 ISSN:2302-8149
e-ISSN:2540-9719

Implementasi Microservices menggunakan REST API untuk UMKM berbasis Raspberry PI

Microservices implementation using Rest API for MSMEs based on Raspberry PI

[1]Norma Ningsih*, [2] Nailul Muna, [3]Faridatun Nadziroh, [4]Endryco Farel Rianrachmatullah, [5]Izzuddin Ahmad Afif
[1,4,5]*Internet Engineering Technology, Department of Electrical Engineering,* PENS
[2,3]*Telecommunication Engineering, Department of Electrical Engineering,* PENS
Jalan Raya ITS, Keputih, Kec. Sukolilo, Surabaya, Indonesia
*e-mail: norma@pens.ac.id

(***received*:** 2 Oktober 2023, ***revised*:** 6 November 2023, ***accepted*:** 18 November 2023)

Abstrak

Usaha Mikro, Kecil dan Menengah (UMKM) merupakan kegiatan usaha yang mampu memperluas lapangan kerja dan memberikan pelayanan ekonomi secara luas kepada masyarakat. Peran teknologi digital meningkat signifikan untuk membantu aktivitas dan transaksi melalui internet baik diakses melalui teknologi *web*, *mobile* maupun *dekstop*. Dalam menjalankan proses bisnisnya, UMKM membutuhkan akses ke layanan lain seperti bank untuk proses pembayaran. selain itu aplikasi multiplatform dibutuhkan untuk dapat memberikan kemudahan *customer* untuk dapat mengakses *services* yang diberikan oleh UMKM. Berdasarkan kebutuhan diatas, maka dikembangkan aplikasi *back-end* UMKM menggunakan arsitektur web service REST API yang dijalankan pada raspberry PI yang difungsikan sebagai server untuk proses komputasi dan penyimpanan data. Metode pengembangan sistem menggunakan metode waterfall dengan bahasa pemrograman golang. sistem yang dikembangkan telah diuji menggunakan tools insomnia untuk REST API yang telah dibuat dan tools vegeta sebagai HTTP *load testing.* Hasil yang diperoleh dalam pengujian ini menggunakan *method* POST dan GET adalah pada jumlah request 25 dan 150 transaksi 100% berhasil, namun pada jumlah request 500 terdapat transaksi yang gagal sebesar 32%. Konsumsi daya semakin meningkat sesuai dengan jumlah akses pada sistem dengan nilai rata-rata pada masing-masing skenario adalah 4.24, 4.54, 4.7 watt.

***Kata kunci*:** UMKM, Rest API, Raspberry PI, Load Testing.

Abstract

Micro, Small and Medium Enterprises (MSMEs) are business activities that are able to expand employment opportunities and provide broad economic services to the community. The role of digital technology has increased significantly to assist activities and transactions via the internet, whether accessed via web, mobile or desktop technology. In carrying out their business processes, MSMEs need access to other services such as banks for payment processing. Apart from that, multiplatform applications are needed to make it easier for customers to access the services provided by MSMEs. Based on the above needs, an MSME back-end application was developed using the REST API web service architecture which runs on a Raspberry PI which functions as a server for computing and data storage processes. The system development method uses the waterfall method with the Golang programming language. The system developed has been tested using the Insomnia tool for the REST API that has been created and the Vegeta tool as HTTP load testing. The results obtained in this test using the POST and GET methods were that for a number of requests of 25 and 150 transactions were 100% successful, however for a number of requests of 500 there were 32% failed transactions. Power consumption increases according to the number of accesses to the system with the average value in each scenario being 4.24, 4.54, 4.7 watts.

Keywords: MSMEs, Rest API, Raspberry PI, Load Tetsing.

 ISSN:2302-8149
e-ISSN:2540-9719

1 Pendahuluan

Usaha Mikro, Kecil dan Menengah (UMKM) merupakan kegiatan usaha yang mampu memperluas lapangan kerja dan memberikan pelayanan ekonomi secara luas kepada masyarakat, dan dapat berperan dalam proses pemerataan dan peningkatan pendapatan masyarakat, mendorong pertumbuhan ekonomi, dan berperan dalam mewujudkan stabilitas nasional [1]. UMKM memberikan kontribusi sekitar 90% dari usaha yang ada di dunia [2]. Banyak sekali tantangan yang harus dihadapi oleh para pelaku UMKM untuk dapat bertahan dan berkembang di era saat ini seperti kurangnya modal usaha, lemahnya kemampuan manajerial, terbatasnya era pemasaran dan daya saing yang lemah [3]. Kebutuhan teknologi sangat diperlukan agar bisnis pada UMKM dapat bersaing dengan dunia luar [4]. Penggunaan TI dapat meningkatkan transformasi bisnis melalui kecepatan, ketepatan dan efisiensi pertukaran informasi dalam jumlah yang besar [5].

Dalam menjalankan proses bisnisnya, UMKM membutuhkan akses ke layanan lain seperti bank untuk proses pembayaran, logistik untuk proses pengiriman dan layanan lainnya untuk mendukung bisnis yang dijalankan. selain itu aplikasi multiplatform dibutuhkan untuk dapat memberikan kemudahan *customer* untuk dapat mengakses *services* yang diberikan oleh UMKM. arsitektur *microservices* membangun sebuah sistem dimana setiap layanannya dapat bekerja secara *independent.* Hal tersebut bermanfaat untuk dapat memfasilitasi pengembangan sistem kedepannya tanpa mengganggu performa layanan yang lain [6], [7]. *web service* adalah modul perangkat lunak yang berisi sekumpulan protokol dan standar dan berjalan di dalam web server sehingga dapat diakses melalui internet. *web service* dimaksudkan untuk menjalankan suatu fungsi tertentu seperti pertukaran data antara aplikasi atau sistem yang berbeda. Pada proses pertukaran data terdapat 3 komponen *web services* yaitu *service provider, service requestor* dan *service registry* [8].

Pada proses pengembangan *web service* diperlukan arsitektur komunikasi seperti apa yang akan diimplementasikan. hal ini diperlukan agar komunikasi antara *client* dan *server* memiliki pola dan gaya yang konsisten. salah satu arsitektur yang banyak digunakan adalah REST. Teknologi Rest API adalah arsitektur yang banyak digunakan saat ini dimana proses komunikasi menggunakan format data JSON [9][10]. *Back-end* adalah bagian dari aplikasi yang berfungsi untuk menyediakan segala kebutuhan sistem yang tak terlihat oleh pengguna (tidak secara langsung berinteraksi dengan pengguna) seperti penyimpanan data, pengolahan data serta transaksi data atau informasi secara aman. proses yang terjadi pada sisi *back-end* digunakan untuk mendukung aplikasi *front-end* bekerja dengan baik. Back-end merupakan suatu program yang berjalan pada sisi server (server-side) yang melakukan tugas untuk berinteraksi langsung dengan basis data dalam melakukan manipulasi data [11]. Penelitian serupa telah dilakukan oleh [7] mengenai pengembangan aplikasi *backend* untuk sistem manajemen tesis menggunakan arsitektur microservices dan restful API. Sistem ini dibagi menjadi 4 layanan yaitu layanan pengguna, diskusi, penjadwalan dan tesis. HTTP method yang digunakan pada penelitian tersebut adalah GET, POST, PUT dan DELETE. Dengan arsitektur dan web service yang digunakan diperoleh hasil pengujian yaitu jumlah permintaan meningkatkan waktu respon kinerja, penggunaan CPU, dan konsumsi memory. rata-rata penggunaan sumber daya setiap layanan berdasarkan waktu respons sebesar 61,64 ms, penggunaan CPU sebesar 8,64%, dan penggunaan memori sebesar 89,47 Mb. Pada penelitian [10] dilakukan pengembangan sistem *backend* server untuk sistem *cashless payment* pada komunitas retail menggunakan Rest API. Pada penelitian ini sudah dibangun protokol data yang dapat diakses oleh aplikasi klien (frontend), layanan backend dengan sistem login, daftar, isi ulang saldo dengan virtual account dll. Sistem backend diuji ketahanannya dengan 100 API permintaan dilakukan dalam 1 detik. Tingkat keberhasilan secara keseluruhan sistem adalah 76,92% dari 13 fitur yang ditawarkan. Pada penelitian tersebut tidak dijelaskan secara spesifik spek server yang digunakan untuk melakukan komputasi. Perbedaan dengan penelitian yang dilakukan adalah terdapat raspberry pi yang digunakan sebagai server untuk melakukan proses komputasi dan penyimpanan data.

Dengan dikembangkannya sistem *microservices* pada aplikasi *backend* UMKM menggunakan raspberry pi diharapkan dapat meningkatkan dan mempermudah pihak UMKM untuk menjalankan proses bisnisnya dengan terus dapat mengembangkan aplikasi yang digunakan tanpa mengganggu aplikasi yang telah *running.* selain itu, diharapkan juga dengan pemanfaatan raspberry pi dapat mengurangi biaya yang harus dikeluarkan UMKM untuk menyewa server pada pihak kedua dan raspberry pi memiliki konsumsi daya yang rendah dibandingkan server konvensional.

 ISSN:2302-8149
e-ISSN:2540-9719

2 Tinjauan Literatur

Konsumsi energi listrik di Indonesia semakin hari semakin meningkat. hal tersebut dikarenakan adanya peningkatan jumlah penduduk yang sekaligus meningkatkan jumlah gedung, rumah dan lainnya [12]. telah banyak penelitian yang telah dilakukan dengan memanfaat raspberry pi sebagai server. selain memiliki konsumsi daya yang rendah, raspberry pi juga relatif lebih murah dan bersifat portable jika digunakan sebagai server. Penelitian yang dilakukan oleh [13], pada penelitian tersebut raspberry digunakan sebagai video server. server streaming video/audio real-time yang dirancang menggunakan kemampuan pemrograman dan kontrol Raspberry pi. Video ditangkap melalui port modul kamera Raspberry pi dan dikompresi dan dikirim menggunakan standar khusus yang menerapkan HTTP sehingga dapat diterima dari jaringan. Sistem ini menemukan solusi untuk pengadaan server dengan biaya tinggi dan sistem server video yang kompleks serta penyimpanan massal yang digunakan untuk sistem tersebut. Pada [14], memanfaatkan *raspberry Pi* sebagai server media *offline* yang menyediakan sinyal wifi dari adapter pi yang berfungsi untuk transmisi data. sistem ini berjalan pada sistem operasi linux dan sebuah SD card sebagai media penyimpanannya serta dapat mengakomodasi sebanyak 50 pengguna. Pada [15], penelitian ini dirancang suatu koneksi VPN SSTP dengan server Raspberry Pi dan client PC dengan sistem operasi Ubuntu. Dan juga, dibuat suatu rancangan VPN PPTP sebagai perbandingan terhadap VPN SSTP. Hasil dari pengujian performa, menunjukkan bahwa VPN SSTP sedikit lebih baik daripada VPN PPTP, terutama pada pengujian *packet loss* dan *round trip time*. Tetapi pada pengujian SFTP file transfer, VPN PPTP lebih baik daripada VPN SSTP. pada pengujian menunjukkan bahwa sistem aman terhadap serangan *sniffing*.

Pada penelitian [16][17][18][19] menggunakan arsitektur rest API dalam pembuatan web service untuk berkomunikasi dengan aplikasi lainnya. Berdasarkan penelitian [19], RESTFul WS merupakan sebuah gaya arsitektur yang diadopsi dari konsep REST dengan tujuan yakni sebagai penghubung antara sumber daya dan client. sistem yang dikembangkan adalah STAR UKSW. STARS UKSW adalah sistem informasi yang digunakan untuk melakukan pencatatan akan kegiatan-kegiatan yang dilaksanakan di lingkungan UKSW yang berbasis web . Berdasarkan hasil penelitian yang dilakukan, terdapat korelasi yang telah disesuaikan antara Metode HTTP (POST, GET, PUT, DELETE) dan operasi CRUD dan batasan dalam membangun RESTFul terutama untuk STARS yang mengacu pada Uniform Interface, Connectedness, Self-Describing Messages dan Stateless Interactions. Pada [17], mengimplementasikan Rest API untuk aplikasi Usaha jasa berbasis multiplatform. Pada aplikasi tersebut Tukang dapat membuat promosi jasa, menerima pemesanan serta melakukan messaging dengan pemesan. Semua action ini dapat dilakukan setelah melakukan action login. pemesan dapat melakukan pencarian jasa tukang, messaging, melakukan pemesanan jasa tukang, serta memberi ulasan jika telah melakukan pemesanan. hasil pengujian responden tukang memiliki jawaban sangat setuju sebesar 51.4%, dikarenakan 51.4% merupakan hasil tertinggi pada respon penilaian jawaban.

Pada paper ini disajikan sebuah sistem aplikasi *back-end* UMKM yang memiliki fitur menampilkan katalog barang, melakukan transaksi berupa menambahkan ke dalam keranjang belanja serta melakukan transaksi pembayaran sesuai barang yang dipesan. aplikasi dikembangkan menggunakan bahasa pemrograman golang berbasis container docker. sistem yang telah dibuat dijalankan pada raspberry PI menggunakan sistem operasi linux. pengujian dilakukan menggunakan tools vegeta untuk menguji http load testing.

3 Metode Penelitian

Penelitian ini terdapat beberapa tahapan yang dilaksanakan untuk mencapai tujuan penelitian yaitu mengembangkan sistem untuk UMKM berupa aplikasi *back-end* menggunakan *web service* REST API berbasis raspberry PI. Metode penelitian yang digunakan ditunjukkan oleh blok diagram pada Gambar 1.

 ISSN:2302-8149
e-ISSN:2540-9719

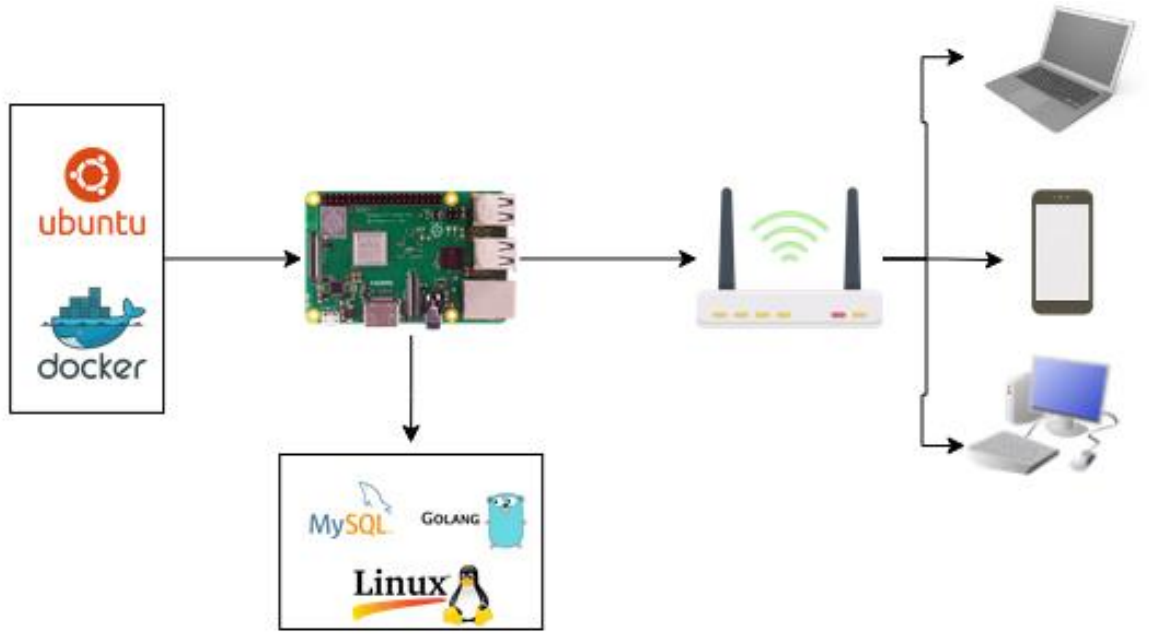
Gambar 1. Blok Diagram Metode Penelitian

3.1 Identifikasi Kebutuhan Sistem

Pada penelitian ini, metode identifikasi menggunakan tahapan observasi, wawancara dan studi literatur. Observasi dilakukan dengan cara mengamati aktivitas yang dilakukan oleh obyek penelitian yaitu UMKM. Bagaimana alur bisnis yang terjadi dalam transaksi jual beli pada UMKM. Wawancara dilakukan pada beberapa pelaku UMKM untuk mengetahui secara lebih detail permasalahan dan kebutuhan apa saja yang diperlukan. Studi *literature* dibutuhkan sebagai pendukung secara teknis terkait pengembangan sistem yang dibuat.

3.2 Desain Sistem UMKM

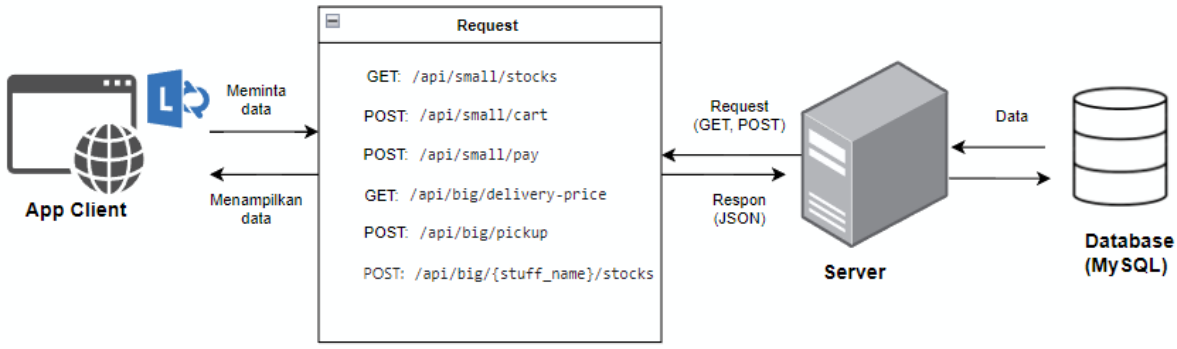
Perancangan sistem UMKM menggunakan perangkat keras *Raspberry Pi* sebagai server untuk melakukan proses komputasi dan mengelola data pada aplikasi. Di dalam *raspberry Pi server*, semua layanan yang digunakan untuk melayani *customer* dikembangkan menggunakan *docker* sebagai *container manager* yang bertujuan agar lebih mudah untuk dikelola dan digunakan. *Container* merupakan suatu wadah atau lingkungan virtualisasi yang terisolasi yang melingkupi satu atau beberapa *dependency (library)* agar mampu mengoperasikan suatu aplikasi . Jenis *container* yang digunakan pada penelitian ini adalah *docker*. *Docker container image* merupakan template yang berisi kumpulan file seperti kode aplikasi, *library* dan *dependency* aplikasi lainnya yang dibutuhkan. docker image yang dibuat ada dua yaitu MySQL untuk penyimpanan datanya serta *image* untuk aplikasi yang dikembangkan. Dalam menjalankan *image* pada *docker* menggunakan *docker compose*. Sistem operasi yang digunakan pada *raspberry Pi* adalah *Linux Ubuntu 20.04.* Dalam pembuatan aplikasi *back-end* menggunakan bahasa pemrograman golang. Aplikasi ini dapat diakses secara lokal melalui router Wifi. Blok diagram sistem UMKM secara lengkap dapat dilihat pada Gambar 2.



Gambar 2. Blok Diagram Sistem UMKM

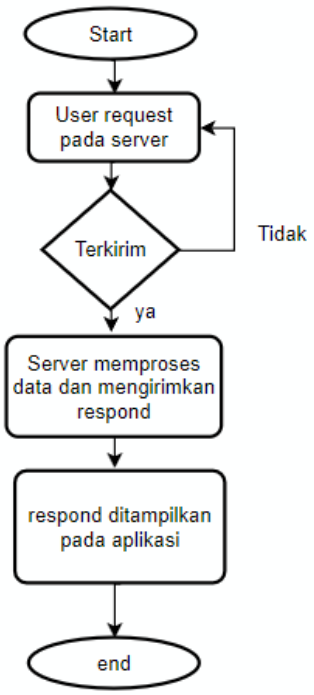
 ISSN:2302-8149
e-ISSN:2540-9719

API merupakan *Application interface Programming* yaitu antar muka yang menyediakan cara atau metode untuk suatu aplikasi dapat berkomunikasi dengan aplikasi lainnya walaupun berbeda platform. REST (*Representational State Transfer*) adalah model arsitektur aplikasi yang umumnya digunakan pada sistem terdistribusi yang berfokus pada skalabilitas dan interaksi antara komponen yang ada pada sistem. Dalam pertukaran data arsitektur REST menggunakan protokol HTTP dan memiliki beberapa komponen seperti *URL design*, *HTTP method*, *HTTP response code* dan *format response*. Pada penelitian ini *HTTP method y*ang digunakan adalah POST dan GET, format respon yang diterima dari server berupa data JSON. Arsitektur rest API yang digunakan pada penelitian ini dapat dilihat pada Gambar 3, dimana transaksi yang terjadi pada sistem aplikasi yaitu komunikasi yang dilakukan dari aplikasi *client (front-end)* tidak boleh diakses secara langsung atau melalui perantara yaitu melalui rest API untuk melakukan request kepada server. jika komunikasi berhasil server akan merespon menggunakan format data JSON.



Gambar 3. Arsitektur Rest API

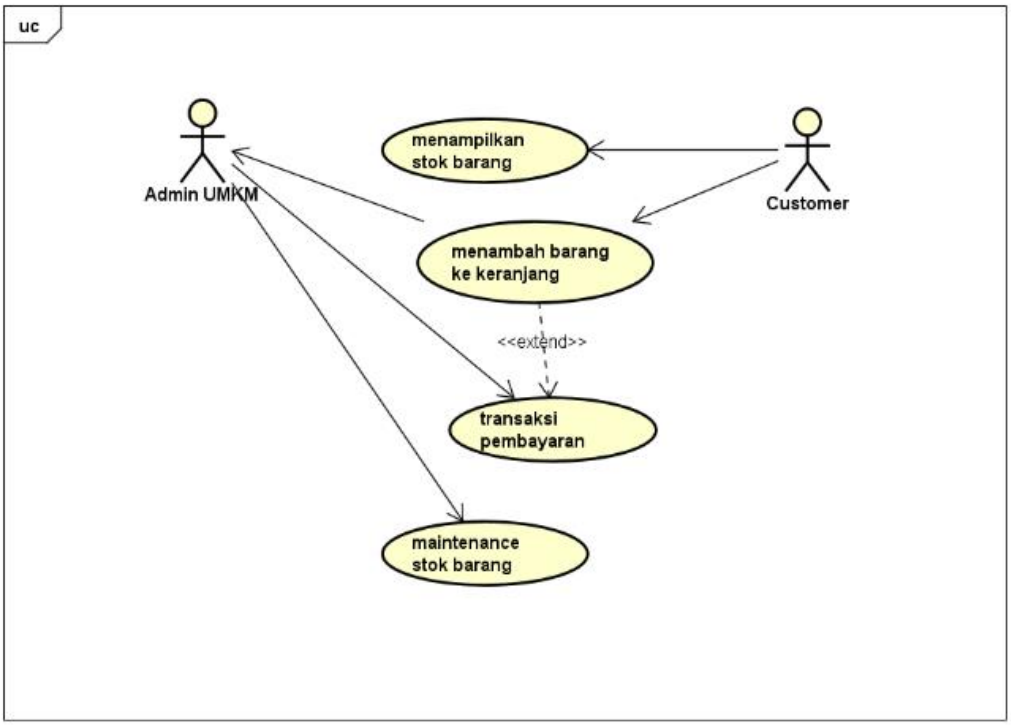
Pada Gambar 4 menunjukkan alur dari sistem yang akan dikembangkan. Pengguna akan melakukan permintaan pada server, jika permintaan berhasil terkirim maka server akan merespon sesuai dengan permintaan dari klien. selanjutnya data respond akan dikirimkan pada klien untuk ditampilkan pada aplikasi. setelah memperoleh data respon, *client* dapat mengurai (*parsing)* data tersebut dan diolah sesuai kebutuhan.



Gambar 4. Alur Diagram Server

 ISSN:2302-8149
e-ISSN:2540-9719

Pada Gambar 5 menunjukkan diagram *usecase* sistem dari aplikasi UMKM yang dikembangkan. pada diagram tersebut dapat dilihat bahwa aktor yang dapat mengakses aplikasi adalah admin dari UMKM dan pelanggan. pelanggan dapat melihat menu atau barang apa saja yang dijual kemudian dapat melakukan pemesanan atau transaksi pada toko tersebut. pelanggan juga akan menerima notifikasi terkait dengan status transaksi yang dilakukan. Aktor admin UMKM dapat melakukan kelola data produk atau barang seperti melakukan proses CRUD (*create, read, update and delete*). selain itu admin juga dapat menerima dan melakukan verifikasi terhadap transaksi yang dilakukan.



Gambar 5. Diagram *Usecase* Aplikasi UMKM

4 Hasil dan Pembahasan

Pada bab hasil dan pembahasan akan dipaparkan hasil produk yang telah dihasilkan pada proses pengembang system.

4.1 Implementasi Rest API

Hasil dari penelitian ini adalah sebuah aplikasi *backend* dengan teknologi REST API menggunakan bahasa pemrograman golang. aplikasi *backend* diimplementasikan menggunakan teknologi *container* yaitu docker dan *operation system* Linux. Aplikasi menggunakan perangkat keras *Raspberry Pi* sebagai server untuk melakukan proses komputasi dan mengelola data pada aplikasi. Adapun fungsionalitas yang terimplementasikan pada aplikasi ini adalah sebagai berikut:

4.1.1. Menampilkan stok barang

Endpoint ini digunakan untuk menampilkan daftar dan jumlah stok barang yang dimiliki oleh UMKM. Pada Tabel 1 menunjukkan *request* dan *response* dari *endpoint* stok barang yang terdiri dari beberapa variabel seperti *method* yang digunakan pada *http request* adalah get dengan url /api/small/stocks, serta beberapa parameter yang diguankan untuk melakukan *request* terhadap *server* dan respon dari server.

ISSN:2302-8149
e-ISSN:2540-9719

Tabel 1. Request dan Response Stok Barang

Variabel	Keterangan
Method	GET
Url	/api/small/stocks
Nama Endpoint	List Of Goods
Query Paramaeters	• page (Number): Halaman yang ingin ditampilkan • total_goods (Number): Jumlah barang yang ingin ditampilkan dalam satu halaman • sort (String): Jenis urutan. Nilai yang valid adalah ASC dan DESC. Default DESC. • sort_by (String): Urut daftar barang berdasarkan property tertentu. Default id, diurut berdasarkan ID barang.
Request	GET /api/small/stocks?sort_by=id HTTP/1.1
Response	{ "code": 200, "status": "OK", "data": [{ "id": 1, "name": "Kopi Gula Aren", "stock": 100, "price": 5000 }, { "id": 2, "name": "Kopi Susu", "stock": 30, "price": 6000 }, { "id": 3, "name": "Pisang Goreng", "stock": 150, "price": 1500 }] }

4.1.2. Menambahkan barang ke keranjang

Endpoint ini digunakan untuk membuat record transaksi belanja untuk seorang user. Di simulasi ini langsung menggunakan record transaksi untuk menyederhanakan proses. Namun transaksi yang statusnya belum dibayarkan disebut sebagai keranjang belanja. Pada Tabel 2 menunjukkan *request* dan *response* dari *endpoint* stok barang yang terdiri dari beberapa variabel seperti *method* yang digunakan pada *http request* adalah post dengan url /api/small/cart, serta beberapa parameter yang diguankan untuk melakukan *request* terhadap *server* dan respon dari server.

 ISSN:2302-8149
e-ISSN:2540-9719

Tabel 2. Request dan Response Tambah Barang

Variabel	Keterangan
Method	POST
Url	/api/small/cart
Nama Endpoint	Add to Cart
Query Paramaeters	Request body: • cart_id (Number, *Optional*): ID dari keranjang belanja dari seorang user. Apabila keranjang belanja sudah ada, property ini harus terisi. • user_id (Number): ID dari pengguna. • goods_id (Number): ID barang yang ingin ditambahkan ke dalam keranjang belanja. • goods_price (Number): Harga satuan barang. • total (Number): Jumlah barang yang ditambahkan. Response: • cart_id (Number): ID keranjang belanja. • total_goods (Number): Jumlah barang yang ada di keranjang belanja saat ini • total_amount (Number): Total belanja keseluruhan saat ini

Request	POST /api/small/cart HTTP/1.1
	Content-Type: application/json
	
	{
	"user_id": 100,
	"goods_id": 1,
	"goods_price": 2000,
	"total": 3
	}

Response	{
	"code": 200,
	"status": "OK",
	"data": {
	"cart_id": 1,
	"total_goods": 3,
	"total_amount": 6000
	}
	}

4.1.3. Melakukan pembayaran atau pembelian

Endpoint ini digunakan untuk membayar barang yang ada pada keranjang belanja dari user. Proses yang terjadi sebenarnya hanyalah mengubah status dari keranjang belanja menjadi "transaksi". Pada Tabel 3 menunjukkan *request* dan *response* dari *endpoint* stok barang yang terdiri dari beberapa variabel seperti *method* yang digunakan pada *http request* adalah get dengan url /api/small/pay, serta beberapa parameter yang diguankan untuk melakukan *request* terhadap *server* dan respon dari server.

 ISSN:2302-8149
e-ISSN:2540-9719

Tabel 3. request dan response pembayaran

Variabel	Keterangan
Method	POST
Url	/api/small/pay
Nama Endpoint	Do Transaction
Query Paramaeters	Request body: • cart_id (Number): ID keranjang belanja • payment_amount (Number): Jumlah pembayaran yang dilakukan oleh user Response: • transaction_id (String): ID transaksi • total_amount (Number): Jumlah harga pembelian barang • payment_amount (Number): Jumlah pembayaran yang dilakukan oleh user • return_amount (Number): Jumlah uang yang dikembalikan oleh merchant kepada user

Request	{
	"cart_id": 1,
	"payment_amount": 4000
	}

Response	{
	"code": 200,
	"status": "OK",
	"data": {
	"transaction_id": "b915141c-82a9-48eb-842f-b4c64794dcb9",
	"total_amount": 4000,
	"payment_amount": 4000,
	"return_amount": 0
	}
	}

4.2 Perangkat dan lingkungan uji coba

Berdasarkan data komponen yang dibutuhkan, maka spesifikasi perangkat keras dan perangkat lunak yang akan digunakan adalah sebagai berikut:

Tabel 4. Spesifikasi *Hardware* dan *Software*

Nama Komponen	Detail Komponen	Spesifikasi
Raspberry Pi 4B	Sistem operasi	Ubuntu Server 22.04
	Processor	Quad Core Cortex A72 @1.8GHz
	Kapasitas RAM	4 GB
Digital Watt Meter	Tipe	-
Router Wifi	Tipe	ZTE F660

 ISSN:2302-8149
e-ISSN:2540-9719

Program	Bahasa Pemrograman	Golang
Tools Testing	Vegeta	HTTP Load Testing

Pada Tabel 4 merupakan spesifikasi hardware dan software yang digunakan dalam proses pengembangan dan pengujian sistem. Pengujian yang dilakukan yaitu Pengujian ketahanan Raspberry Pi. Pengujian dilakukan dengan jumlah request dari user yang berbeda-beda terhadap server, hal ini dilakukan untuk mengetahui ketahanan Raspberry Pi sebagai server dapat menangani beban permintaan dari aplikasi yang terinstall yaitu pada kondisi idle, trafik rendah dan trafik tinggi. Pengujian ini juga bertujuan untuk memastikan bahwa permintaan yang dikirim kepada server dimana sistem dipasang dapat diterima dan diproses dengan baik.

4.3 Pengujian Ketahanan Sistem

Pada pengujian ini dilakukan pengujian menggunakan *tool* vegeta untuk pengujian *http load testing*. Vegeta merupakan alat pengujian beban HTTP yang ditulis menggunakan bahasa pemrograman Go. Tujuan dari pengujian ini yaitu untuk mengetahui aplikasi yang berbasis HTTP merespon terhadap jumlah request diterima dalam waktu yang bersamaan. vegeta digunakan untuk menghasilkan jumlah permintaan per detik yang *continue* dan konstan untuk mengetahui ketahanan dari suatu layanan. perintah menjalankan permintaan dr sisi *client* melalui terminal vscode : go run cmd/load-test/*.go.

4.3.1 Skenario 1

Pada Gambar 6 menunjukkan skenario pengujian yang pertama. pada pengujian ini dilakukan uji coba selama 5 detik dengan jumlah request per detik yaitu 5 transaksi menambahkan ke keranjang dan 10 request untuk menampilkan barang. alamat IP server adalah 192.168.1.8. Dari skenario yang telah dilakukan diperoleh laporan yaitu total *request* adalah 25 dengan durasi respon seluruh transaksi adalah 4.8 s dan *latencies* 29.07ms. ratio tingkat kesuksesan pada skenario ini adalah 100% berhasil dengan status code 200. *throughput* pada skenario ini sebesar 5.17

```
const (
    testDurationInSeconds    = 5
    addToCartFreqPerSecond   = 5
    listOfGoodsFreqPerSecond = 10
    serverAddr               = "http://192.168.1.8:9900"
)
```

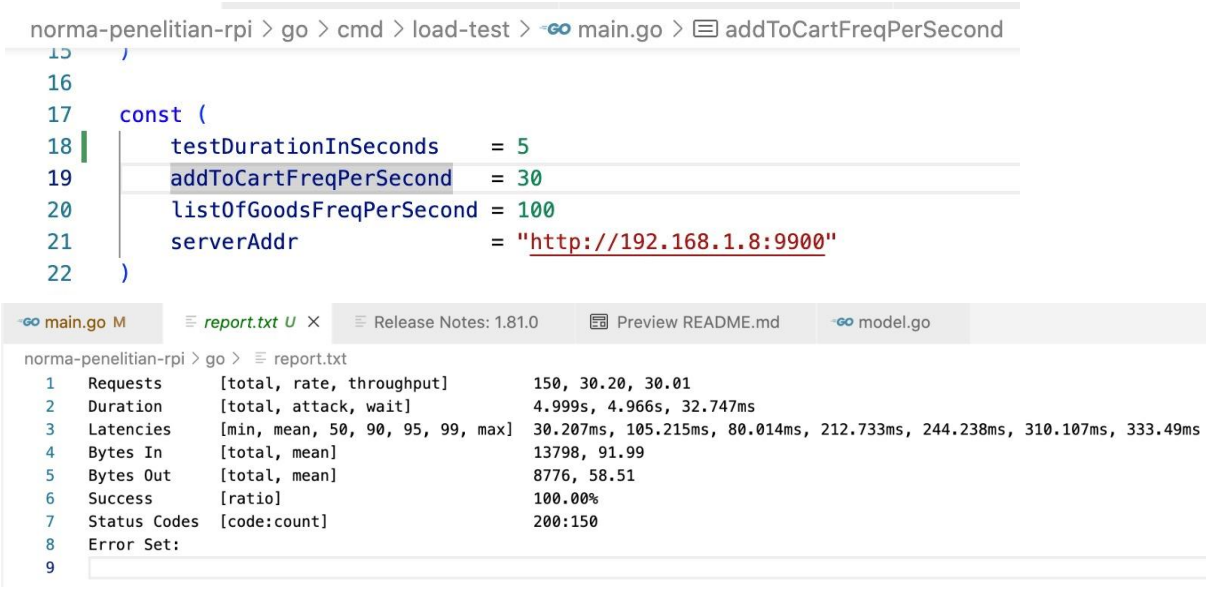
```
norma-penelitian-rpi > go > report.txt
1   Requests      [total, rate, throughput]         25, 5.21, 5.17
2   Duration      [total, attack, wait]             4.833s, 4.799s, 34.177ms
3   Latencies     [min, mean, 50, 90, 95, 99, max]  29.079ms, 37.947ms, 36.862ms, 43.128ms, 57.104ms, 68.391ms, 68.391ms
4   Bytes In      [total, mean]                     2326, 93.04
5   Bytes Out     [total, mean]                     1407, 56.28
6   Success       [ratio]                           100.00%
7   Status Codes  [code:count]                     200:25
8   Error Set:
9
```

Gambar 6. Skenario Pengujian 1 dan *Report*

4.3.2 Skenario 2

Pada Gambar 7 menunjukkan skenario pengujian yang kedua. pada pengujian ini dilakukan uji coba selama 5 detik dengan jumlah request per detik yaitu 30 transaksi menambahkan ke keranjang dan 100 request untuk menampilkan barang. alamat IP server adalah 192.168.1.8. Dari skenario yang telah dilakukan diperoleh laporan yaitu total *request* adalah 150 dengan durasi respon seluruh transaksi adalah 4.99 s dan *latencies* 30.2ms. ratio tingkat kesuksesan pada skenario ini adalah 100% berhasil dengan status code 200. *throughput* pada skenario ini sebesar 30.01

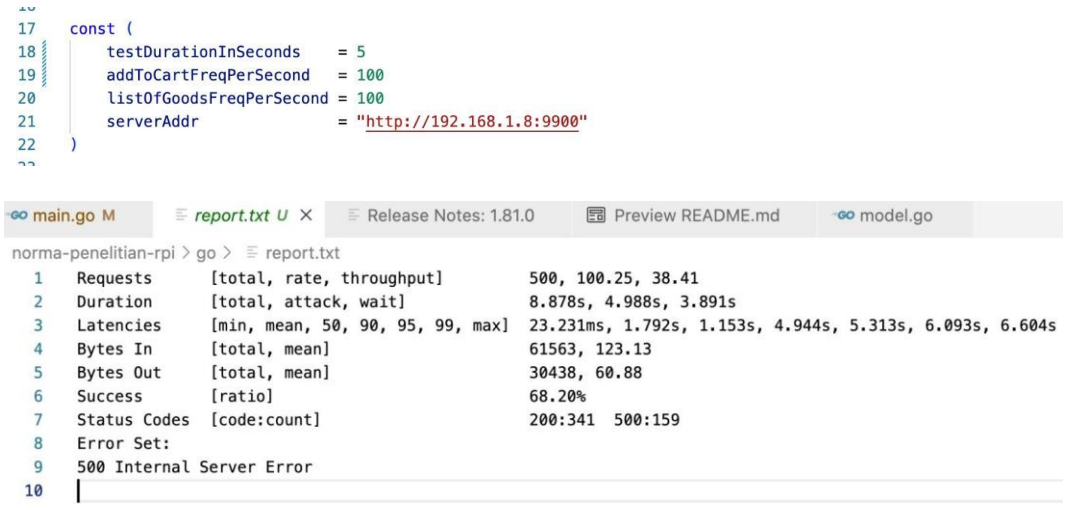
 ISSN:2302-8149
e-ISSN:2540-9719



Gambar 7. Skenario Pengujian 2 dan *Report*

4.3.3 Skenario 3

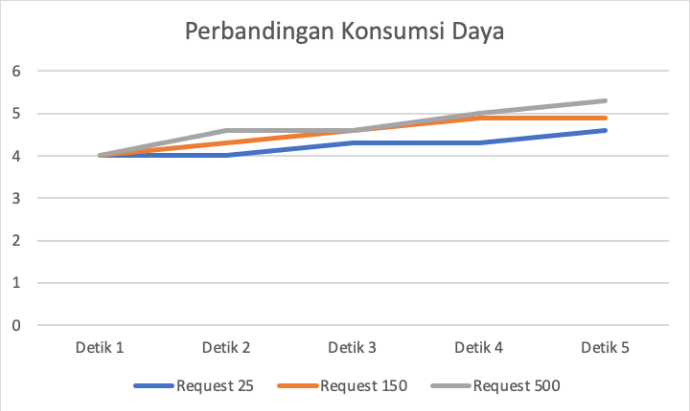
Pada Gambar 8 menunjukkan skenario pengujian yang ketiga. pada pengujian ini dilakukan uji coba selama 5 detik dengan jumlah request per detik yaitu 100 transaksi menambahkan ke keranjang dan 100 request untuk menampilkan barang. alamat IP server adalah 192.168.1.8. Dari skenario yang telah dilakukan diperoleh laporan yaitu total *request* adalah 500 dengan durasi respon seluruh transaksi adalah 8.87 s dan *latencies* 23.23ms. ratio tingkat kesuksesan pada skenario ini adalah 68% berhasil dengan status code 200 dan sisanya gagal dengan status code 500. *throughput* pada skenario ini sebesar 38.41



Gambar 8. Skenario Pengujian 3 dan *Report*

Pada Gambar 9 menunjukkan tingkat konsumsi daya yang dibutuhkan berdasarkan 3 skenario pengujian yang telah dilakukan yaitu 25, 150 dan 500 *customer* yang melakukan akses terhadap sistem. Dari diagram dapat dilihat bahwa semakin besar jumlah request maka semakin besar pula konsumsi daya yang dibutuhkan. nilai rata-rata daya yang dibutuhkan untuk masing-masing skenario selama 5 detik adalah 4.24, 4.54, 4.7 watt. nilai ini diperoleh dengan menggunakan alat voltmeter yang diaktifkan saat bersamaan terjadi request pada sistem.

 ISSN:2302-8149
e-ISSN:2540-9719



Gambar 9. Perbandingan Konsumsi Daya

5 Kesimpulan

Telah dibangun aplikasi backend UMKM menggunakan teknologi REST API yang diimplementasikan pada raspberry Pi sebagai server. Aplikasi ini dikembangkan dengan bahasa pemrograman golang, sistem operasi linux dan dijalankan dengan teknik container menggunakan docker. fungsionalitas aplikasi yang dikembangkan adalah melihat stok barang, menambahkan barang ke keranjang dan melakukan pembayaran sesuai transaksi. Sistem telah diuji menggunakan vegeta dan voltmeter. Dalam pengujian tersebut menggunakan jumlah request yaitu 25, 150 dan 500 selama 5 detik. Hasil yang didapat dalam pengujian ini adalah pada jumlah request 25 dan 150 transaksi 100% berhasil, namun pada jumlah request 500 terdapat transaksi yang gagal sebesar 32%. Konsumsi daya semakin meningkat sesuai dengan jumlah akses pada sistem dengan nilai rata-rata pada masing-masing skenario adalah 4.24, 4.54, 4.7 watt. Rest API dapat diimplementasikan untuk membuat aplikasi microservices dengan berbagai *platform*.

Ucapan Terima Kasih

Terima kasih kepada PPM (Penelitian dan Pengabdian Kepada Masyarakat) Politeknik Elektronika Negeri Surabaya yang telah mendanai dan memfasilitasi penelitian ini.

Referensi

[1] D.P Rahayu, D. Fardiaz, and A.N Aini, “Kajian Keberhasilan Program Pendampingan Usaha Mikro, Kecil, dan Menengah (Umkm) Pangan Direktorat Surveilan dan Penyuluhan Keamanan Pangan dalam Penerapan Prinsip Keamanan Pangan,” Institut Pertanian Bogor, Bogor, 2016.

[2] C.Y. Lin, “Success Factors of Small-andMedium-Sized Entreprises in Taiwan ,” *Journal of Small Business Management*, , vol. 36, no. 4, pp. 43–65, 1998.

[3] Y. Rahmini, S. Sekolah, T. Ilmu, and E. Balikpapan, “Perkembangan Umkm (Usaha Mikro Kecil Dan Menengah) Di Indonesia.”

[4] N. Ningsih, “Penerapan Konsep Gamification Pada Aplikasi E-Commerce Untuk Umkm Makanan Article Info Abstrak,” *JSAI : Journal Scientific and Applied Informatics*, vol. 4, no. 01, 2021, doi: 10.36085.

[5] K. Alia Akhmad, T. Maulidavito, dan Y. Tri Prakoso, “Penerapan Aplikasi Teknologi Informasi pada Usaha MikroKecil dan Menengah (UMKM),” 2022.

[6] R. Zhongshan, wan wei, and W. Gouquan, “Migrating Web Applications from Monolithic Structure to Microservices Architecture,” in *ACM the Tenth Asia-Pacific Symposium*, 2018.

[7] A. Khozaimi, Y. D. Pramudita, F. Solihin, and A. K. Ramadan, “Design And Development Of Backend Application For Thesis Management System Using Microservice Architecture And Restful Api,” vol. 11, no. 4, 2022.

[8] B. ; H. K. Dwi Pratomo, “Perancangan RESTful Web Service Satuan Kredit Partisipasi di Universitas Islam Indonesia,” *Seri Prosiding Seminar Nasional Dinamika Informatika*, vol. 4, no. 1, 2020.

[9] A. Belkhir, M. Abdellatif, R. Tighilt, N. Moha, Y. G. Gueheneuc, and E. Beaudry, "An observational study on the state of REST API uses in android mobile applications," in Proceedings ," in *IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems*, 2019, pp. 66–75.

[10] A. Zainudin *et al.*, "Backend Server System Design Based on REST API for Cashless Payment System on Retail Community," in *Proceedings, IES 2019 : IES, International Electronics Symposium : Surabaya, Indonesia, September 27-28, 2019 : the Role of Techno-intelligence in Creating an Open Energy System Towards Energy Democracy*, PENS.

[11] A. Mubariz *et al.*, "Perancangan Back-End Server Menggunakan Arsitektur Rest dan Platform Node.JS (Studi Kasus: Sistem Pendaftaran Ujian Masuk Politeknik Negeri Ujung Pandang)," 2020.

[12] S. Ayu Kartika, J. Pupuk Raya, and K. Timur, "Analisis Konsumsi Energi Dan Program Konservasi Energi (Studi Kasus: Gedung Perkantoran Dan Kompleks Perumahan Ti)".

[13] salih Fatma and S. A. O. Mysoon, "Raspberry pi as a Video Server," in *2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, IEEE, 2018.

[14] A.P. Jadhav and V.B. Malode, "Raspberry pi Based Offline Media Server," in *Proceedings of the 3rd International Conference on Computing Methodologies and Communication (ICCMC 2019) : 27-29, March 2019*, 2019.

[15] Sugiyatno and D. A. Prima, "Virtual Private Network (VPN) Secure Socket Tunneling Protocol (SSTP) Menggunakan Raspberry Pi," *Information System For Educators And Professionals*, vol. 2, no. 2, pp. 155–166, 2018.

[16] R. Choirudin and A. Adil, "Implementasi Rest Api Web Service dalam Membangun Aplikasi Multiplatform untuk Usaha Jasa," *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 18, no. 2, pp. 284–293, May 2019, doi: 10.30812/matrik.v18i2.407.

[17] M. M. Hidayat, R. D Adityo, and A. Siswanto, "Design of Restaurant Billing System (E Bill Resto) by Applying Synchronization of Data Billing in Branch Companies to Main Companies Based on Rest API," in *Proceeding - ICoSTA 2020: 2020 International Conference on Smart Technology and Applications: Empowering Industrial IoT by Implementing Green Technology for Sustainable Development*, Institute of Electrical and Electronics Engineers Inc., Feb. 2020. doi: 10.1109/ICoSTA48221.2020.1570615039.

[18] P.F Tanaem, *dkk*, "Penerapan RESFUL Web Service pada Disain Arsitektur Sistem Informasi pada Perguruan Tinggi (Studi Kasus: STARS UKSW)," *JASIEK (Jurnal Aplikasi Sains, Informasi, Elektronika dan Komputer)*, vol. 2, no. 1, Jun. 2020, doi: 10.26905/jasiek.v1i1.3098.

[19] L. Li, W. Chou, W. Zhou, and M. Luo, "Design Patterns and Extensibility of REST API for Networking Applications," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 154–167, Mar. 2016, doi: 10.1109/TNSM.2016.2516946.