# Implementasi Arsitektur Event-Driven dan Microservices untuk Mesin Vending

[1]Ahmad Fauzi,[2]Eko Harli, Yuli Haryanto[3]
*Fakultas Teknik dan Ilmu Komputer, Universitas Indraprasta PGRI Jakarta[1,2]*
*Jl. Nangka Raya No.58 C, Tj. Bar., Kec. Jagakarsa, Kota Jakarta Selatan*
ahmadfauzi.udzi@gmail.com[1] , ekoharli@gmail.com[2] , haryanto_yuli@yahoo.co.id[3]

***Abstract - The development of vending machines in Indonesia is very fast with the progress of society's need for machines that can be accessed anytime and anywhere. Most vending machines today are integrated with a micro controller on each device in the vending machine. Vending machines that use micro controllers still lack the need for remote control from a centralized system. This problem will apply event-driven architecture to the vending machine as a control center to replace the micro controller position. The results show that an event-controlled architecture replaces the role of the micro controller as a vending machine controller so that the need for distance control on the vending machine is met and increases efficiency in monitoring every transaction in the vending machine.***

***Keyword: event-driven architecture, vending machine, micro controller.***

***Abstrak* – Perkembangan vending machine di Indonesia sangatlah cepat seiring dengan meningkatnya kebutuhan masyarakat akan mesin otomatis yang dapat diakses kapan dan dimana saja. Kebanyakan vending machine sekarang ini terintegrasi dengan mikro kontroler sebagai penggerak pada setiap alat yang ada di *vending machine*. *Vending machine* yang menggunakan mikro kontroler masih memiliki kekurangan disaat adanya kebutuhan pengendalian jarak jauh dari sistem kontrol yang ada di pusat. Masalah ini akan diselesaikan dengan penerapan *event-driven* arsitektur pada *vending machine* sebagai pusat kendali menggantikan posisi mikro kontroler. Hasil penelitian menunjukan bahwa penerapan event-driven arsitektur berhasil menggantikan peran mikro kontroler sebagai pengendali *vending machine* sehingga kebutuhan kontrol jarak jauh terhadap vending machine tersebut terpenuhi dan meningkatkan efisiensi dalam memonitoring setiap transaksi yang ada pada *vending machine*.**

**Keyword: event-driven arsitektur, vending machine, mikro kontroler.**

## I. PENDAHULUAN

*Vending machine* merupakan salah satu contoh sistem otomatis yang akan bekerja secara mandiri dan terus menerus tanpa harus dikendalikan oleh manusia [1]. Di Indonesia, perkembangan *vending machine* tidak hanya sebatas pada bidang makanan dan minuman [2], melainkan telah merambah pada transaksi pembelian tiket, pembelian kartu perdana, dan lain-lain. [3] mengimplementasikan sebuah alat seperti *vending machine* untuk pengelolaan obat**.**

Peningkatan kemampuan vending machine makin hari juga makin berkembang, dimana sekarang ini sudah ada vending machine yang mampu memberikan kembalian uang kepada penggunanya [4]. Selain itu, *vending machine* sekarang ini tidak hanya menerima pembayaran melalui uang cash, teknologi penggunaan e-wallet dan *sms-gateway* juga sudah mulai diterapkan pada sebuah *vending machine* [5].

Penggunaan *vending machine* yang akan bekerja secara terus menerus tentu saja akan membutuhkan perawatan dan monitoring yang baik untuk setiap device yang berada pada sebuah *vending machine*. Penggunaan *microcontroller* yang sifatnya berfokus pada pengendali device hanya berjalan satu arah, dan hanya akan mengendalikan vending machine jika ada permintaan dari pengguna. Hal ini tentu saja menjadi hambatan bagi pengelola *vending machine* untuk mengetahui apakah *vending machine* mereka berjalan normal atau terdapat kerusakan.

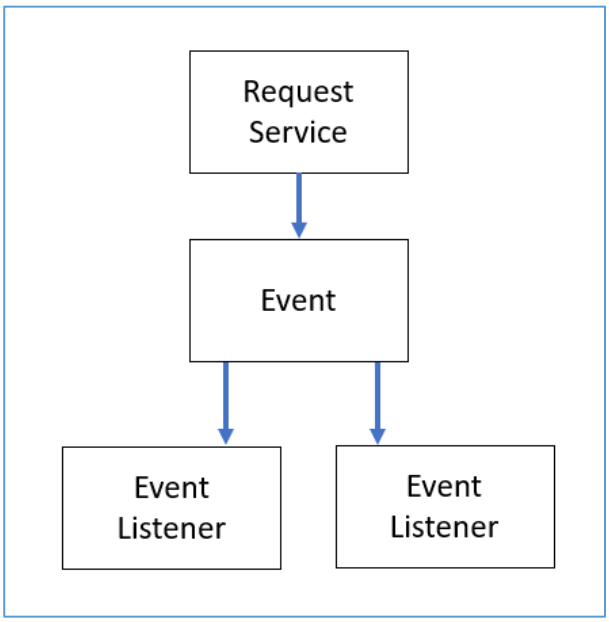
## II. LANDASAN TEORI

### A. Event-driven Architecture

Event-driven Architecture (EDA) adalah sebuah arsitektur dimana suatu event (kejadian) tertentu akan mentrigger suatu proses [6]. EDA bisa dibilang lebih realistis dalam lingkungan bisnis yang canggih, dinamis, modern, dan dapat dilihat sebagai spesialisasi *Service Oriented Architecture* (SOA) dimana komunikasi antar komponen dilakukan sehubungan dengan satu antarmuka acara umum [7].

Arsitektur ini telah banyak digunakan dalam pengembangan sistem informasi seperti didunia industri yang dikembangkan pada penelitian [8] pada penelitian tersebut, arsitektur event-driven terbukti lebih fleksibel untuk mengontrol aplikasi.

Dalam penelitian lainnya, arsitektur ini juga digunakan sebagai *middleware* dalam aplikasi berbasis IoT, seperti yang dilakukan oleh [9]. Penelitian lain juga menggunakan aristektur ini sebagai *middleware* untuk sikronisasi database [10] seperti pada gambar 1.

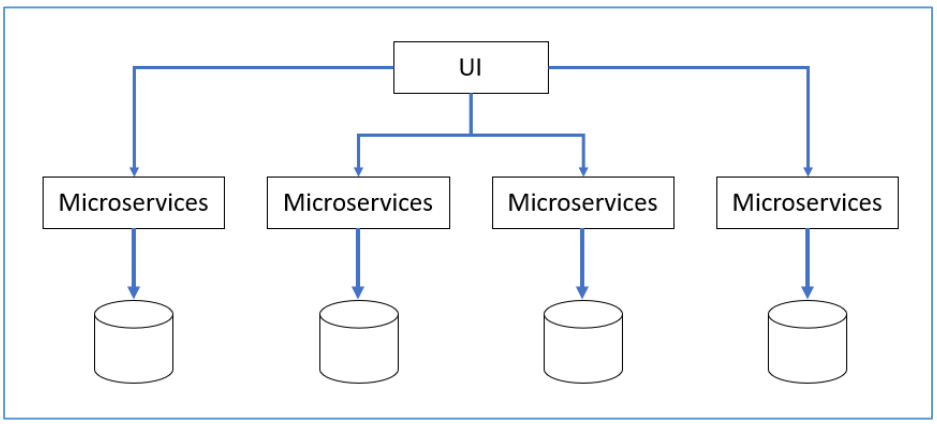



**Gambar 1**. Contoh Sederhana Event-driver Arsitektur

B. *Microservice Architecture*

*Microservice* adalah arsitektur cloud-native yang dikembangkan dari SOA [11]. Arsitektur ini memiliki keunggulan dalam hal *scalability*, *performance*, *availability*, *monitorability*, *security* dan *testability* [12].

Penelitian-penelitian sebelumnya menunjukan bahwa konsep microservice dalam pembuatan aplikasi dapat dilakukan dengan mudah, seperti pada penelitian yang dilakukan oleh [13] yang menggunakan *microservice* sebagai arsitektur dalam pengembangan web pembelajaran. Penelitian lainya [14] menerapkan arsitektur ini sebagai pengolahan data pada IoT, dimana arsitektur ini menunjukan bahwa dapat digunakan untuk mengolah data secara *realtime* seperti pada gambar 2.




**Gambar 2**. Arsitektur Microservices

C. REST API

REST atau RESTful (*Representational State Transfer*) adalah sebuah arsitektur webservice yang memungkinkan sistem request dapat mengakses dan memanipulasi teks yang direpresentasikan dari sebuah webservice [15]. Arsitektur REST sangat sering digunakan untuk membangun microservices, karena integrasi antra service dapat melalui HTTP Request.

REST dapat dijelaskan dalam 5 batasan [16], yaitu:

1. Resource Identification
2. Connectedness
3. Uniform Interface
4. Self-Describing Messages

5. Stateless Interaction

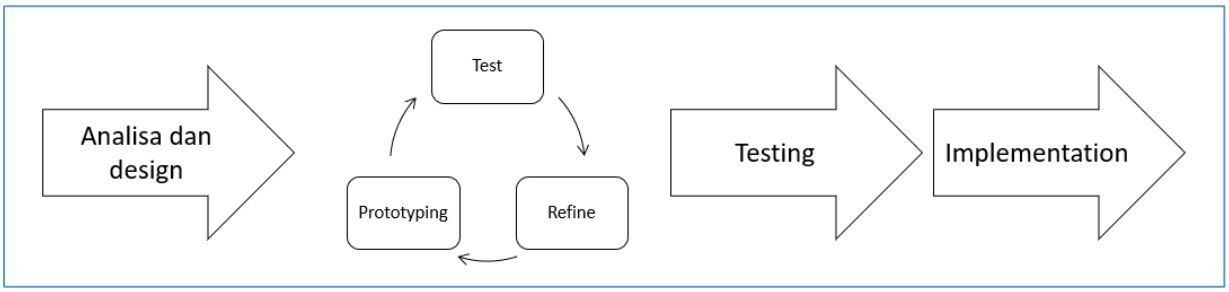
Protokol yang digunakan pada REST, yaitu HTTP, memungkinkan untuk memetakan setiap method yang digunakan dalam REST API, tabel 1 menggambarkan pemetaan HTTP Method pada REST.

Tabel 1. Pemetaan HTTP Status pada REST

| Crud Operation | HTTP Method | HTTP Status Response |
|---|---|---|
| Create | POST | 201 |
| Read | GET | 200 |
| Update | PUT | 200 |
| Delete | DELETE | 200 |

## III. METODE PENELITIAN

Penelitian ini menggunakan metode Rapid Application Development (RAD) dalam mengimplementasikan event-driven arsitektur dan microservice pada vending machine. Metode ini sangat baik digunakan untuk project yang kebutuhan-kebutuhan aplikasi sudah jelas diketahui diawal dan dipahami dengan baik. Selain itu metode ini juga memiliki kelebihan dari sisi waktu yang relatif lebih cepat dan efisien [17].
Langkah-langkah atau alur pelaksanaan penelitian berdasarkan model RAD dapat dilihat pada gambar 1. Terdapat 4 langkah pokok yaitu 1) Analisa dan design, 2) Prototyping, 3) Testing, 4) Implementation seperti yang ditampilkan pada gambar 3.

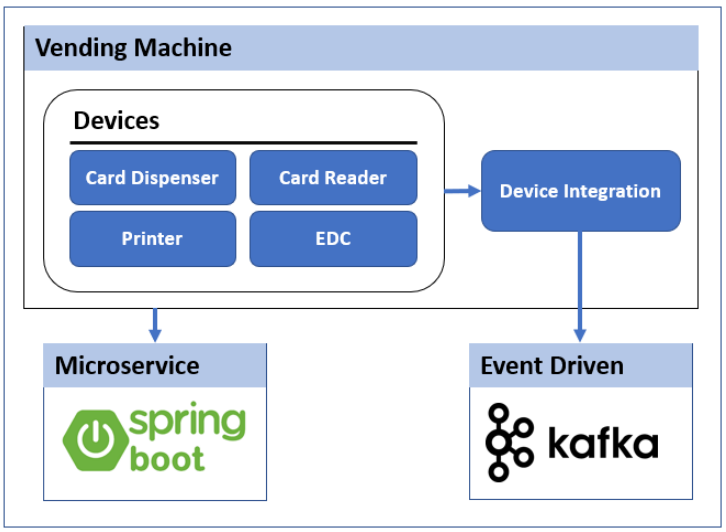



**Gambar 3**. Metode RAD

Tahapan Analisa dan design adalah tahapan dimana proses Analisa permasalahan dan rencana penyelesaian masalah penelitian ditentukan. Analisa permasalahan yang didapat dan solusi yang akan diberikan kemudian diimplementasikan dalam tahapan RAD yaitu *prototyping*.
Pada tahapan RAD ini, siklus development dan internal test dilakukan, diantaranya adalah membangun *event-driven architecture* dan *microservice architecture*. Proses ini merupakan bentuk sederhana dari metode waterfall.
Tahapan selanjutnya adalah testing dan implementation, testing pada tahapan ini sudah menggunakan data real dan dilakukan pada server pre-production. Pada tahapan ini juga dilihat bagaimana sistem bekerja dan kekurangan yang didapat sehingga bisa langsung diperbaiki sebelum tahapan implementation pada server *production*.

## IV. HASIL DAN PEMBAHASAN

A. Rancangan Arsitektur
Arsitektur dari sebuah *vending machine* yang mengimplementasikan *event-driven* dan *microservice* pada penelitian ini digambarkan pada gambar 4 dibawah ini.




**Gambar 4**. Arsitektur Vending Machine

*Event-driven architecture* sangat identik dengan pengiriman pesan sebagai sebuah event, pada penelitian ini, penerapan *event-driven architecture* menggunakan aplikasi bernama Kafka. Aplikasi ini memiliki 2 kunci utama yaitu *producer* dan *concumer*. Producer bertugas untuk menuliskan event pada suatu topic tertentu. Sedangkan consumer bertugas untuk mengambil setiap event yang ada pada suatu topic.
Setiap vending machine pada penelitian ini akan berperan sebagai *consumer*, yang akan membaca setiap *event* yang masuk pada sebuah topic yang sudah ditentukan. Setiap vending machine akan meng-consume 1 topic yang diberi nomor sesuai dengan ID *vending machine* tersebut. Hal ini agar setiap *vending machine* tidak melakukan pekerjaan yang sama satu sama lain.

Untuk mempermudah dalam pemetaan terhadap perangkat mana yang akan dikontrol, maka setiap event yang dituliskan harus juga menyertakan action apa yang diperintahkan, sehingga vending machine dapat dengan mudah mengidentifikasi. Tabel 2 berisi field dan dekskripsi dari format pesan yang dikirim ke sebuah topic pada kafka.

Tabel 2. Deskripsi Message

| Nama field | Deskripsi |
|---|---|
| Topic | Berisi no topic yang berfungsi sebagai unique ID setiap vending machine |
| Key | Unique id setiap proses |
| Action_type | Pekerjaan yang akan dilakukan oleh vending machine |
| url_response | Callback dari setiap pekerjaan yang dilakukan, melaporkan data dan status dari action yang dikerjakan, baik itu sukses maupun gagal |

B. Rancangan API Service
API services yang digunakan dalam penelitian ini terdiri dari 10 Service untuk 5 device yang diujicobakan:
Dengan strutktur API seperti terdapat pada tabel 3 berikut:

Tabel 3. Deskripsi Format API

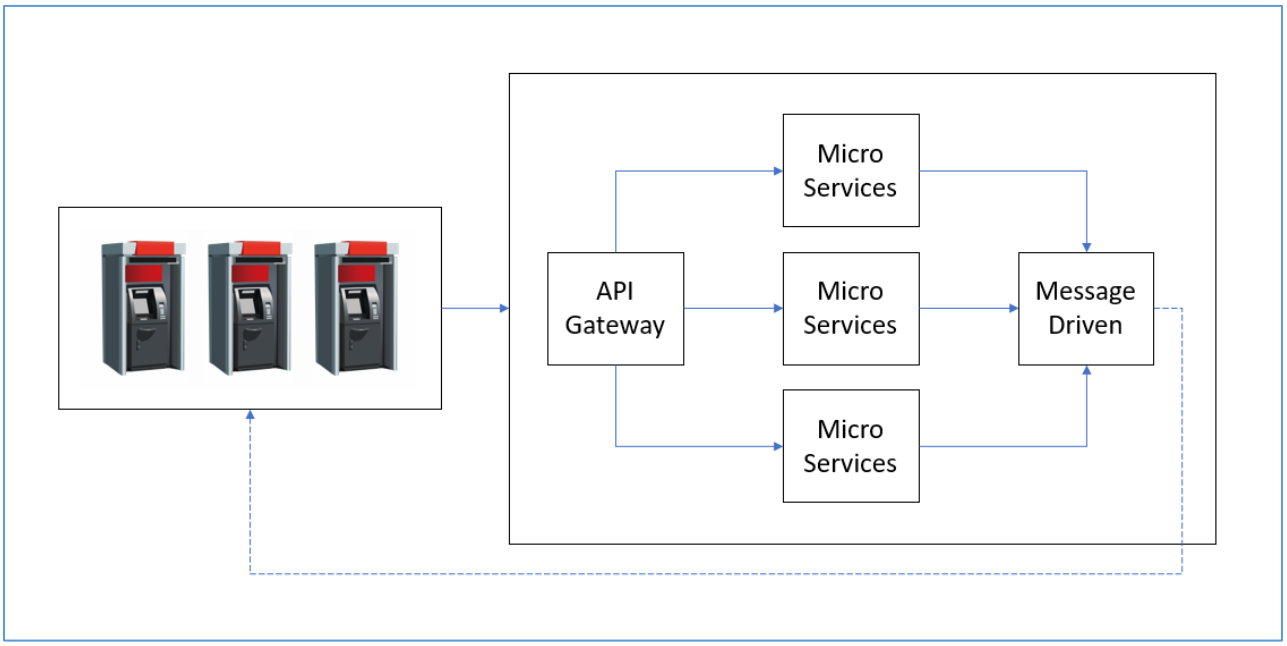
| | | |
|---|---|---|
| Transport Protocol | : | HTTP |
| Content Type | : | JSON |
| Interaction Type | : | Syncronous |
| Security Policy | : | Token Key |
| Format Template Request | : | { "trans_id": "", "action": "", "data": "" } |
| Format Template Response | : | { "trans_id": "", "response_code": "", "response_message": "" } |

Berikut adalah daftar API yang digunakan untuk implementasi pada penelitian ini seperti nampak pada tabel 4:

Tabel 4. Deskripsi API Service

| API | Description |
|---|---|
| /device/read_sn | Membaca serial number kartu dari device card dispenser |
| /device/move_card | Mengeluarkan kartu dari device card dispenser |
| /device/send_edc | Mengirim data amount kedalam EDC |
| /device/print | Mencetak hasil transaksi |

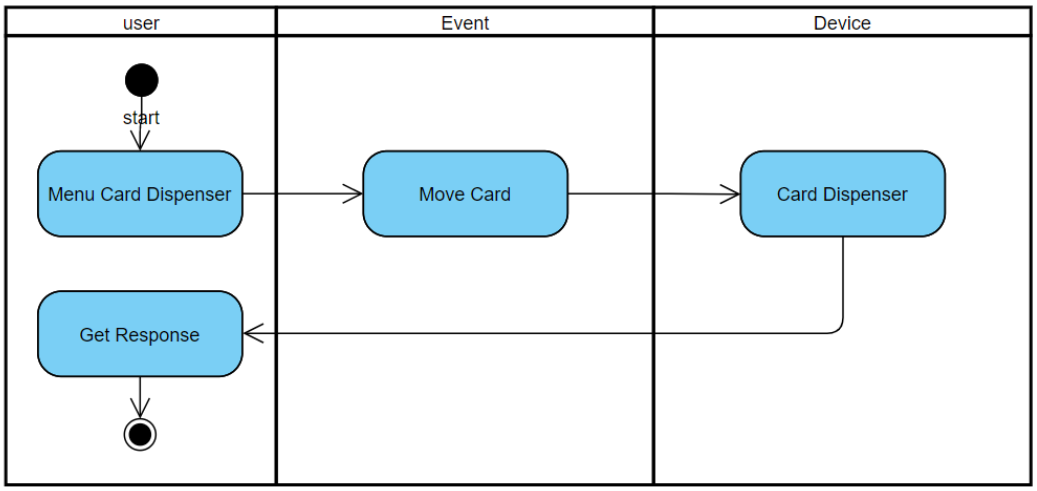
Setiap satu device, penelitian ini membuatnya kedalam 1 microservices, untuk integrasi dari vending machine ke setiap microservice penelitian ini menggunakan reverse proxy dari product NGINX untuk API Gateway, Gambar Menunjukan arsitektur lebih detail dari sistem yang dibangun pada gambar 5.



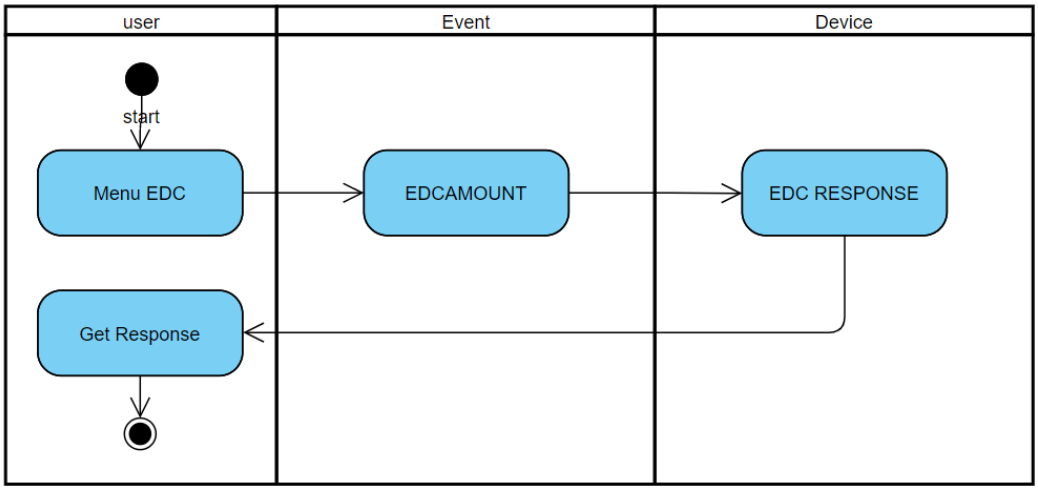

**Gambar 5.** Arsitektur Sistem Secara Keseluruhan

C. Rancangan Flow Proses

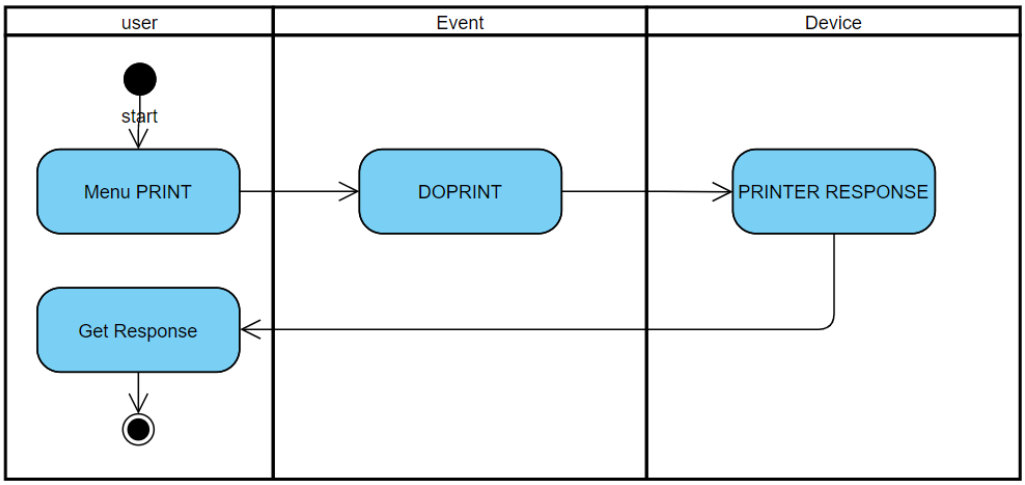
Dalam merancang flow proses, penelitian ini menggunakan activity diagram agar lebih memudahkan melihat integrasi antara service, berikut beberapa flow proses yang ada pada gambar 6, 7 dan 8:




**Gambar 6.** Flow Proses Event Move Card



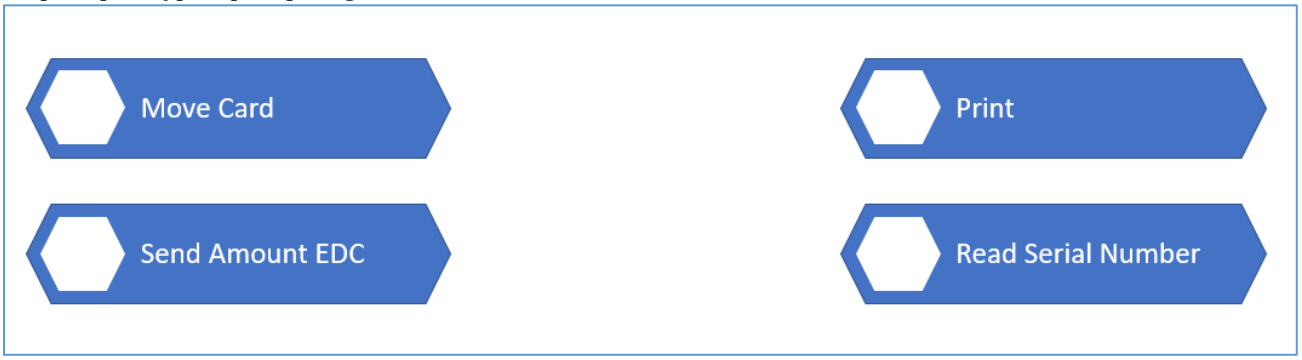

**Gambar 7.** Flow Proses Event Send EDC




**Gambar 8.** Flow Proses Event Print

D. Rancangan Layar

Untuk tampilan layar front-end pada prototype yang dibuat ini, kami menggunakan framework angular sebagai komunikasi antara userinterface dengan device. Hal ini agar tampilan yang dibuat lebih menarik. Berikut rancangan layar untuk tampilan prototype seperti pada gambar 9:




**Gambar 9**. Rancangan Menu Utama

E. Pengujian

Pengujian unit test dilakukan sebelum diimplementasikan pada vending machine. Pengujian meliputi testing microservice API untuk memberikan message kepada event-driven architecture. Tabel 5 menunjukan hasil unit testing terhadap service yang dibangun.

Tabel 5. Hasil Pengujian Microservice

| API | Jumlah Percobaan | Message Driven Acceptable |
|---|---|---|
| /device/read_sn | 50 | 100% |
| /device/move_card | 50 | 100% |
| /device/send_edc | 50 | 100% |
| /device/print | 50 | 100% |

Dari hasil pengujian *microservice* dapat terlihat bahwa message yang dikirim ke *event-driven architecture* dapat dibaca dengan baik. Yang nantinya message ini akan dibaca oleh *vending machine* untuk melakukan *action* yang diharapkan.

F. Hasil Implementasi

Implementasi event-driven arsitektur dan *microservice* pada *vending machine* dilakukan sebagai pengganti microcontroller. Uji coba dilakukan terhadap beberapa perangkat keras yang ada pada sebuah vending machine seperti 1) card dispenser, 2) card reader, 3) Printer, 4) EDC. Untuk mengetahui apakah perangkat tersebut meresponse atau tidak. Penelitian ini menambahkan mekanisme callback untuk setiap event yang diperintahkan. Sehingga setiap perangkat melaporkan status event yang diterima apakah sukses atau gagal. Tabel 6 menunjukan setiap action yang dilakukan oleh setiap perangkat pada *vending machine.*

Tabel 6. Deskrispsi Event

| Perangkat | Action Type | Deskripsi |
|---|---|---|
| Card Dispenser | CARDDISPEF | Menggerakkan kartu keluar |
| | READSN | Membaca Serial Number Kartu |
| Card Reader | PERSO | Menuliskan data pada kartu di *vending machine* |
| EDC | EDCAMOUNT | Mengirim nominal transaksi ke EDC |
| Printer | PRINTST | Mencetak struk untuk sebuah transaksi |

Tabel 7 menyajikan hasil ujicoba implementasi penerapan *event-driven architecture* dan *microservices* pada vending machine.

Tabel 7. Hasil Implementasi

| Perangkat | Action Type | Jumlah Percobaan | Hasil | Waktu Rata-rata |
|---|---|---|---|---|
| Card Dispenser | CARDDISPEF | 100 | OK | ± 10 detik |
| Card Reader | READSN | 100 | OK | ± 15 detik |
| | PERSO | 100 | OK | ± 10 detik |
| EDC | EDCAMOUNT | 100 | OK | ± 1 menit |
| Printer | PRINTST | 100 | OK | ± 5 detik |

Dari hasil tersebut dapat terlihat bahwa setiap perangkat yang ada pada vending machine dapat dengan baik meresponse event yang ada pada producer. Waktu yang dibutuhkan juga relatif cepat yaitu masih dalam hitungan detik. Untuk EDC response nya lebih kurang adalah 1 menit. Hal ini dikaranekan response yang dikirimkan EDC harus menunggu adanya transaksi di EDC tersebut ataupun menunggu timeout dari EDC. Sehingga proses dari mengirimkan perintah untuk EDC hingga adanya laporan sukses atau gagal butuh waktu relatif lebih lama dari perangkat yang lain.

## V. SIMPULAN

Penerapan event-driven arsitektur pada sebuah *vending machine* dapat menggantikan posisi mikrokontroler sebagai pengendali modul-modul yang ada pada vending machine, hasil yang dicapai menunjukan peningkatan kemudahan dalam mengontrol *vending machine* dari jarak jauh secara *realtime* dan dapat mengetahui kerusakan atau kendala yang ada pada vending machine secara cepat. Penelitian selanjutnya dapat menggabungkan event-driven arsitektur dengan mikrocontroler pada sebuah *vending machine*.

## REFERENSI

[1] R. Pradana Putra, I. G. A. P. Raka Agung, and P. Rahardjo, "Rancang Bangun Vending Machine Menggunakan Qr Code Berbasis Mikrokontroler," *J. SPEKTRUM*, vol. 6, no. 2, p. 102, 2019, doi: 10.24843/spektrum.2019.v06.i02.p15.

[2] S. Hafizhuddin, "Rancang Bangun Mesin Penjual Roti Otomatis Berbasis Internet of Things," vol. 8, no. 1, pp. 27–30, 2019, doi: http://dx.doi.org/10.12962/j23373539.v8i1.38362.

[3] L. I. U. Xiangquan, Y. U. N. Chao, Z. Xuefeng, W. Wei, and M. A. Yongbo, "Design and Application for Automated Medicine

Depositing and Dispensing System of Pharmacy," pp. 332–336, 2008, doi: 10.1109/ICCSIT.2008.20.

[4] Alamsyah and I. T. Putri, "Penerapan Algoritma Greedy Pada Mesin Penjual Otomatis (Vending Machine)," *Sci. J. Informatics*, vol. 1, no. 2, pp. 201–209, 2014, doi: 10.15294/sji.v1i2.4608.

[5] S. M. S. Arifin *et al.*, "Smart vending machine based on SMS gateway for general transactions," *QiR 2017 - 2017 15th Int. Conf. Qual. Res. Int. Symp. Electr. Comput. Eng.*, vol. 2017-Decem, no. 2, pp. 34–39, 2017, doi: 10.1109/QIR.2017.8168447.

[6] O. Etzion, "Towards an event-driven architecture: An infrastructure for event processing position paper," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3791 LNCS, pp. 1–7, 2005, doi: 10.1007/11580072_1.

[7] T. Clark and B. S. Barn, "Event driven architecture modelling and simulation," *Proc. - 6th IEEE Int. Symp. Serv. Syst. Eng. SOSE 2011*, no. Sose, pp. 43–54, 2011, doi: 10.1109/SOSE.2011.6139091.

[8] A. Theorin *et al.*, "An Event-Driven Manufacturing Information System Architecture," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 547–554, 2015, doi: 10.1016/j.ifacol.2015.06.138.

[9] U. Auliya, I. Irsahnda, E. S. Pramukantoro, and R. A. Siregar, "Analisis Kinerja IoT Middleware Berbasis Event-Driven pada Raspberry Pi Zero dan Raspberry Pi 2," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya*, vol. 2, no. 10, pp. 3451–3457, 2018.

[10] T. A. Gani and Y. Away, "Pengembangan Middleware Berbasis Metode Event-Driven Untuk Sinkronisasi Database Rfid Book Drop Dan Slims," *J. Komputer, Inf. Teknol. dan Elektro*, vol. 3, no. 3, pp. 26–31, 2018.

[11] M. Waseem, P. Liang, and M. Shahin, "A Systematic Mapping Study on Microservices Architecture in DevOps," *J. Syst. Softw.*, vol. 170, p. 110798, 2020, doi: 10.1016/j.jss.2020.110798.

[12] S. Li *et al.*, "Understanding and Addressing Quality Attributes of Microservices Architecture: A Systematic Literature Review," *Inf. Softw. Technol.*, p. 106449, 2020, doi: 10.1016/j.infsof.2020.106449.

[13] Y. Chandra, T. Putra, T. Adi, P. Sidi, and J. E. Samodra, "Implementasi Arsitektur Microservice pada Aplikasi Web Pengajaran Agama Islam Home Pesantren," vol. 1, no. November, pp. 88–97, 2020.

[14] L. Bixio, G. Delzanno, S. Rebora, and M. Rulli, "A flexible IoT stream processing architecture based on microservices," *Inf.*, vol. 11, no. 12, pp. 1–19, 2020, doi: 10.3390/info11120565.

[15] E. Edy, F. Ferdiansyah, W. Pramusinto, and S. Waluyo, "Pengamanan Restful API menggunakan JWT untuk Aplikasi Sales Order," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 3, no. 2, pp. 106–112, 2019, doi: 10.29207/resti.v3i2.860.

[16] P. F. Tanaem, D. Manongga, and A. Iriani, "RESTFul Web Service Untuk Sistem Pencatatan Transaksi Studi Kasus PT . XYZ," vol. 2, no. April, 2016.

[17] B. Prashanth Kumar and Y. Prashanth, "Improving the Rapid Application Development process model," *Proc. 2014 Conf. IT Business, Ind. Gov. An Int. Conf. by CSI Big Data, CSIBIG 2014*, pp. 1–3, 2014, doi: 10.1109/CSIBIG.2014.7056962.