas Akses Sistem Parkir yang dilukiskan pada Gambar 2, untuk pengelola semua *service* di sistem ini dilakukan oleh teknisi.

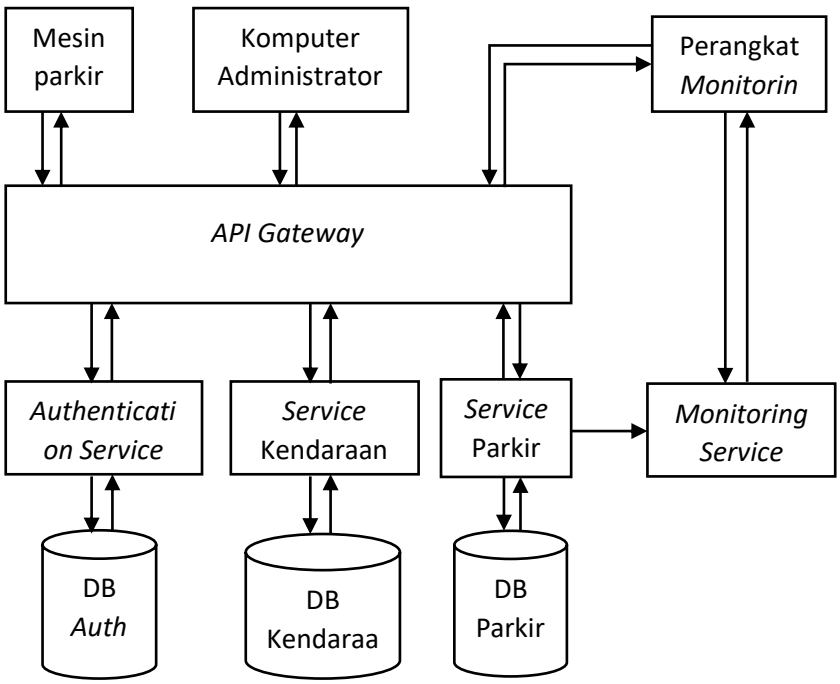


Gambar 2 Diagram Entitas Akses Sistem Parkir

Untuk memudahkan memahami bagaimana cara kerja sistem parkir ini dapat dilukiskan dalam bentuk *flowchart* yang dilukiskan pada Gambar 3, sedangkan dengan menggunakan konsep arsitektur *Microservice* , Sistem Parkir ini dibangun terdiri dari beberapa modul-modul yang di organisir oleh *microservice*, ada yang terhubung melalui web serta ada pula yang melalui konektor biasa, hal ini dapat dilukiskan pada Gambar 4



Gambar 3 Diagram Alir Kerja Mesin Parkir



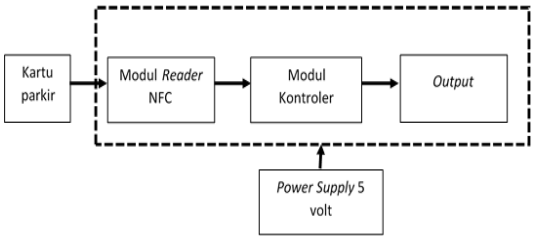
Gambar 4. Blok Diagram Sistem Parkir

Komunikasi dari luar *server* dan komunikasi antar *service* menggunakan API *gateway*. API *gateway* ini digunakan untuk sentralisasi endpoint API setiap *service* sehingga akses dari luar *service* tidak langsung terhubung. API *gateway* akan melakukan *routing* permintaan dari luar *server* ke *service* yang dituju. Sebelum melakukan *routing*, API

gateway akan melakukan autentikasi *request*. Autentikasi dilakukan untuk pengecekan apakah pengguna dapat menggunakan *service* yang dituju. Jika *request* diijinkan maka API *gateway* akan meneruskan *request* tersebut, sebaliknya jika tidak maka API *gateway* akan menolak meneruskan *request* dan memberikan respon status HTTP 401 *Not Authorized*. Dengan menggunakan API *gateway* ini diharapkan dapat meningkatkan keamanan sistem parkir ini dari pihak-pihak yang tidak bertanggung jawab.

Modul -Modul *Service* Pendukung

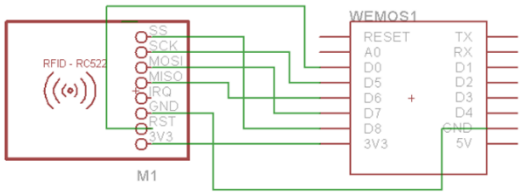
Modul *service* yang mendukung sistem parkir ini dapat dilukiskan: pada Gambar: 5



Gambar 5. Lima *service* pokok pendukung sistem parkir

Modul Reader

Blok diagram mesin parkir yang dirancang terdapat tiga bagian utama, yaitu modul reader NFC, berfungsi untuk pembacaan kartu parkir seperti pada Gambar 6.



Gambar 6. Antarmuka MFRC522 dengan WeMos D1 Mini

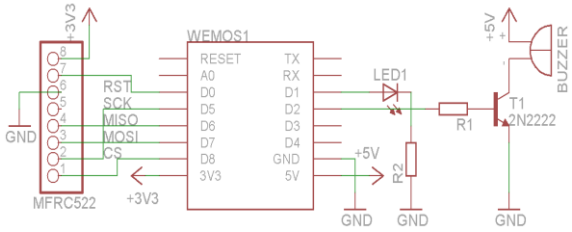
Modul Kontroler

Modul kontroler menggunakan modul WeMos D1 *Mini* yang berbasis ESP8266. Modul ini telah terintegrasi dengan modul komunikasi WiFi sehingga tidak memerlukan modul khusus untuk WiFi.

Blok Output

Blok *output* digunakan sebagai indikator saat mesin parkir bekerja. Dalam penelitian ini, komponen yang digunakan adalah *light emiting diode* (LED) dan *buzzer*.

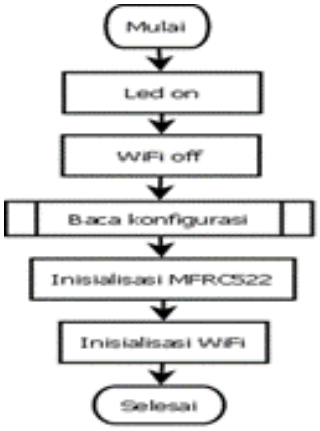
LED digunakan sebagai indikator kondisi *on/off* mesin parkir serta kondisi *connect/disconnect* dengan jaringan WiFi. Led menyala saat mesin parkir dalam kondisi on dan sebaliknya akan mati saat kondisi off. Saat mesin parkir tidak terhubung dengan jaringan WiFi, led akan berkedip (menyala dan mati secara bergantian) hingga mesin kembali terkoneksi dengan jaringan. *Buzzer* digunakan sebagai indikator berhasil tidaknya pembacaan kartu parkir. Saat pembacaan kartu parkir berhasil dan respon *server* untuk keluar/masuk parkir berhasil maka *buzzer* mengeluarkan bunyi beep satu kali. Jika proses pembacaan parkir atau respon server gagal maka *buzzer* tidak mengeluarkan bunyi. Hal ini dilukiskan pada Gambar 7.



Gambar 7 Skema Rangkaian Mesin Parkir

Modul *Firmware*

Untuk dapat dignakan pada perancangan ini, penulisan kode program untuk *firmware* menggunakan *framework* Arduino sehingga format kode mengikuti *framework* tersebut.. Diagram alir blok fungsi *setup*() dapat dilihat pada Gambar 8



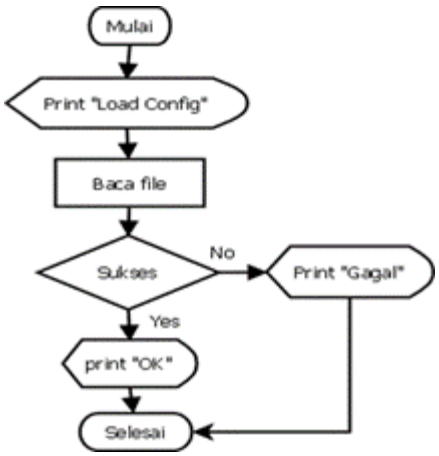
Gambar 8 Diagram Alir Blok Fungsi *setup*()

Dalam Arduino terdapat dua blok fungsi, yaitu *setup*() dan *loop*(). Blok fungsi *setup*() berjalan

pada awal program dan hanya dijalankan satu kali setiap *power up* atau *reset*. Digunakan untuk inisialisasi variabel, set mode pin, penggunaan pustaka, dan lain-lain. Sedangkan *loop*() merupakan inti program yang dijalankan berulang-ulang hingga perangkat dimatikan. Blok fungsi *loop*() berjalan segera setelah blok fungsi *setup*() selesai dijalankan

Modul baca Konfigurasi

Dari diagram alir pada Gambar 9 penulis membuat sub rutin untuk baca konfigurasi, *file* konfigurasi yang disimpan dalam *filesystem* dimuat dan nilai-nilainya disimpan ke dalam memori. Saat pembacaan konfigurasi gagal, maka mesin parkir menampilkan keterangan seperti yang ada pada diagram alir yang kemudian melakukan looping sehingga mesin parkir tidak melanjutkan program. Diagram alir sub-rutin baca konfigurasi dapat dilihat pada Gambar 9

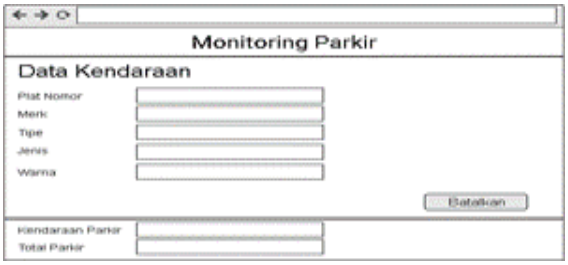


Gambar 9 Diagram Alir Baca Konfigurasi

Modul Aplikasi *Monitoring* Parkir

Program *monitoring* pada sistem parkir ini didesain untuk bisa digunakan pada perangkat *smartphone* maupun komputer sehingga pengelola dapat memilih perangkat yang akan digunakan.

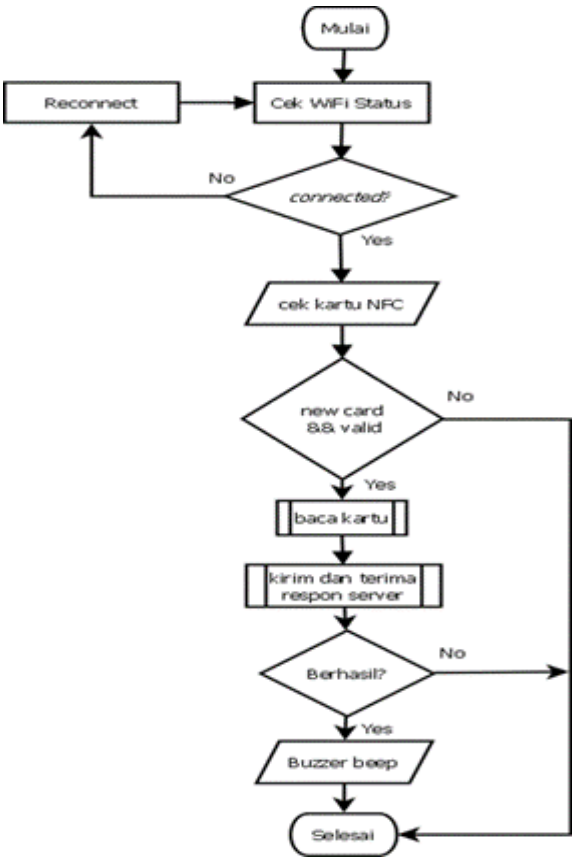
Aplikasi *monitoring* mengakses *monitoring service* menggunakan protokol *websocket* untuk memfasilitasi transfer data secara *real-time*. Data yang diterima dari *monitoring service* ditampilkan pada bagian data kendaraan. Data jumlah kendaraan parkir diperoleh dari *service* parkir dengan melakukan *request* tiap 10 detik. Desain antarmuka aplikasi *monitoring* yang akan dibangun dapat dilihat pada Gambar 10.



Gambar 10. Desain Antarmuka Aplikasi Monitoring

Modul Pengecekan status koneksi WIFI

Proses awal blok fungsi *loop* adalah pengecekan status koneksi WiFi. Jika status koneksi WiFi adalah *not connected* maka program akan menunggu hingga kembali terkoneksi dengan jaringan mengingat mesin parkir membutuhkan koneksi WiFi untuk berkomunikasi dengan *server*. Diagram alir untuk blok fungsi loop() dapat dilihat pada Gambar 11

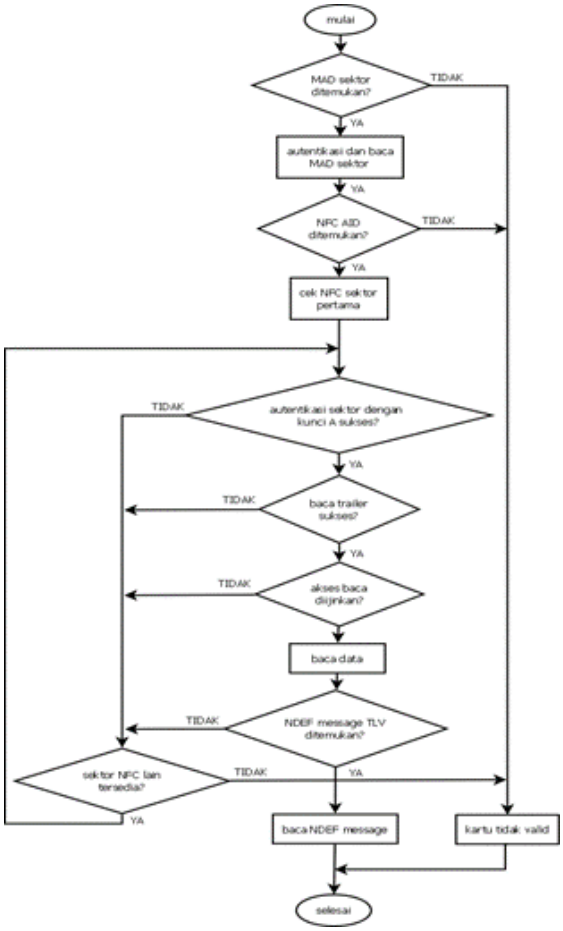


Gambar 11. Diagram Alir Blok Fungsi *loop*()

Modul pengecekan kartu parkir

Pengecekan kartu dilakukan dengan dua tahap. Pertama adalah pendeteksian kondisi

valid kartu dengan format NDEF dan kedua adalah pengecekan NDEF *message*. Prosedur deteksi dan pembacaan NDEF untuk kartu parkir digambarkan pada Gambar 12 berikut ini.

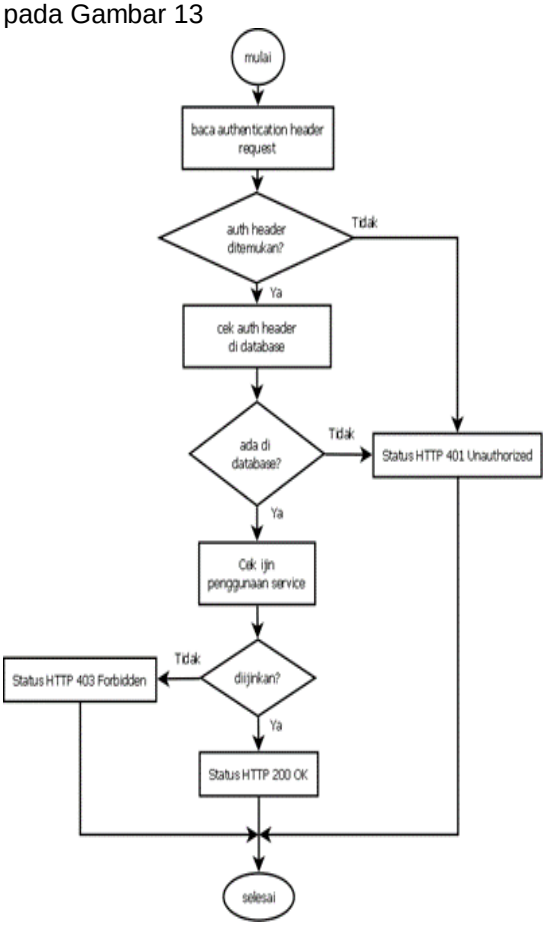


Gambar 12. Diagram Alir Prosedur Deteksi dan Pembacaan NDEF

Modul *Authentication Service*

Authentication service digunakan untuk melakukan autentikasi pengguna yang mengakses *service-service* yang ada pada server. Menggunakan protokol HTTP untuk berkomunikasi dengan *client. Service* ini dirancang untuk membaca Authentication header dari request yang dikirimkan oleh client.

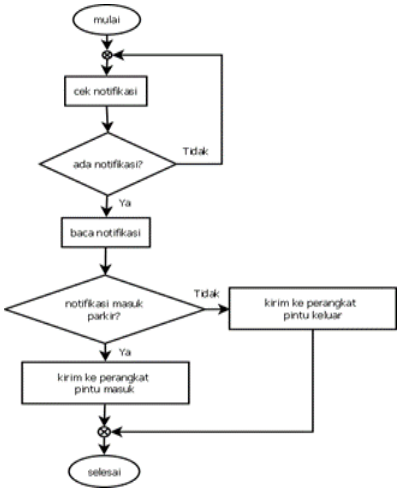
Authentication Service ini digunakan bersama dengan API gateway untuk proses authentikasi dan otorisasi *routing request*. Request dengan username dan password yang tidak terdaftar akan diblok dengan menggunakan kode respon HTTP 401 *Unauthorized*. Jika autentikasi berhasil selanjutnya dilakukan pengecekan otorisasi penggunaan *service*. Jika tidak termasuk pengguna yang diperbolehkan maka routing tidak diteruskan dan *service* mengirimkan respon berupa kode status HTTP 403 *Forbidden*. Request yang valid akan memberikan respon dengan kode status HTTP 200 OK sehingga API gateway akan meneruskan request ke *service* yang dituju. Diagram alir authentication *service* dapat dilihat pada Gambar 13



Gambar 13. Diagram Alir *Authentication Service*

Modul *Monitoring Service*

Monitoring service dirancang menggunakan protokol *web socket* untuk berkomunikasi dengan perangkat *monitoring* agar pertukaran data dapat dilakukan sercara *real-time*. Pengiriman data menunggu adanya notifikasi status dari *service* parkir. Setelah notifikasi parkir diterima, *monitoring service* akan mengirimkan data ke perangkat *monitoring*. Pengiriman data kendaraan disesuaikan dengan notifikasi yang diberikan, apakah notifikasi masuk atau keluar parkir. Diagram alir *monitoring service* dapat dilihat pada gambar 14 berikut.



Gambar 14 Diagram Alir *Monitoring Service*

HASIL PENGUJIAN

Pengujian Aplikasi *Monitoring* Parkir

Pengujian aplikasi *monitoring* parkir dilakukan untuk menguji apakah aplikasi dapat berjalan sesuai desain. Pengujian dilakukan pada *monitoring* pintu masuk dan pintu keluar. Pengujian pada aplikasi *monitoring* parkir pada Tabel 1 dan Tabel 2 sebagai berikut:

1. Pengujian Halaman *Login*

Tabel 1 Pengujian Halaman *Login*

Kasus dan Hasil Uji Pintu Masuk (Data Normal)	
Input	*Username*: monitor, *password*: parkir, pintu masuk
Hasil yang Diharapkan	Masuk ke halaman *monitoring* pintu masuk
Hasil Pengujian	*Login* berhasil dan berpindah ke halaman *monitoring* pintu masuk
Kesimpulan	Sesuai
Kasus dan Hasil Uji Pintu Keluar (Data Normal)	
Input	*Username*: monitor, *password*: parkir, pintu keluar
Hasil yang Diharapkan	Masuk ke halaman *monitoring* pintu keluar
Hasil Pengujian	*Login* berhasil dan berpindah ke halaman *monitoring* pintu keluar
Kesimpulan	Sesuai
Kasus dan Hasil Uji (Data Salah)	
Input	*Username*: a dan *password*: a, pintu masuk
Hasil yang Diharapkan	Terdapat peringatan *login* gagal
Hasil Pengujian	Muncul peringatan *login* gagal
Kesimpulan	Sesuai

2. Pengujian Halaman *Monitoring*

Tabel 2 Pengujian Halaman *Monitoring*

Kasus dan Hasil Uji Pintu Masuk	
Input	*Tap* kartu parkir
Hasil yang Diharapkan	Menampilkan data kendaraan yang masuk parkir
Hasil Pengujian	Menampilkan data kendaraan yang masuk parkir
Kesimpulan	Sesuai
Kasus dan Hasil Uji Pintu Keluar	
Input	*Tap* kartu parkir
Hasil yang Diharapkan	Menampilkan data kendaraan yang keluar parkir
Hasil Pengujian	Menampilkan data kendaraan yang keluar parkir
Kesimpulan	Sesuai
Kasus dan Hasil Uji Pembatalan Parkir Masuk	
Input	Tekan tombol batal
Hasil yang Diharapkan	Menampilkan notifikasi sukses dan data parkir terhapus dari *database*
Hasil Pengujian	Muncul notifikasi sukses dan data parkir terhapus dari *database*
Kesimpulan	Sesuai
Kasus dan Hasil Uji Pembatalan Parkir Keluar	
Input	Tekan tombol batal
Hasil yang Diharapkan	Menampilkan notifikasi sukses dan status parkir kembali menjadi terparkir *database*
Hasil Pengujian	Muncul notifikasi sukses dan data

	parkir menjadi sedang parkir
Kesimpulan	Sesuai

Pengujian Mesin Parkir

Pada mesin parkir yang telah dibangun, pengujian yang dilakukan pertama adalah pengujian lama waktu pembacaan kartu, jarak maksimal terbacanya kartu dan lama waktu mengirim dan menerima data dari *server*. Pengujian dilakukan dengan menggunakan ID kartu 61CD96B0. Hasil pengujian dapat dilihat pada Tabel 3 berikut.

Tabel 3 Pengujian Pertama Mesin Parkir

Perco-baan	Jarak Maksimal (cm)	Pembacaan Kartu (ms)	Respon Dari Server (ms)
1	4,1	42	568
2	4,1	43	932
3	4,0	43	304
4	4,2	43	442
5	4,2	43	1148
6	4,0	43	499
7	4,1	42	1167
8	3,9	43	328
9	3,9	42	635
10	4,0	43	813
11	4,0	43	743
12	4,2	43	537
13	3,8	42	209
14	3,9	43	328
15	4,1	43	1342
16	4,0	43	1191
17	4,0	42	364
18	4,1	43	652
19	4,0	43	435
20	4,1	43	863

Pengujian kedua adalah pengujian proses *update firmware* dan konfigurasi secara *wireless*. Pada pengujian *update firmware* dilakukan dengan mempersiapkan dua *firmware* yang sama dengan perbedaan terletak pada jumlah bunyi *buzzer* pada saat mesin parkir dinyalakan. Untuk *firmware* pertama, jumlah bunyi *buzzer* diset sejumlah dua kali dan *firmware* kedua diset sejumlah tiga kali. Pada pengujian *update* konfigurasi perbedaan terletak pada konfigurasi posisi mesin parkir. Konfigurasi pertama adalah konfigurasi untuk pintu masuk dan konfigurasi kedua digunakan untuk pintu keluar. Hasil pengujian dapat dilihat pada Tabel 4

Tabel 4 Pengujian Kedua Mesin Parkir

Kasus dan Hasil Uji *Update Firmware* Pertama	
Input	*File firmware* pertama
Hasil yang Diharapkan	Bunyi *buzzer* saat mesin dihidupkan sebanyak dua kali
Hasil Pengujian	Terdengar bunyi *buzzer* sebanyak dua kali
Kesimpulan	Sesuai
Kasus dan Hasil Uji *Update Firmware* Kedua	
Input	*File firmware* kedua
Hasil yang Diharapkan	Bunyi *buzzer* saat mesin dihidupkan sebanyak tiga kali
Hasil Pengujian	Terdengar bunyi *buzzer* sebanyak tiga kali
Kesimpulan	Sesuai
Kasus dan Hasil Uji *Update* Konfigurasi Pintu Masuk	
Input	*File* konfigurasi pintu masuk
Hasil yang Diharapkan	Konfigurasi mesin parkir pada posisi mesin untuk pintu masuk
Hasil Pengujian	Konfigurasi mesin parkir pada posisi mesin untuk pintu masuk
Kesimpulan	Sesuai
Kasus dan Hasil Uji *Update* Konfigurasi Pintu Keluar	
Input	*File* konfigurasi pintu keluar
Hasil yang Diharapkan	Konfigurasi mesin parkir pada posisi mesin untuk pintu keluar
Hasil Pengujian	Konfigurasi mesin parkir pada posisi mesin untuk pintu keluar
Kesimpulan	Sesuai

Pengujian Sistem Parkir

Pengujian sistem parkir dilakukan untuk menguji apakah sistem yang dibangun dapat berjalan sesuai desain. *Input* berupa *tap* kartu parkir pada mesin parkir dan *output* adalah status parkir yang ditandai dengan adanya

bunyi dari mesin parkir. Selain bunyi pada mesin parkir, pada perangkat *monitoring* yang digunakan oleh petugas parkir akan menampilkan data kendaraan. Pengujian dilakukan dengan memberikan *input* kartu yang telah terdaftar dan kartu yang tidak terdaftar. Hasil pengujian dengan metode *black box* pada sistem parkir dapat dilihat pada Tabel 5.

Tabel 5. Pengujian Sistem Parkir

Kasus dan Hasil Uji (Data Normal)	
Input	Kartu parkir yang terdaftar
Hasil yang Diharapkan	Mesin parkir mengeluarkan bunyi *beep* dan perangkat monitoring menampilkan data kendaraan.
Hasil Pengujian	Mesin parkir meneluarkan bunyi dan perangkat monitoring menampilkan data kendaraan.
Kesimpulan	Sesuai
Kasus dan Hasil Uji (Data Salah)	
Input	Kartu parkir yang tidak terdaftar
Hasil yang Diharapkan	Mesin parkir tidak mengeluarkan bunyi *beep* dan perangkat monitoring tidak menampilkan data kendaraan.
Hasil Pengujian	Mesin parkir tidak berbunyi dan perangkat monitoring tidak menampilkan data kendaraan.
Kesimpulan	Sesuai

KESIMPULAN

1. Kerja pengontrolan menggunakan arsitektur *microservice* berhasil dengan baik. Yang mana masing-masing modul dapat bekerja dengan baik.
2. Dengan menggunakan *microservice*, beberapa modul yang beroperasi dengan Bahasa yang berbeda-beda namun dapat di koneksikan dan dapat beroperasi sesuai dengan fungsi kerja masing-masing service.
3. Dengan menggunakan arsitektur microservice yang melakukan Pemecahan sistem parkir menjadi beberapa *service* kecil, yaitu service kendaraan, service parkir dan service *monitoring* membuat ketiganya menjadi independen dan tidak bergantung pada bahasa pemrograman sehingga pengembangan dapat dilakukan tanpa perlu merombak keseluruhan sistem.

DAFTAR PUSTAKA

K..Katuwal."Microservices: A Flexible Architecture for Digital Age Version 1.0"American *Journal of Computer Science and Engineering*, Volume3, September 2016 Pages 20-24.

Messina,Antonio,et.al (2016). *A simplefiet Database Pattern for the Microservice Arsitecture.The Eight Internasional Conference on Advances in Database Knolage,and Data,and Applications*.

N.Dragoni et al(2017,September 6) Microservicec:Yesterday,Today and Tomorow.[OnLine].Available:https://link.springer.com/chapter/10.1007/978-3-319-67425-4-2 [Januari 15,2019]

Newman,S.(2015).*Building Microservices*, O,Reilly Media,Inc

NGINX, Inc. (2016). *MICROSERVICE From Design to Deployment.*

Purnama (2016), *Menggunakan microservice dengan Node.Js,* Skripsi AMIKOM Yogyakarta

Sathya (2017), *Smart parking System based on RFID and GSM Tecnologi*

Setiadi, H., Priyandari, Y., & Cahyono, S. I. (2017). Implementation of Parking System Based on Radio Frequency Identification at the Faculty of Engineering Sebelas Maret University. *ITSMART, 6*(1), 39-44.

Trisongko Suryo (2017), *Pola komunikasi REST Service dengan format Javascript Obyect Notation(JSON)*, skripsi AMIKOM