APLIKASI *TAKING ORDER (FRONT END)* MENGGUNAKAN *MICROSERVICE* PADA PT. XYZ

Parjiono[1], Septafiansyah Dwi Putra[2], Tri Sandhika Jaya[3]
[1] mahasiswa jurusan ekonomi dan bisnis, [2] pembimbing 1, [3] pembimbing 2

Abstrak

Penelitian ini bertujuan untuk : (1) merancang dan membuat aplikasi *taking order (front end)* menggunakan *microservice* pada PT. XYZ, (2) membangun *e-commerce* dengan layanan *microservice* arsitektur, (3) pengembangan perangkat lunak dengan pendekatan pemrograman berbasis teks dengan konsep *Object Oriented Programming* (OOP).

Metode yang digunakan adalah *Design Science Research Methodology* (DSRM). Tujuan DSRM adalah untuk memecahkan masalah dengan membuat penemuan baru dan pengetahuan desain yang relevan dalam konteks penelitian. Pengembangan sistem informasi dan perangkat lunak yang memiliki model efesiensi waktu dan sumber daya dengan tahapan meliputi *requirement planning*, *user design*, *construction* dan *cutover*.

Hasil dari penelitian ini adalah: (1) Aplikasi *Taking Order (Front End)* Menggunakan *Microservice* Pada PT. XYZ, (2) Hasil pengujian perbandingan *web* aplikasi yang dibangun antara *monolithic* dengan *microservices* arsitektur. Sehingga dapat disimpulkan bahwa kualitas perangkat lunak sudah memenuhi kriteria perangkat lunak.

Kata Kunci: *Microservice, Object Oriented Programming, RAD.*

PENDAHULUAN

Pesatnya perkembangan internet menjadi salah satu infrastruktur komunikasi yang termurah dan dengan tingkat penerimaan yang luas, maka penggunaan internet sebagai fasilitas pendukung dan bahkan sebagai urat nadi bisnis menjadi semakin nyata keunggulannya. Salah satu tren yang menyertai bisnis dalam jaringan internet adalah *e-commerce* baik *business-to-customer* maupun *busines-to-business*. Dengan membawa keunggulan *internet* seperti pelayanan 24 jam, akses dari segala penjuru dengan biaya yang relatif murah dan kemudahan-kemudahan lainnya, maka tidaklah mengherankan jika sekarang banyak organisasi bisnis yang merambah ke dalam *e-commerce*.

E-commerce atau bisa disebut perdagangan elektronik atau *ebussines* adalah penyebaran, pembelian, penjualan, pemasaran barang dan jasa melalui internet atau jaringan komputer. Perkembangan internet yang semakin maju merupakan salah satu faktor pendorong berkembangnya *ecommerce*. Saat ini, *e-commerce* adalah salah satu saluran transaksi yang paling penting. Orang, perusahaan, dan pemerintah semua berpartisipasi dalam transaksi *e-commerce* (Huang et al., 2019).

Saat ini terdapat pendekatan teknologi baru yaitu *microservice* yang dapat diterapkan dalam *sistem e-commerce*. Salah satu tujuan arsitektur *microservice* adalah untuk mengatasi keterbatasan kemampuan *monolithic* arsitektur. *Monolithic* arsitektur merupakan suatu sistem yang memiliki arsitektur *microservices* ketika sistem itu terdiri dari banyak layanan mikro yang saling berkolaborasi untuk menyajikan data yang relevan dalam layanan *e-commerce*. Berdasarkan survey data presentasi perusahaan yang menerapkan *microservices* yaitu otto.de, otto.de adalah salah satu toko *online* terbesar di Eropa yang memiliki omset bisnis *e-commerce* lebih dari 2,563 miliar Euro di tahun 2015/2016 dan hingga 1 juta pengunjung per hari yang mengakses layanan *e-commerce* otto.de (Hasselbring and Steinacker, 2017).

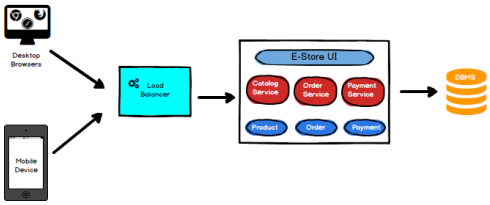
TINJAUAN PUSTAKA

Konsep dalam pemrograman sangat penting untuk pengembangan perangkat lunak, *Object Oriented Programming* (OOP). OOP merupakan konsep dalam pemrograman yang begitu penting untuk pengembangan perangkat lunak modern, konsep OOP didasarkan pada konsep *object* dengan demikian, konsep berbasis *object* bermanfaat untuk desain dan pengembangan perangkat lunak. Pemrograman OOP memiliki enam konsep penting untuk mendukung fitur-fitur OOP, yaitu *Class*, *Object*, *Method*, *Encapsulation*, *Inheritance*, dan *Polimorfisme* (Su and Hsu, 2017).

A. Konsep *monolithic* arsitektur

Dalam konsep *monolithic* arsitektur, modularisasi bergantung pada pembagian sumber daya dari mesin yang sama *memori*, *basisdata*, atau *file* dan komponen tidak dapat

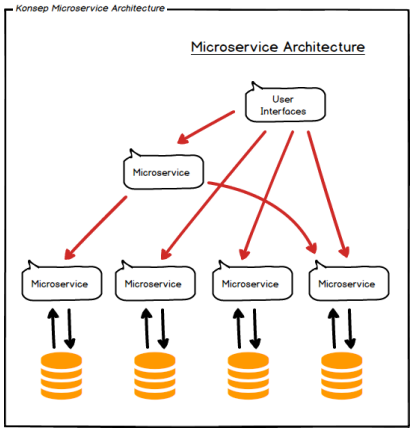
dieksekusi secara independen. secara umum masalah utama *monolithic* yaitu menyangkut skalabilitas semua aspek yang terkait dengan perubahan (Bucchiarone et al, 2018). Adapun konsep tersebut yaitu: *Authorization*, *Presentation*, *Business logic*, *Database layer*, *Application integration* dan *Notification module* (Siraj, 2018) seperti yang disajikan pada Gambar 1.



Gambar 1. *Monolithic* arsitektur.

B. Konsep *microservices* arsitektur

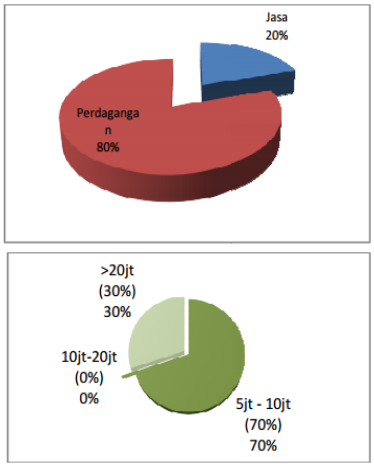
Menurut (Hasselbring and Steinacker, 2017) Secara tradisional, integrasi sistem informasi bertujuan untuk mencapai koherensi data yang tinggi di antara sumber informasi heterogen. Salah satu tujuan arsitektur layanan mikro adalah untuk mengatasi keterbatasan kemampuan *monolithic* arsitektur seperti yang disajikan pada Gambar 2.



Gambar 2. Konsep *microservices* arsitektur.

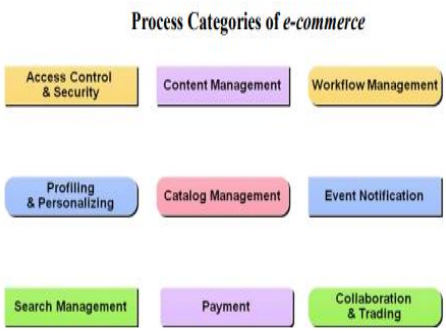
C. Penelitian Terkait

Penelitian yang dilakukan oleh (Maryama 2018), tentang "Penerapan *E-Commerce* Dalam Meningkatkan Daya Saing Usaha" penelitian ini melibatkan 10 responden sebagai sampel. Kesepuluh responden tersebut adalah para pelaku usaha baik dalam bidang perdagangan (barang/produk) maupun yang bergerak di bidang jasa (*service*) dengan rincian sebagai berikut: 80% responden adalah para pelaku usaha di bidang perdagangan (barang) dan sisanya sebanyak 20% adalah para pelaku usaha yang bergerak di bidang jasa seperti yang disajikan pada Gambar 3.



Gambar 3. Jenis usaha responden dan omset usaha

Penelitian yang dilakukan oleh (Irmawati, 2011), tentang "Pemanfaatan *E-Commerce* dalam Dunia Bisnis". Manfaat penggunaan *e-commerce* dalam banyak kasus, sebuah perusahaan *e-commerce* bisa bertahan tidak hanya mengandalkan kekuatan produk saja tapi dengan adanya tim manajemen yang handal, pengirman yang tepat waktu, pelayanan yang bagus, struktur organisasi bisnis yang baik, jaringan infrastruktur dan keamanan, desain situs web yang bagus. Dalam proses aplikasi *e-commerce*, perusahaan membutuhkan beberapa komponen utama agar operasi dan manajemen aktivitas *e-commerce* berjalan dengan baik. Komponen-komponen pokok yang memiliki peran penting dalam proses *e-commerce* dunia usaha tampak seperti pada Gambar 4.

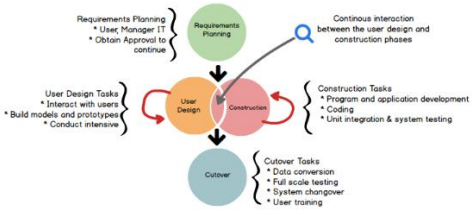


Gambar 4. Proses *e-commerce*

METODOLOGI PENELITIAN

Pengembangan aplikasi *taking order* merupakan bagian terpenting dalam sebuah bisnis di bidang *e-commerce*, oleh karena itu dalam perancangan sebuah sistem seorang *programmer* harus berfikir kritis dalam membangun sebuah sistem. Seperti yang telah di jelaskan pada penjelasan di atas, aplikasi *taking order* pada PT XYZ masih menerapkan pendekatan *monolithic* arsitektur. Dimana, metode pengembangan aplikasi dengan pendekatan *monolithic* arsiktetur tersebut masih menggunakan satu sumber data (*single database)*, sehingga proses integrasi data menjadi tidak efektif. Melihat dari permasalahan tersebut, tujuan yang kami ajukan yaitu membangun aplikasi *taking order* dengan pendekatan *microservice* arsitektur

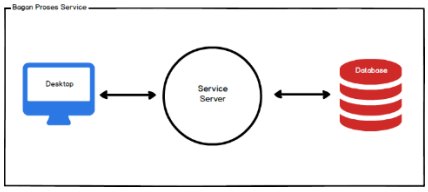
dengan metode pengembangan aplikasi *Rapid Aplication Development* (RAD). Metode RAD memiliki 4 tahapan yaitu tahapan perencanaan, tahap desain, pembuatan aplikasi, pengujian aplikasi. Pada tahapan desain dan pengembangan, peneliti akan mendesain, mengimplementasikan pengembangan aplikasi yaitu dengan metode pengembangan *Rapid Application Development* (RAD). Desain pengembangan sistem menurut (Tilley, 2016) seperti yang diasajikan pada Gambar 5.



Gambar 5. *Rapid Application Development* (RAD)

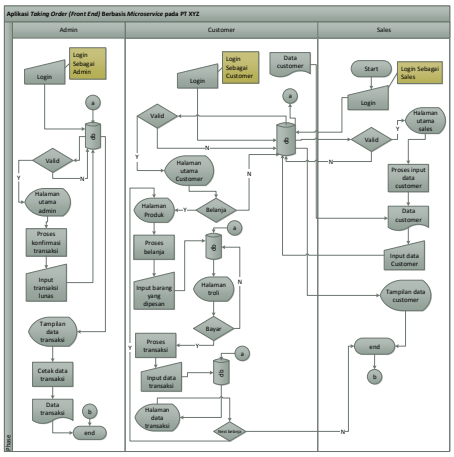
HASIL DAN PEMBAHASAN

Monolithic arsitektur merupakan cara standar untuk membangun sebuah aplikasi. Selama bertahun-tahun menggunakan basis kode tunggal yang membuat pengembang lebih mudah dan aplikasi relatif kecil. Sistem yang sedang berjalan menerapkan sistem layanan *monolithic* arsitektur dibawah ini dapat diliat gambar sistem yang sedang berjalan di PT XYZ disajikan pada Gambar 6.



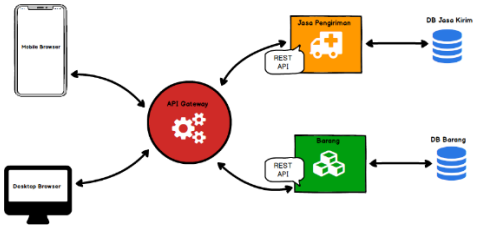
Gambar 6. Sistem yang sedang berjalan

Setiap *user* melakukan proses transaksi di bagian *front end* (1) *user* terhubung langsung dengan *server* induk dan keadaan *server* induk terhubung langsung dengan *database.* Semua proses yang dilakukan di dalam satu *server* maka ketika dibagian *database* sedang dilakukan perbaikan, semua akses dari *server* induk tidak dapat dijalankan. Mekanisme kerja sistem yang sedang berjalan saat ini dapat dilihat pada Gambar 7.



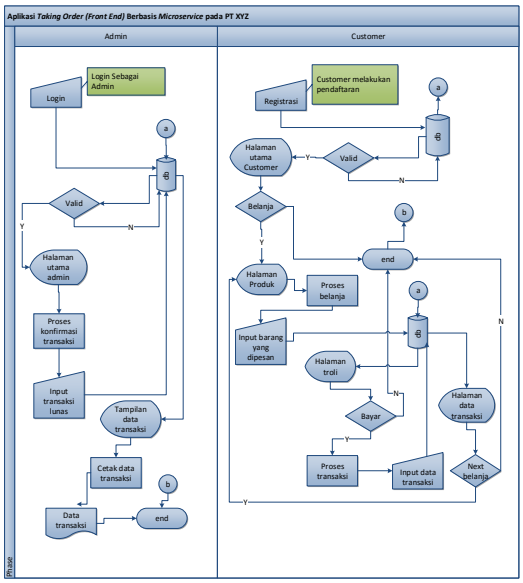
Gambar 7. *Mapping chart* sistem yang berjalan.

Berdasarkan mendefinisikan tujuan untuk solusi permasalahan pada sistem yang masih berjalan, diketahui bahwa sistem yang berjalan masih belum optimal, maka diperlukan suatu sistem baru yang dapat memecahkan masalah dari sistem yang masih berjalan, salah satu sistem baru yang menerapkan sistem layanan *microservices* arsitektur dapat dilihat pada Gambar 8.



Gambar 8. Sistem baru (yang diajukan)

Mapping chart sistem yang diusulkan pada aplikasi *taking order* ini merupakan hasil dari pengembangan dari sistem yang masih berjalan di dalam sistem yang diusulkan ini menerapkan sistem pengembang menggunakan layanan *microservices* arsitektur yang bertujuan untuk memudahkan melakukan kerja sama dengan pengembang lain agar dapat menjalankan akses *REST* API yang disediakan oleh pengembang lain. Sistem baru yang diusulkan mempunyai mekanisme kerja sistem dapat dilihat pada Gambar 9.



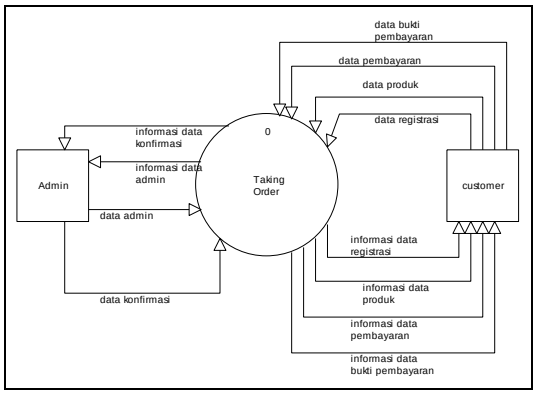
Gambar 9. *Mapping chart* sistem baru yang diusulkan.

Aliran data dalam suatu *entitas* ke sistem atau sistem ke *entitas* digambarkan dengan *Data Flow Diagram* (DFD). DFD dapat diartikan sebagai teknis grafis yang menggambarkan aliran data dari sistem tersebut, sedangkan transformasi digunakan sebagai data dari *input* atau *output*. DFD yang dibuat dalam sistem baru ini terdiri dari *level* 0 dan *level* 1.

1) DFD *level* 0

DFD *level* 0 salah satu *level* yang tertinggi yang menggambarkan hubungan sistem dengan lingkungan luarnya atau entitas luar yang terlibat dalam sistem, aliran data masuk,

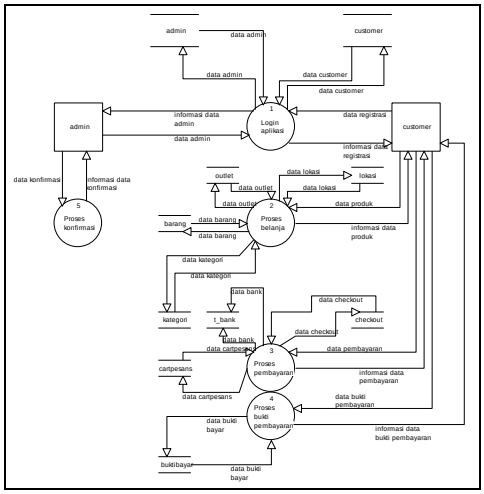
aliran data keluar. DFD *level* 0 disajikan pada Gambar 10.



Gambar 10. DFD *level* 0

2) DFD *level* 1

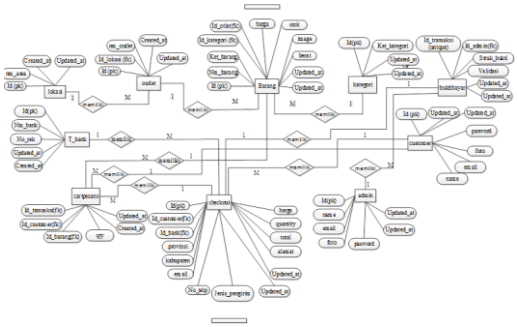
DFD *level* 1 merupakan gambaran dari proses yang terdapat pada DFD *level* 0, *level* ini menunjukkan komponen dalam proses yang terjadi pada sistem. DFD *level* 1 dapat dilihat pada Gambar 11.



Gambar 11. DFD *level* 1

A. *Entity Relational Diagram* (ERD)

ERD merupakan rancangan untuk menentukan *entity* serta *primary key*, kemudian menentukan relasi antara *entity*. ERD dari aplikasi *taking order* dari pengembangan aplikasi yang sedang berjalan disajikan pada Gambar 12.



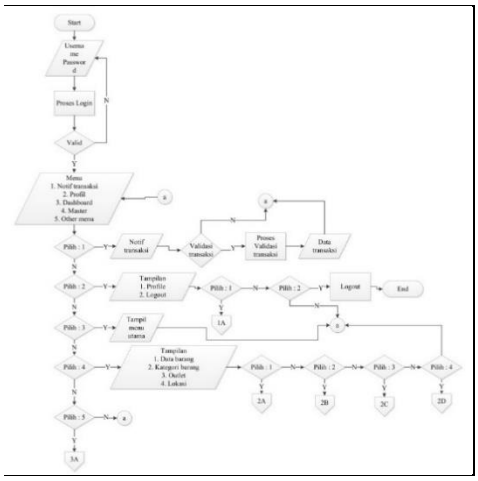
Gambar 12. *Entity Relational Diagram* (ERD)

B. Flowchart

Merupakan rancangan alur kerja pada sistem yang akan dibuat dengan menggunakan simbol-simbol standar yang disebut *flowchart*..

1) *Flowchart* halaman *admin*

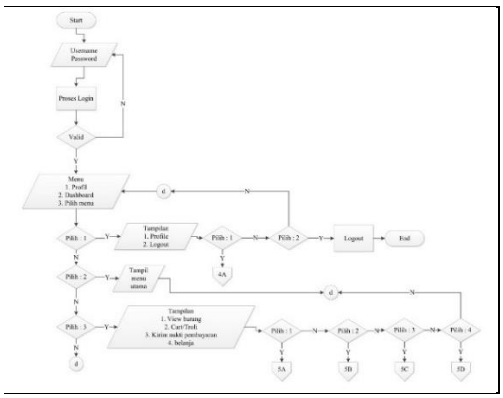
Aplikasi *taking order* dimulai dengan halaman awal atau *form login*, *form login* berisi kolom *username*, kolom *password*, dan tombol *login*. *Flowchart* halaman awal akan disajikan pada Gambar 13.



Gambar 13. *Flowchart* halaman admin.

2) *Flowchart* halaman *customer*

Aplikasi *taking order* dimulai dengan halaman awal atau *form login*, *form login* berisi kolom *username*, kolom *password*, dan tombol *login*. *Flowchart* halaman awal akan disajikan pada Gambar 14.

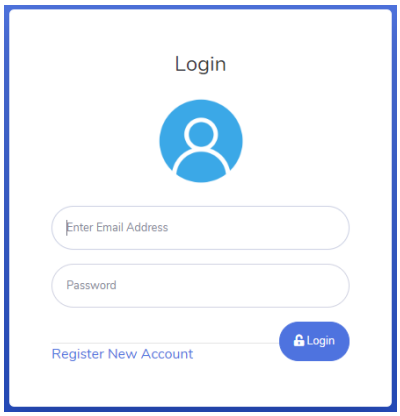


Gambar 14. *Flowchart* halaman *customer*

C. Tampilan Aplikasi

1) Tampilan halaman *login*

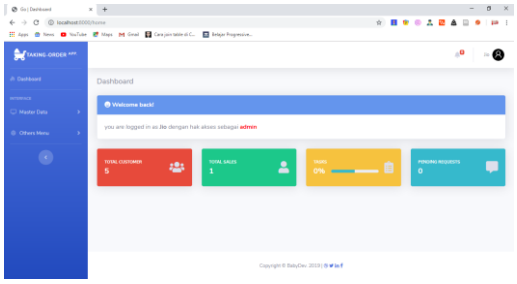
Halaman *login* berfungsi sebagai memberikan hak akses aplikasi, untuk dapat mengakses aplikasi. Tampilan halaman *login* aplikasi disajikan pada Gambar 15.



Gambar 15. Halaman *login*

2) Tampilan *dashboard admin*

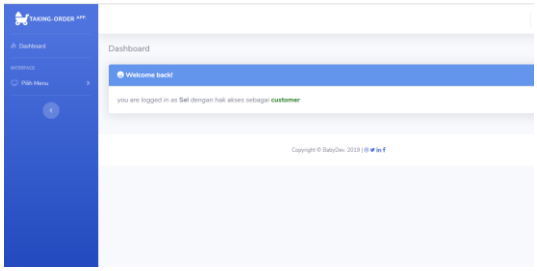
Halaman *dashboard* utama *admin* disajikan tampilan *menu* pada Gambar 16.



Gambar 16. *dashboard* admin

3) Tampilan *dashboard customer*

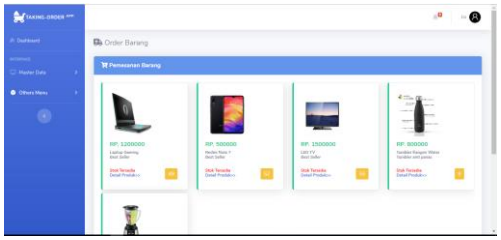
Halaman *dashboard customer* tampilan halaman *dashboard user/customer* disajikan pada Gambar 17.



Gambar 17. Tampilan halaman *dashboard customer*

4) Tampilan halaman *view* barang (*Order* Barang)

Halaman *view* barang menampilkan semua data barang yang akan dipesan oleh *customer*. Tampilan halaman *view* barang disajikan pada Gambar 18.



Gambar 18. Tampilan halaman *view* barang

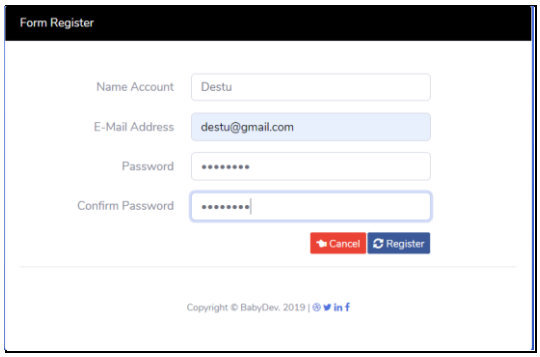
D. Hasil Demonstrasi

Hasil demontrasi ini merupakan hasil dari pengujian sistem yang sudah dibangun menggunakan *microservices* arsitektur, tahapan yang dilakukan uji coba adalah *create*, *read*, *update*, dan *delete* (CRUD). Pengujian CRUD akan dilakukan yaitu: kelola data *customer*, barang, kategori, lokasi, *outlet* dan *checkout*.

a. Pengujian *registrasi customer*

Tabel 1. Hasil menambah data *customer*.

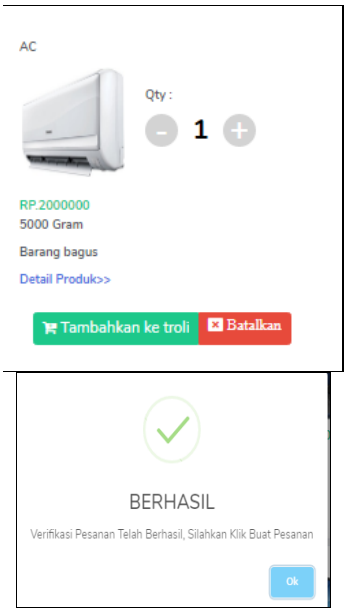
Skenario Uji	Hasil yang diharapkan	Hasil pengujian
Menambah data ke dalam form pendaftaran lalu tekan tombol daftar	Data akan tersimpan ke dalam tabel *customer*	✓



Gambar 19. *Form registrasi customer*

Tabel 2. Hasil tambah barang ke *troli*

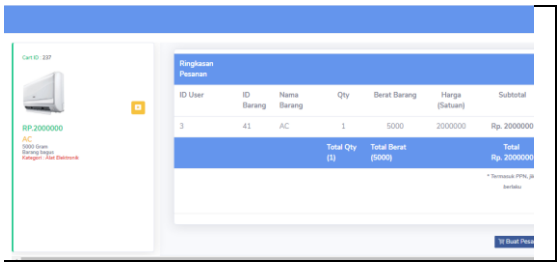
Skenario Uji	Hasil yang diharapkan	Hasil pengujian
Menambah data barang ke dalam *troli*	Data akan tersimpan ke dalam tabel *charpesan* dan pesan akan muncul	✓



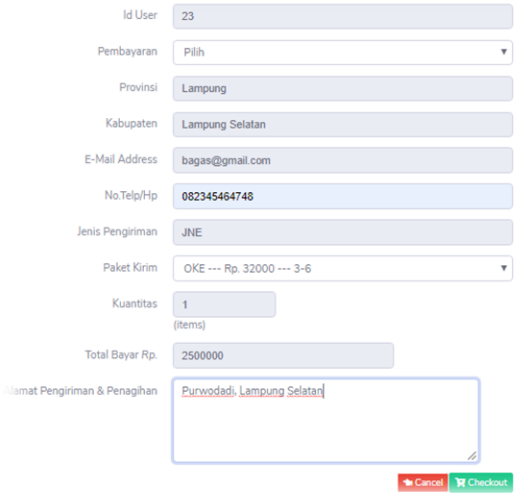
Gambar 20. Pemesanan barang dan *notifikasi* pesanan berhasil

Tabel 3. Hasil pembuatan pesanan

Skenario Uji	Hasil yang diharapkan	Hasil pengujian
Pembuatan pesanan barang	Data akan diarahkan ke halaman pengisian alamat pemesan	✓



Gambar 21. Pembuatan pesanan



Gambar 22. Proses *checkout*

Kesimpulan dan Saran

a. Kesimpulan

Pengembangan sistem aplikasi *taking order* sudah dilakukan dan menghasilkan aplikasi *taking order (front end)* menggunakan *microservice* pada PT XYZ untuk diimplementasikan di PT XYZ.

b. Saran

saran yang dapat diberikan untuk pengembangan aplikasi *e-commerce* di PT XYZ untuk selanjutnya aplikasi *e-commerce* PT XYZ dapat ditambahkan fitur *online chat* dan *tracking* pengiriman barang pada aplikasi tersebut supaya *customer* dapat menanyakan produk dan

pemantauan proses pengiriman barang yang dipesan.

REFERENSI

Azasoo, Julius Quarshie, and Kwame Osei Boateng. 2015. "A Retrofit Design Science Methodology for Smart Metering Design in Developing Countries." In *2015 15th International Conference on Computational Science and Its Applications*, Banff, AB, Canada: IEEE, 1–7. http://ieeexplore.ieee.org/document/7166156/ (July 27, 2019).

Bucchiarone, Antonio et al. 2018. "From Monolithic to Microservices: An Experience Report from the Banking Domain." *IEEE Software* 35(3): 50–55.

Hasselbring, Wilhelm, and Guido Steinacker. 2017. "Microservice Architectures for Scalability, Agility and Reliability in E-Commerce." In *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, Gothenburg, Sweden: IEEE, 243–46. http://ieeexplore.ieee.org/document/7958496/ (June 21, 2019).

Huang, Yadong, Yueting Chai, Yi Liu, and Jianping Shen. 2019. "Architecture of Next-Generation e-Commerce Platform." *Tsinghua Science and Technology* 24(1): 18–29.

Irmawati, Dewi. 2011. "PEMANFAATAN E-COMMERCE DALAM DUNIA BISNIS." : 18.

Maryama, Siti. 2018. "Penerapan E-Commerce Dalam Meningkatkan Daya Saing Usaha." *Liquidity* 2(1): 73–79.

Phil, Wittmer. 2019. "Microservices Disadvantages & Advantages – Tiempo Development." *tiempo development*. https://www.tiempodev.com/blog/disadvantages-of-a-microservices-architecture/ (July 19, 2019).

Siraj, Ul Haq. 2018. "Introduction to Monolithic Architecture and MicroServices Architecture." *Koder Labs*. https://medium.com/koderlabs/introduction-to-monolithic-architecture-and-microservices-architecture-b211a5955c63 (July 19, 2019).

Su, Jun-Ming, and Feng-Yuan Hsu. 2017. "Building a Visualized Learning Tool to Facilitate the Concept Learning of Object-Oriented Programming." In *2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*,

Tilley, Scott. 2016. *Systems Analysis and Design*. 11th edition. Boston, MA: Cengage Learning.

Karya Ilmiah.docx
4 jam yang lalu

13% Risiko dari plagiarisme
HIGH

Parafrase	1%
Kutipan salah	0%
Concentration	★★☆

Bagikan

Deep $ 1.00

Monetize

View report $ 3.01