p_ISNN: 2776-7418 e-ISSN 2746-508X
JIKI (JURNAL ILMU KOMPUTER DAN INFORMATIKA)
Vol. 4 No. 1, Juli, 2023, Hal. 77-88

IMPLEMENTASI ARSITEKTUR *MICROSERVICES* PADA APLIKASI *POINT OF SALE* TOKO FLYOVER STIKER

Siti Saidah[1], Latipaturachmaniah[2], Rahul Syaban[3]
[1,3] Sistem Informasi, Fakultas Ilmu Komputer Dan Teknologi Informasi
Universitas Gunadarma
[2] Teknik Informatika, Fakultas Teknik Industri, Universitas Gunadarma

[1,2,3] Jl. Margonda Raya No. 100, Depok 16424, Jawa Barat
[1]sitisaidah@staff.gunadarma.ac.id, [2]latifa_rachman@staff.gunadarma.ac.id
[3]rahulsyaban@student.gunadarma.ac.id

Abstrak*:* Perkembangan teknologi yang pesat memberikan kemudahan dalam mengimplementasikan teknologi yang baru untuk pengembangan aplikasi *(software*), baik dari sisi bahasa pemrograman *(programming languages*), kerangka kerja *(framework)* dan arsitektur aplikasi *(software architecture). Point of sale* (POS) merupakan sebuah software pencatatan transaksi penjualan yang bertujuan untuk mempermudah pencatatan suatu transaksi, melakukan kontrol persediaan barang dan monitoring laporan penjualan. Proses pengembangan aplikasi umumnya menggunakan arsitektur monolitik (*monolith*), arsitektur monolitik merupakan sebuah metode pengembangan software yang dimana software yang memiliki banyak fungsi atau fitur diletakan pada satu sistem software. Pengembangan software dengan menggunakan arsitektur monolith cocok digunakan untuk aplikasi skala kecil, namun sangat sulit dikembangkan, ketika skala aplikasi menjadi besar dibutuhkan waktu untuk developer memahami kode dan melakukan kompilasi aplikasi. Arsitektur *microservices* dapat menjadi sebuah solusi untuk memecahkan masalah kompleksitas pada pengembangan aplikasi yang menggunakan arsitektur monolitik dengan membagi beberapa layanan (*services*) besar menjadi layanan – layanan kecil. Penelitian ini bertujuan membuat aplikasi Implementasi Arsitektur Microservices Pada Aplikasi Point of Sales Toko Flyover Stiker. Adapun metode penelitian menggunakan tahapan *Systems Development Life Cycle* (SDLC) terdiri dari tahap perencanaan, analisis, perancangan, implementasi dan uji coba. Pembuatan aplikasi point of sale penjualan bahan stiker untuk toko “Flyover Stiker” dengan arsitektur microservice terdiri dari 6 kategori pelayanan yaitu service *user*, produk, order, *payment* dan kurir. Aplikasi microservices telah berhasil diimplementasikan dengan alamat *url https://flyoverstiker.online/*, aplikasi terbagi menjadi 6 layanan (service) dan telah berhasil dilakukan pengujian *end-to-end* secara otomatis dengan framework Cypress.

Kata Kunci*: Microservices, Point Of Sales, Monolitik, end-to-end testing, javascript*

Abstract*: Rapid technological developments make it easy to implement new technologies for application development (software), both in terms of programming languages, frameworks, and application architectures. Point of sale (POS) is software for recording sales transactions that aim to facilitate recording a transaction, controlling inventory, and monitoring sales reports. The application development process generally uses a monolithic architecture (monolith). Monolithic architecture is a software development method in which software with many functions or features is placed on one software system. Software development using the monolith architecture is suitable for small-scale applications, but it is very difficult to*

develop, when the application scale becomes large it takes time for developers to understand the code and compile applications. Microservices architecture can be a solution for solving complex problems in application development that uses a monolithic architecture by dividing several large services into small services. This study aims to create an application for the Implementation of Microservices Architecture in Point of Sales Applications for Sticker Flyover Stores. The research method uses the Systems Development Life Cycle (SDLC) stages of planning, analysis, design, implementation, and testing. Development of a point-of-sale application for the sale of sticker materials for "Flyover Sticker" stores with a microservice architecture consisting of 6 service categories, namely user service, product, order, payment, and courier. The microservices application has been successfully implemented with the address https://flyoverstiker.online/, the application is divided into 6 services and has been successfully tested end-to-end automatically with the Cypress framework.

***Keywords**: Microservices, Point Of Sales, Monolitik, end-to-end testing, javascript*

PENDAHULUAN

Implementasi teknologi dapat diterapkan pada segala bidang, salah satunya pada bidang usaha penjualan barang, baik untuk usaha kecil atau menengah. Bidang usaha salah satu teknologi yang dapat diterapkan yaitu pada proses pencatatan penjualan atau transaksi dengan menggunakan *software* bernama *point of sales*. *Point of sales (POS)* atau aplikasi kasir merupakan *software* pencatatan penjualan atau transaksi dengan aplikasi *point of sale* dapat mempercepat proses pencatatan suatu transaksi, mempermudah dalam melakukan kontrol persediaan barang dan mendapatkan data laporan dari tiap transaksi yang dibuat dengan cepat. Proses pengembangan aplikasi *(software)* menggunakan arsitektur monolitik pada jenis pembuatan *software* di mana sistem *software* tersebut memiliki fungsi yang banyak bahkan semua fungsi dapat dimasukkan dalam satu sistem *software* tersebut (Stefanus & Ardyansyah, 2021). Aplikasi dengan arsitektur monolitik sulit dikembangkan beriringan dengan bertambahnya fitur yang dikembangkan pada aplikasi menjadikan basis kode bertambah banyak dan kompleks. Salah satu masalah utama dalam pengembangan dengan arsitektur monolitik ketika aplikasi menjadi sangat kompleks dan sulit dipahami oleh *developer* yang berakibat, dalam memperbaiki bug atau menambahkan fitur baru menjadi sulit dan memakan banyak waktu (Richardson. C & Smith.F , 2016).

Arsitektur *microservices* dapat menjadi solusi untuk mengatasi terjadinya kompleksitas pada proses pengembangan aplikasi dengan menggunakan arsitektur monolitik. Proses pengembangan *software* arsitektur *microservices* dengan membagi beberapa fungsi atau fitur ke dalam fungsinya tersendiri.

Perumusan masalah dalam pembuatan aplikasi point of sale dengan arsitektur microservices terdiri dari bagaimana pembuatan aplikasi point of sale penjualan bahan stiker dengan arsitektur microservices? bagaimana implementasi arsitektur microservices dilakukan dengan membagi aplikasi menjadi dua bagian yaitu client *(front end)* untuk tampilan antarmuka aplikasi dan server *(back end)*? bagaimana implementasi arsitektur microservices sebagai penyedia *API* mampu melakukan perhitungan perkiraan biaya pengiriman barang menggunakan *API* Raja Ongkir dengan jasa layanan yang tersedia yaitu JNE, TIKI Dan POS? bagaimana Layanan pembayaran online *(Payment Gateway)* yang digunakan untuk pengujian dalam aplikasi pos adalah pembayaran dengan kartu kredit *(visa, master card)* dan transfer *(virtual account)* dan bagaimana metode pengujian menggunakan pengujian *end-to-*

end testing secara otomatis *(automated testing)* pada sisi client *(front end)*.

Tujuan penelitian ini adalah membuat aplikasi *point of sale* penjualan bahan stiker untuk toko "Flyover Stiker" menggunakan arsitektur *microservice* untuk mempermudah penjual dan pembeli pada saat penentuan perkiraan biaya pengiriman karena ada fitur penghitungan biaya pengiriman yang terhubung dengan *API* Raja Ongkir, juga menerapkan sistem pembayaran secara online dengan *payment gateway* untuk mempermudah proses transaksi penjualan serta pengujian dilakukan menggunakan metode *end-to-end testing* secara otomatis *(automated testing)* pada sisi client *(front end)*.

Kelebihan pengembangan aplikasi dengan menggunakan arsitektur *microservices* yaitu maintain *(maintenances)* tiap layanan *(service)* yang dipisah menjadikan proses *development*, *testing* dan proses hosting (deploy) menjadi mudah karena setiap layanan tersebut sederhana tidak menggunakan banyak *resource*, Produktivitas *(Productivity)*, Pemantapan Desain *(Design enforcement),* Enkapsulasi pengetahuan *(Knowledge encapsulation),* Tergantikan *(Replaceable), technology agnostic,* setiap layanan dapat menggunakan tools dan bahasa pemrograman yang berbeda-beda, performa *(performant)* setiap layanan dipisah menjadi layanan yang kecil dan ringan dan dapat ditingkatkan *(upgradeable)* setiap layanan dapat ditingkatkan secara terpisah dan tidak mengganggu layanan lain yang sedang berjalan.

Berdasarkan penelitian terdahulu, peneliti tertarik untuk mengamati penjualan stiker pada toko "flyover stiker" dengan membangun aplikasi point of sale dengan arsitektur *microservices* dengan menggunakan bahasa pemrograman javascript. Javascript mendukung pengembangan aplikasi baik untuk sisi client dan server, dengan memberikan banyak pilihan jenis library atau *framework* yang dapat digunakan. Implementasi arsitektur *microservices* menggunakan *framework* dari javascript yaitu express js, sedangkan untuk sisi client menggunakan library dari javascript yaitu react js.

Penelitian yang berjudul Simulasi Aplikasi *Microservices* Untuk Layanan Pendaftaran SIP Berbasis Website. Penelitian tersebut membahas bagaimana mengembangkan aplikasi layanan pendaftaran SIP dengan menerapkan arsitektur *microservices*, hasil penelitian tersebut telah berhasil membuat rancangan *microservices* yang baik dan memecahkan kompleksitas aplikasi yaitu dengan membagi layanan*(service)* sesuai dengan tugasnya masing – masing (Satria, Y., 2020).

Penelitian yang berjudul Refactoring Arsitektur *Microservices* Dengan Metode *Microservices* Migration Patterns Dan Strategi Rangler Pattern Pada Aplikasi Absensi PT Graha Usaha Teknik. Penelitian tersebut membahas bagaimana melakukan migrasi pada aplikasi yang dikembangkan dengan arsitektur monolith menjadi ke arsitektur *microservices*. Hasil penelitian berhasil melakukan migrasi dari arsitektur monolith ke *microservices* dan berdasarkan hasil pengujian pengembangan dengan arsitektur *microservices* lebih optimal dari segi kecepatan dalam mengirimkan atau mengembalikan data ke client *(response time)* (Mufrizal & Riski, 2020).

Peneitian yang berjudul Analisa Dan Implementasi *Microservice* Pada *Container* Menggunakan Docker, peneltian tesebut membahas bagaimana melakukan implemintasi arsitektur microservice pada container menggunakan docker, hasil penelitian tersebut behasil mengatasi masalah skalabilitas dan masalah lainnya yang terdapat pada arsitektur monolitis. (Stefanus & Ardyansyah, 2021).

KAJIAN PUSTAKA

Microservices

Microservices atau disebut arsitektur microservices merupakan gaya arsitektur untuk pengembangan aplikasi dengan membagi layanan *(services)* besar menjadi service - service kecil. Tujuan utama penggunaan arsitektur microservice adalah memecahkan layanan yang kompleks menjadi lebih kecil dan

sederhana untuk tujuan fungsionalitas umum. Setiap service yang pisahkan harus lebih sederhana atau kecil bertujuan untuk memudahkan dalaman *maintenance*, *develop*, *tested* dan mudah di *scale* (Resende D & Osman P,. 2019).
Beberapa hal yang harus diperhatikan dalam pembuatan aplikasi dengan arsitektur *microservice* yaitu:
1. Ketergantungan *(Dependencies)*
Arsitektur microservices adalah teknologi agnostic, ketergantungan berbeda untuk setiap layanan yang berbeda mungkin muncul.
2. Kompleksitas *(Complexity)*
Aplikasi yang kecil, kompleksitas bootstrap lebih besar dibandingkan dengan monolitik
3. Pengujian *(End to end testing)*
Menjadi lebih kompleks untuk pengujian *end to end test*, karena jumlah service untuk saling terhubung pasti lebih besar dari pada aplikasi dengan arsitektur monolitik.

Software Development Life Cycle (SDLC)

SDLC adalah metode pengembangan aplikasi yang umum digunakan dalam membangun atau mengembangkan sistem informasi. Rekayasa perangkat lunak, konsep SDLC mendasari berbagai jenis metodologi pengembangan perangkat lunak. Metodologi-metodologi ini membentuk suatu kerangka kerja untuk perencanaan dan pengendalian pembuatan sistem informasi, yaitu proses pengembangan perangkat lunak (Susanto, R dan Andriana, A, 2016). Terdapat beberapa model dalam pengembangan dengan metode SDLC yaitu *waterfall, prototype, interative, rapid application development* dan lainnya. Tahapan dalam pengembangan aplikasi menggunakan metode SDLC dengan model *waterfall* yaitu:
1. Analisis, terdiri dari analisis kebutuhan fungsional dan non-fungsional.
2. Perancangan, melakukan pembuatan rancangan sistem UML, baik dari alur sistem, rancangan database dan rancangan *user* interface.
3. Implementasi menggunakan koding bahasa pemrograman Javascript terhadap racangan sistem yang telah dibuat sebelumnya.
4. Pengujian dilakukan pengujian *end to end* terhadap sistem yang telah dibuat.
5. Pemeliharaan *(maintenance) m*elakukan perawatan dan monitoring terhadap aplikasi yang telah berjalan.

Unified Modeling Language (UML)

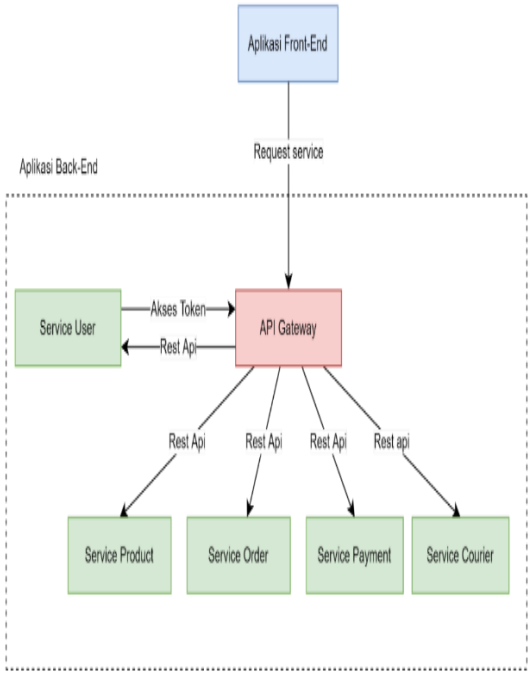
Unified Modeling Language adalah sebuah teknik pengembangan sistem yang menggunakan bahasa grafis sebagai alat untuk mendokumentasi dan melakukan spesifikasi pada sistem (Mulyani S,. 2017). UML adalah metode perancangan untuk pengembangan sistem yang berorientasi object dan juga sebagai alat pendukung perancangan sistem. Terdapat beberapa jenis diagram UML yang dapat digunakan sebagai alat perancangan sistem antara lain adalah *Use Case* Diagram, *Activity* Diagram, *Sequence* Diagram dan Class Diagram.

METODE

Metode penelitian yang digunakan dalam pembuatan aplikasi Point of sale dengan arsitektur *microservices* menggunakan SDLC *(Software Development Life Cycle).* Beberapa tahapan dalam metode SDLC meliputi :
1. Analisis
Tahap analisis dilakukan pendefinisian dan memahami segala kebutuhan untuk pembuatan aplikasi point of sale dengan Arsitektur Microservices serta dilakukan percarian infromasi terkait arsitektur microservices dengan cara studi pustaka seperti mengumpulkan jurnal dan artikel.
2. Perancangan
Tahap perancangan dilakukan pembuatan rancangan untuk membangun aplikasi mulai dari rancangan model untuk arsitektur *microservices*, rancangan UML seperti, *use case diagram*, *activity diagram, sequence diagram* dan *class diagram*, rancangan database untuk tiap services pada arsitektur *microservices* dan rancangan tampilan antarmuka *(User Interface)* aplikasi.

p_ISNN: 2776-7418 e-ISSN 2746-508X
JIKI (JURNAL ILMU KOMPUTER DAN INFORMATIKA)
Vol. 4 No. 1, Juli, 2023, Hal. 77-88

a. Model Arsitektur *Microservices*
Berikut merupakan rancangan model arsitektur *microservices* yang dibuat dengan membagi layanan menjadi 6 yaitu layanan *API-gateway*, *user*, produk, order, payment, dan kurir.



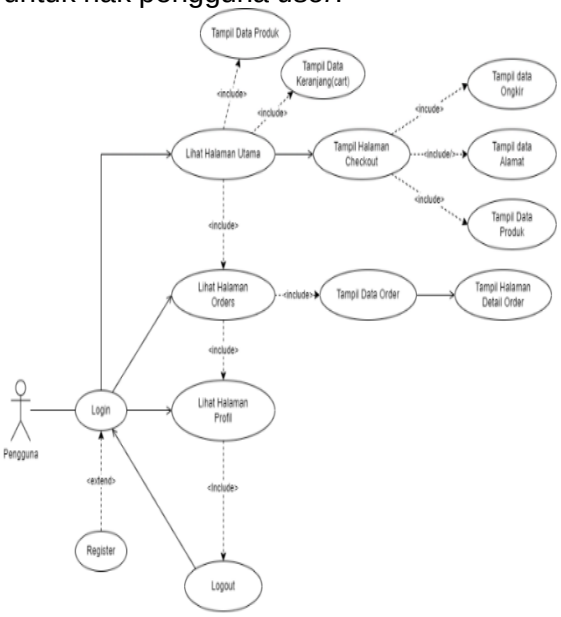
Gambar 1. Model Arsitektur *Microservices*

Gambar 1 Merupakan rancangan model Arsitektur *Microservices* untuk aplikasi POS. Setiap permintaan *(request)* yang dilakukan dari aplikasi Front-end merujuk pada layanan *(service)* yang bernama API Gateway. API Gateway berfungsi sebagai perantara antara request dari aplikasi *front-end* ke service - service yang dipanggil oleh *user* dari aplikasi *front-end*. Komunikasi antara aplikasi Front end, API gateway dan service - service lainnya dalam Arsitektur *Microservices* menggunakan Rest *(Representational State Transfer)* merupakan bentuk komunikasi antar web service melalui HTTP (*Hypertext Transfer Protocol)*.

b. Rancangan UML
Rancangan UML terdiri rancangan *use case diagram*, *activity diagram, sequence diagram* dan *class diagram*

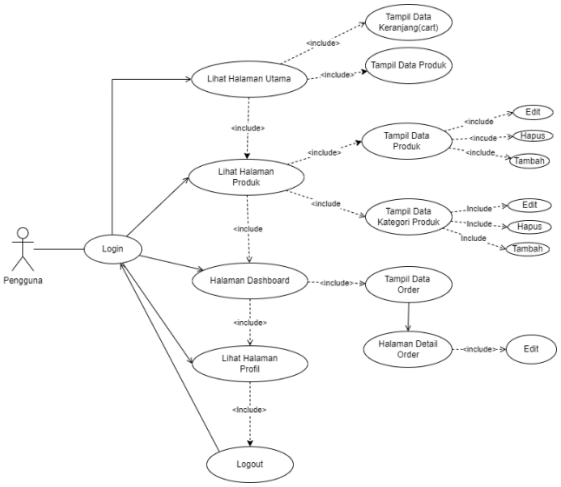
- Use Case Diagram

Use case diagram terdiri dari use case untuk hak pengguna *user* dan admin.
Berikut merupakan rancangan use case untuk hak pengguna *user*.



Gambar 2. Use Case *User*

Gambar 2 merupakan use case untuk hak akses *user*, setelah *user* melakukan login / daftar akun, *user* dapat melakukan transaksi pemesanan produk pada halaman utama. *User* dapat mengakses dan melihat status data transksi yang telah dilakukan sebelumnya. *User* juga dapat melakukan pengubahan data profil seperti data alamat, email dan *password* pada halaman profil. Berikut merupakan rancangan use case untuk hak pengguna admin.

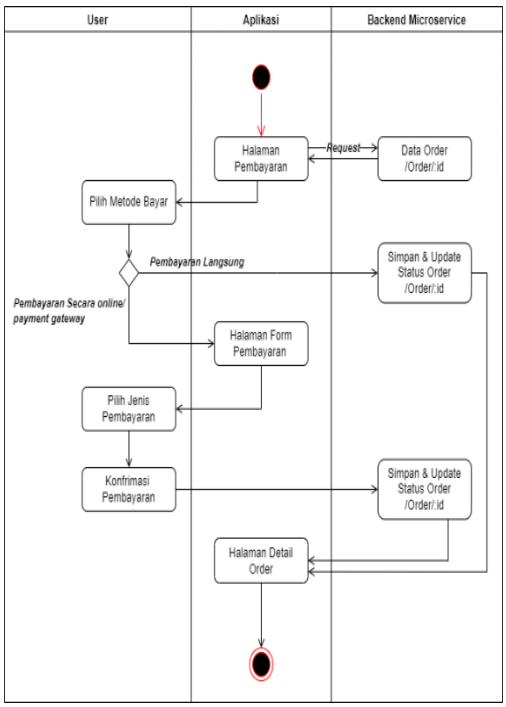


Gambar 3. Use Case Admin

Gambar 3 merupakan use case untuk pengguna dengan hak akses pengguna admin. Admin dapat melakukan transaksi pemesanan produk, jika *user* atau pembeli melakukan transaksi secara langsung (offline) pada toko dengan pilihan pemesanan pada aplikasi adalah pembayaran langsung. Admin dapat melakukan manajemen produk dan kategori untuk mengatur jumlah stok barang dan jenis kategori barang pada halaman produk. Admin dapat memanajemen data pesanan yang dilakukan oleh *user* secara online seperti mengubah status pemesanan dan menambahkan nomor resi pengiriman pada halaman dashboard. Admin dapat mengubah data alamat, email dan password pada halaman profil.

- Activity Diagram
  Activity diagram menggambarkan bagaimana alur aktivitas yang terjadi dalam sistem, bagaimana tiap - tiap aktivitas saling berinteraksi dengan aktivitas lain dan sampai bagaimana berakhirnya suatu aktivitas. Berikut merupakan activity untuk proses order dan proses pembayaran.

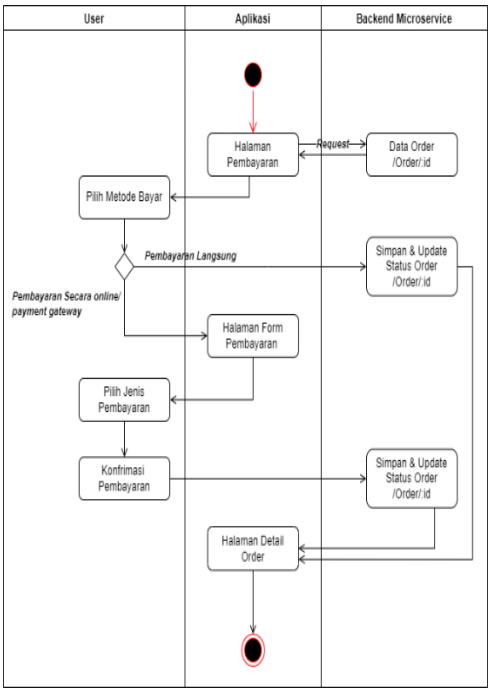
Activity Proses Order.



Gambar 4. Activity Proses Order

Gambar 4 merupakan bentuk activity untuk proses order. Setelah *user* login, pada aplikasi ditampilkan halaman utama pada halaman tersebut dilakukan request data produk kepada backend *microservices* dan ditampilkan pada aplikasi, lalu *user* dapat memilih produk yang ingin dipesan. Halaman checkout melakukan request ke backend *microservices* untuk data produk, *user* dan estimasi pengiriman lalu *user* memilih jenis jasa pengiriman, konfirmasi data yang dipesan dan backend *microservices* memproses menyimpan data order, setelah order berhasil dirahkan pada halaman pembayaran dan activity berakhir.

Activity Proses Pembayaran



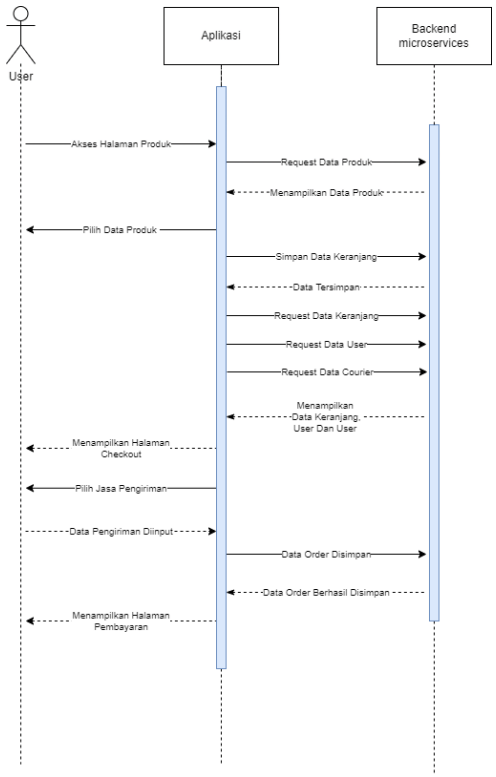
Gambar 5. Activity proses Pembayaran

Gambar 5 merupakan bentuk activity untuk pembayaran. Activity dimulai pada halaman pembayaran dengan meminta data order dari backend *microservices* lalu *user* diminta untuk memilih metode pembayaran. Opsi metode bayar terbagi menjadi dua yaitu pembayaran secara langsung dan melalui payment gateway, jika memilih pembayaran langsung *user* melakukan pembayaran melalui toko secara offline yang diproses oleh admin

dan, jika metode pembayaran melalui payment gateway *user* akan diarahkan ke halaman form pembayaran lalu *user* pilih jenis pembayaran, konfirmasi pembayaran, dan backend *microservices* secara otomatis melakukan update status order setelah sukses ditampilkan halaman detail order dan activity berakhir.

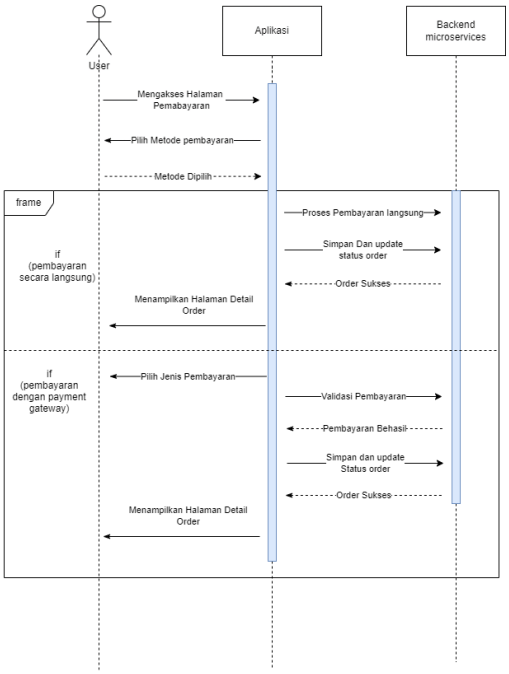
- Sequence Diagram
  Sequence diagram menggambarkan bagaimana alur interaksi antara object yang terjadi pada aplikasi secara dinamis.

Sequence Diagram Proses Order



Gambar 6. Sequence Diagram Order

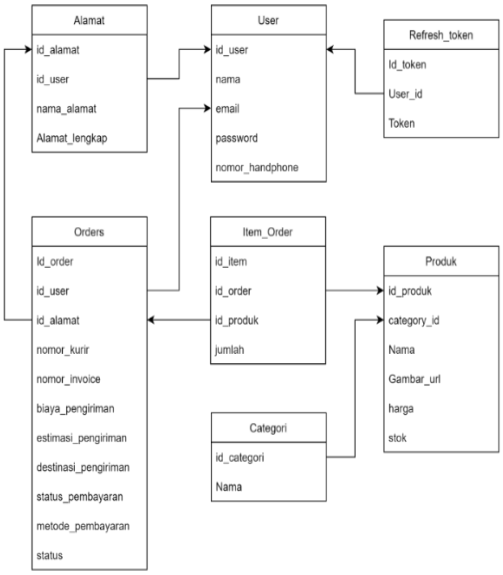
Gambar 6 merupakan sequence diagram untuk proses order. Pertama kali *user* ditampilkan halaman produk / utama. *User* diminta untuk memilih data produk yang ingin dipesan, setelah memilih produk yang dipesan, lalu bagian back end microservice menyimpan data produk, setelah data berhasil disimpan backend microservice melakukan request data keranjang, *user* dan kurir lalu *user* diarahkan pada halaman checkout lalu *user* diminta untuk memilih jasa pengiriman, setelah *user* memilih jasa pengiriman lalu data order disimpan oleh backend microservices, setelah data order disimpan pada aplikasi akan ditampilkan halaman pembayaran.
Sequence Diagram Proses Pembayaran



Gambar 7. Sequence Diagram Pembayaran

Gambar 7 merupakan proses pembayaran dilakukan setelah *user* melakukan pembuatan order pada halaman produk. Petama kali *user* diminta untuk memilih metode pembayaran, jenis metode pembayaran yang disediakan yaitu pembayaran secara langsung yang dilakukan secara langsung pada toko dan pembayaran secara online *(payment gateway)*.
Pilihan pembayaran secara langsung pada backend microservice secara otomatis melakukan penyimpanan data order dan mengupdate status pemesanan dan jika memilih pembayaran dengan *payment gateway user* diminta memilih jenis pembayaran yang disedikan oleh *payment gateway* setelah memilih jenis pembayaran secara otomatis back end microservice mem-validasi pembayaran dan merubah status pembayaran, lalu *user* diarahkan pada halaman detail order.

- Class Diagram
  Class diagram *microservices* menggambarkan bentuk interaksi antara class yang saling berkaitan pada aplikasi *microservices*.



Gambar 8. Class Diagram *Microservices*

Berdasarkan gambar 8. class diagram terdapat tujuh class yang saling terhubung. Class *user* memiliki atribut id_*user*, nama_*user*, email, password dan nomor_handphone, Class refresh_token memiliki atribut yang tediri dari id_token, *user*_id dan token, class alamat terdiri dari atribut id_alamat, id_*user*, alamat_kirim dan kodepos, class produk terdiri dari id_produk, id_category, nama_produk, gambar, harga dan stok, class category terdiri dari id_categori dan nama_category, class order terdiri dari id_order, id_*user*, id_alamat, nomor_kurir, nomor_invoice, biaya_pengiriman, estimasi_pengiriman, destination_pengiriman, status_ pembayaran, metode_pembayaran dan status. Class Order_item terdiri dari atribut id_item, id_produk, id_order dan jumlah.

3. Implementasi
   Tahap implementasi dilakukan implementasi berdasarkan rancangan yang telah dibuat. Implementasi arsitektur *microservices* pada sisi server *(back end),* membuat dokumentasi API untuk digunakan pada sisi client *(front end)* dan implementasi tampilan antarmuka *(User Interface)* pada sisi client *(front end)*. Proses implementasi pembuatan aplikasi menggunakan bahasa pemrograman javascript, baik untuk sisi *backend* dan *frontend*.
4. Uji Coba
   Tahapan ini dilakukan uji coba aplikasi pada bagian *front end* dengan metode *end-to-end testing* secara otomatis *(automated testing)* menggunakan *framework testing* dari javascript yaitu *Cypress*.
5. Perawatan *(Maintenance)*
   Tahapan maintenance dilakukan perawatan aplikasi, baik untuk sisi front end dan backend serta dilakukan monitoring untuk setiap layanan *(service)* yang telah berjalan untuk masing – masing service.

HASIL DAN PEMBAHASAN

1. Implementasi

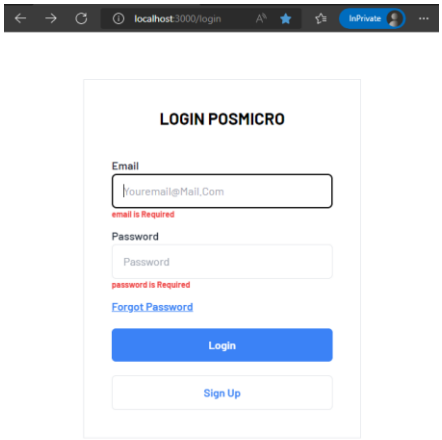
Implementasi pembuatan Arsitektur *Microservices* melalui beberapa tahapan yaitu pembuatan aplikasi untuk sisi server *(back-end)* yang berisi layanan - layanan *(services)* kecil yang dibuat dalam bentuk Rest-API yang dapat digunakan pada aplikasi sisi client *(front-end)* dan pembuatan aplikasi *front-end* dalam bentuk tampilan aplikasi yang dapat dilihat oleh pengguna, serta melakukan pengujian *white box testing* dengan menggunakan metode *end-to-end testing*.

1.1 Implementasi Tampilan Aplikasi

Hasil output pada *back-end* Arsitektur *Microservices* berupa API yang berisi *endpoint* yang dapat dipahami oleh seorang *developer*, agar pengguna dapat berinteraksi dengan aplikasi diperlukan tampilan antarmuka aplikasi. Implementasi tampilan yang dibuat adalah tampilan *Login, Register, Beranda (utama), Products, Shipment, Orders*, *Detail Order*, *Dashboard* dan *Profile*.

1.1.1 Tampilan Halaman Login

Halaman Login merupakan halaman yang ditampilkan sebelum pengguna atau admin dapat masuk aplikasi.

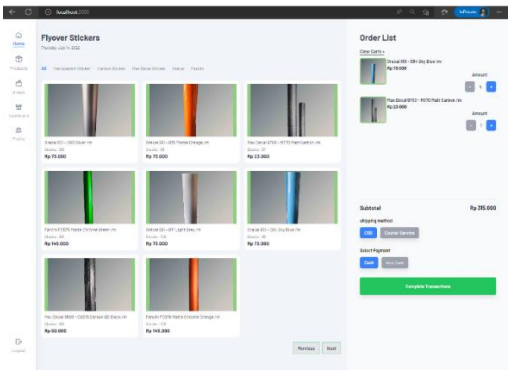


Gambar 9. Tampilan Halaman Login

Penjelasan gambar 9 merupakan bentuk output tampilan untuk halaman login dan telah ditambahkan validasi untuk setiap inputan pada form login.

1.1.2 Halaman Beranda

Halaman beranda ditampilkan setelah pengguna melakukan *login*, pada halaman beranda ditampilkan data produk dalam bentuk komponen *card* dan terdapat komponen keranjang yang berisi draft data produk yang ingin dipesan.

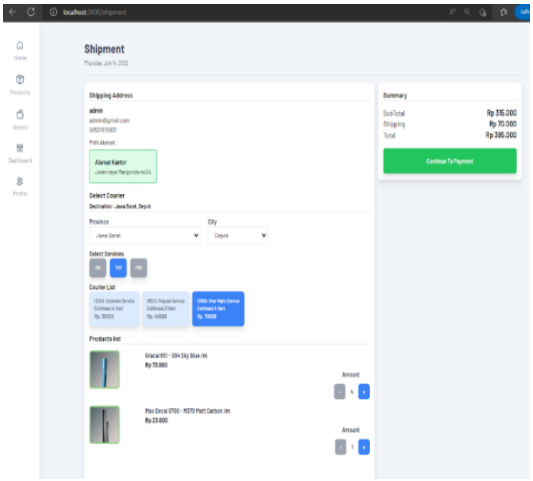


Gambar 10. Tampilan Halaman Beranda

Gambar 10 merupakan output untuk tampilan halaman beranda, pada halaman beranda ditampilkan data produk dan data keranjang serta metode jenis pembayarannya.

1.1.3 Halaman Pengiriman

Halaman pengiriman ditampilkan ketika pengguna memilih menggunakan jasa pengiriman *(courier service*) pada saat melakukan order di halaman beranda.

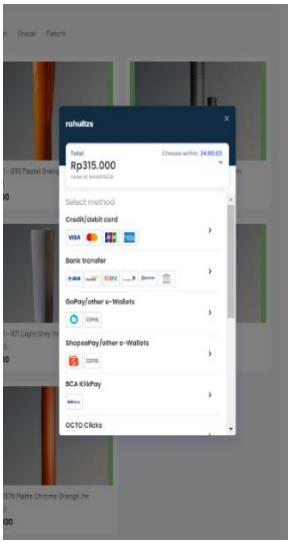


Gambar 11. Tampilan Halaman Pengiriman

Gambar 11 merupakan tampilan output halaman pengiriman yang berisikan data alamat pemesan, form untuk memilih jasa kurir, data produk yang dipesan dan rincian jumlah pembelian barang.

1.1.4 Tampilan Form pembayaran

Tampilan form pembayaran merupakan tampilan yang disedakan oleh penyedia layanan *Payment Gateway* yaitu Midtrans yang digunakan untuk menangani pembayaran secara online.

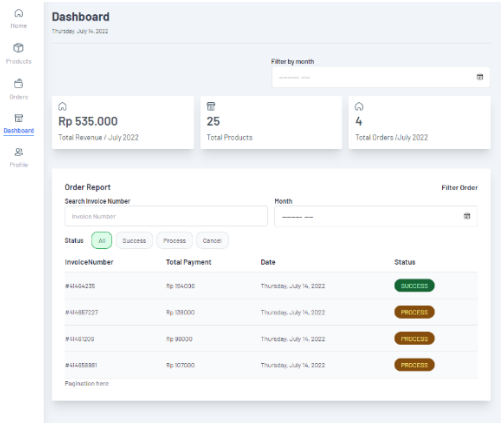


Gambar 12. Tampilan Form Pembayaran

Gambar 12 form pembayaran berisikan list metode pembayaran yaitu pembayaran memalui *e-wallet*, transfer melalui bank dan kartu kredit pada pengujian, metode pembayaran yang digunakan adalah

menggunakan transfer melalui bank dan kartu kredit.

1.1.5 Halaman Dashboard
Halaman Dashboard berisi data laporan pemesanan, stok produk dan jumlah pendapatan bulanan.



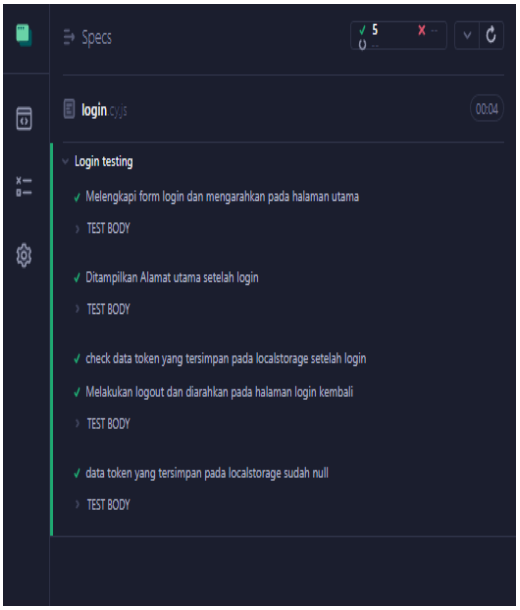
Gambar 13. Tampilan Halaman Dashboard

Gambar 13 halaman dashboard ditampilkan komponen dalam bentuk card yang berisi data total penjualan bulan, jumlah produk, dan jumlah transaksi, serta ditampilkan komponen table yang berisi semua data order yang telah dibuat oleh user dan terdapat komponen untuk mengurutkan data order berdasarkan nomor_invoice, status_order dan tanggal_order.

2. Pengujian

Metode pengujian yang digunakan adalah pengujian *end-to-end testing* secara otomatis *(automated testing)* dengan menggunakan *framework* testing yaitu Cypress. Pengujian di aplikasi front-end, Cypress dipasangkan pada *framework* web yaitu React js. Tahapan pengujian yang dilakukan yaitu persiapan pengujian *testing* dan membuat test case dan pengujian.
Pengujian dilakukan pada setiap halaman pada aplikasi *front-end*. Test case yang dibuat yaitu test *login, register*, membuat *order*, pengiriman, halaman order, produk, kategori, halaman profil dan alamat serta halaman dashboard. Berikut merupakan output hasil pengujian pada aplikasi untuk sisi *front end*.

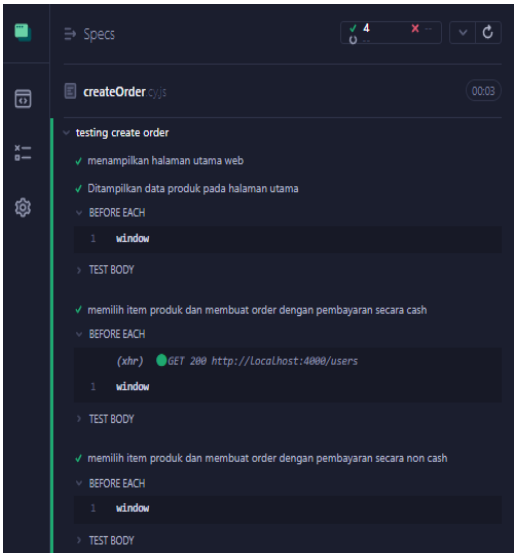
2.1 Pengujian Login
Pengujian yang dilakukan adalah pengujian pada form inputan *login* dan mencoba untuk melakukan *login* yang sudah terhubung dengan *endpoint* API *login*. Berikut merupakan output hasil pengujian *login*.



Gambar 14. Hasil Pengujian Login

Gambar 14 menampilkan hasil test case yang dibuat untuk halaman Hasil Pengujian *Login* telah berhasil memenuhi testing yang dibuat.

2.2 Pengujian Membuat Pesanan
Pengujian membuat pesanan dilakukan pada halaman utama aplikasi *front-end*, testing yang dilakukan adalah membuat pemesanan dengan pembayaran cash dan non-cash. Berikut merupakan output hasil pengujian membuat pesanan.

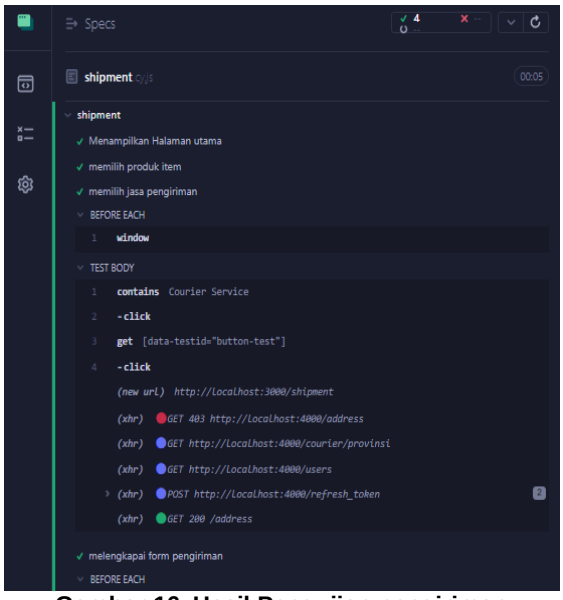


Gambar 15. Hasil Pengujian Membuat Pesanan

Gambar 15 menampilkan hasil output yang dari test case Halaman Hasil Pengujian Membuat Pesanan telah berhasil memenuhi test case yang dibuat.

2.3 Pengujian Halaman Pengiriman

Pengujian yang dilakukan pada halaman pengiriman adalah pengisian form data pengiriman, membuat pemesanan dan pengujian pada *endpoint* API data kurir, alamat *payment* dan data pengguna. Berikut merupakan output hasil pengujian membuat pengiriman. Berikut merupakan output hasil pengujian pengiriman.



Gambar 16. Hasil Pengujian pengiriman

Penjelasan gambar 16. hasil pengujian memenuhi test case yang dibuat dan telah berhasil melakukan testing yang sudah terintegrasi dengan endpoint API backend Microservices. Endpoint API yang digunakan yaitu data kurir, alamat, payment dan data *user*

Kesimpulan Dan Saran

1. Kesimpulan

Berdasarkan hasil analisis, perancangan, implementasi dan pengujian yang telah dilakukan, disimpulkan bawah Implementasi Arsitektur *Microservices* aplikasi POS *(point of sale)* pada toko Flyover Sticker telah berhasil dilakukan. Aplikasi *Microservices* dibagi menjadi dua bagian yaitu aplikasi *front end* dan aplikasi *backend* yang terdiri dari beberapa layanan service yang telah dibagi menjadi 6 (enam) buah layanan kecil.

Aplikasi *Microservices* telah diterapkan keamanan untuk tiap *service* dengan menerapkan sistem tokenisasi sebagai autentikasi untuk mengakses suatu *service*.

Testing dilakukan dengan metode *end-to-end testing* secara otomatis *(automated testing)* menggunakan *framework testing* yaitu *Cypress*.

Pengujian yang dilakukan yaitu proses login, registrasi, menampilkan halaman utama, melakukan manajemen data produk dan kategori, melakukan proses transaksi, menampilkan *pop-up snap* pembayaran *payment gateway* dari midtrans, menampilkan data pengiriman *(shipment)*, menampilkan data pemesanan, menampilkan detail data pesanan, melakukan manajemen data profil, menampilkan data laporan penjualan dan melakukan proses logout. Berdasarkan pengujian *end-to-end* yang dilakukan, *test case* yang dibuat telah berhasil memenuhi kebutuhan untuk setiap *test case* yang dibuat pada setiap halaman aplikasi.

2. Saran

Implementasi arsitektur *microservices* aplikasi POS *(Point Of Sale)* dapat dikembangkan lebih lanjut, baik untuk sisi

front end atau *back end*, pada sisi *back end* belum dilakukannya pengujian unit *(unit testing)* untuk setiap layanan yang dibuat.
Terdapat masalah pada pengiriman *email* ketika *user* melakukan *request* lupa password, yaitu email yang dikirimkan dari *service users* masuk pada bagian spam email.
Belum adanya fitur notifikasi untuk status pembayaran yang berhasil pada aplikasi, saat ini status pembayaran berhasil hanya dikirimkan melalui *email* oleh layanan *payment gateway midtrans,* diharapkan pada pengembangan selanjutnya dapat menerapkan fitur yang belum pada aplikasi.

REFERENSI

[1] Fajrin, R. (2017). *Pengembangan Sistem Informasi Geografis Berbasis Node. JS untuk Pemetaan Mesin dan Tracking Engineer dengan Pemanfaatan Geolocation pada PT IBM Indonesia.* Jurnal Komputer Terapan, 3(1), 33-40.

[2] GBB995126, B. N., Wilson, M. A., & Park, M. (2019) *Building Microservices with JavaScript. Birmingham* : [online resource]

[3] Richardson, C., & Smith, F. (2016). *Microservices: from design to deployment*. Nginx Inc, *1*, 24-31.

[4] Fitri, R., Kom, S., & Kom, M. (2020). *Pemrograman Basis Data Menggunakan MySQL*. Deepublish.

[5] Susanto Anna Dara Andriana, R. (2016). *Perbandingan model waterfall dan prototyping untuk pengembangan sistem informasi*. Majalah Ilmiah UNIKOM.

[6] Mulyani, S. (2017). *Analisis dan Perancangan Sistem Informasi Manajemen Keuangan Daerah: Notasi Pemodelan Unified Modeling Language (UML)*. Abdi sistematika.

[7] Prasetyo, S. E., & Wijaya, A. (2021, March). *Analisa dan Implementasi Microservice pada Container Menggunakan Docker*. In CoMBInES-Conference on Management, Business, Innovation, Education and Social Sciences (Vol. 1, No. 1, pp. 557-564).

[8] Urva, G., & Siregar, H. F. (2015).*Pemodelan uml e-marketing minyak goreng*. *Jurteksi Royal Edisi2*.