Pembuatan Aplikasi MOSTRANS Transporter Berbasis Mobile Menggunakan React-Native JavaScript

Rubin Salim [1)] Desi Arisandi [2)] Janson Hendryli [3)]

[1)2)3)] Teknik Informatika Universitas Tarumanagara
Jl. Letjen S. Parman No. 1 Jakarta Barat 11440, Indonesia
Email: rubinsalim17@gmail.com[1)], desi@fti.untar.ac.id [2)], jansonh@fti.untar.ac.id[3)]

ABSTRACT

MOSTRANS Mobile Transporter, the mobile application in this paper aims to advance technology in the health supply chain ecosystem so that every part of the operation can be more efficient and modern. In dealing with this problem as well, MOSTRANS Mobile Transporter aims to meet the needs of MOSTRANS Software as a Service (SaaS) clients by assisting them in carrying out their daily operational activities by answering problems such as what features are needed. This application is built using JavaScript and React-Native for the front-end. With GraphQL as a complementary component for the back-end. To test this application, the Black Box Testing method is used as well as testing by the user with the User Acceptance Test to ensure the application can be used for everyday use. From the test results, it was concluded that the application created had fulfilled the purpose of assisting the operational activities of the MOSTRANS SaaS client by facilitating the features required by the MOSTRANS SaaS client. By making this application, the daily operational activities of the parties involved will be much more efficient and mobile..

Key words

React-Native, JavaScript, GraphQL, Android, Transporter

1. Pendahuluan

Pada Oktober 2019 PT Kalbe Farma Tbk (Kalbe) melalui salah satu entitas anak usahanya, PT Enseval Putera Megatrading Tbk (Enseval), meluncurkan platform transportasi digital yang berfokus pada produk-produk kesehatan. Platform transportasi digital yang diberi nama MOSTRANS ini menghubungkan ekosistem rantai pasok produk kesehatan (healthcare supply chain eco-system) antara perusahaan transportasi dengan pemilik barang. MOSTRANS akan mengintegrasikan layanan dari offline ke online antara perusahaan transportasi dan pemilik barang yang memiliki kebutuhan jasa pengiriman untuk produk kesehatan.

Seiring dengan maraknya pergerakan teknologi ke era digitalisasi, dibuatlah aplikasi berbasis *mobile* untuk klien MOSTRANS ini dengan beberapa tujuan. Tujuan pertama dan yang paling penting adalah untuk menambah jumlah pesanan ke *Transporter*. Di mana semakin banyak pesanan, semakin tinggi keuntungan perusahaan. Lalu yang ke dua adalah menambah perusahaan yang masuk ke dalam ekosistem *Shipper-Transporter* MOSTRANS tetapi dengan keuntungan perusahaan tersebut mengurus masalah logistik pesanan dan pengiriman dengan sendirinya. Lalu yang ke tiga, dengan diingatnya pandemi yang mengimbas seluruh dunia, banyak karyawan sebuah perusahaan yang tidak dapat datang ke kantor perusahaannya. Oleh karena itu, dibutuhkanlah alternatif dari web *application* MOSTRANS yang berbentuk website, yaitu aplikasi berbasis *mobile*. Dengan aplikasi yang berbentuk *mobile*, kegiatan operasional sebuah perusahaan pun tetap dapat berjalan tanpa ketergantungan kepada perangkat keras seperti komputer atau laptop yang kurang fleksibel jika dibandingkan sebuah ponsel.

2. Dasar Teori

Berikut teori yang digunakan dalam aplikasi MOSTRANS *Transporter*:

2.1. MOSTRANS

MOSTRANS menghubungkan ekosistem rantai pasok produk kesehatan (healthcare supply chain eco-system) antara perusahaan transportasi (*transporter)* dengan pemilik barang(*shipper)*. MOSTRANS juga memberikan beberapa solusi bagi perusahaan transportasi dan pemilik barang, antara lain Software as a Service (SaaS), Marketplace dan Supply Chain Financing. Dengan Software as a Service (SaaS) yang juga merupakan tujuan utama dari pembuatan aplikasi ini, MOSTRANS membantu mitranya dalam bertransformasi digital agar tetap relevan dalam menghadapi tantangan di era industri 4.0.

2.2. Shipper

MOSTRANS merupakan platform pengiriman kargo obat-obatan dan alat Kesehatan lainnya. Jika diterjemahkan dari bahasa Inggris, *Shipper* bisa dibilang adalah “pengirim”. Lalu dalam konteks aplikasi ini seorang *Shipper* merupakan orang yang meminta melakukan pengiriman melalui jasa ekspedisi atau jasa pengiriman[1].

2.3. Transporter

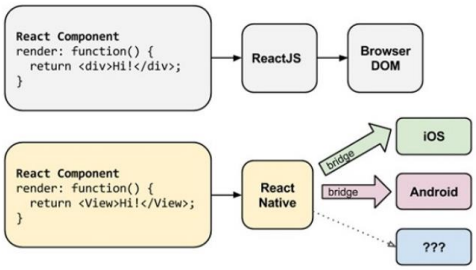
Transporter atau yang biasa dikenal sebagai perusahaan truk Indonesia menanggung tugas sebagai jasa logistik pengiriman barang dari hasil produksi pada sebuah perusahaan bisnis hingga perusahaan manufaktur di Indonesia[2].

2.4. Android

Android merupakan sistem operasi *mobile* yang paling popular dengan *market share* sekitar tujuh puluh tiga persen dari seluruh *market share* sistem operasi *mobile*.

2.5. React-Native

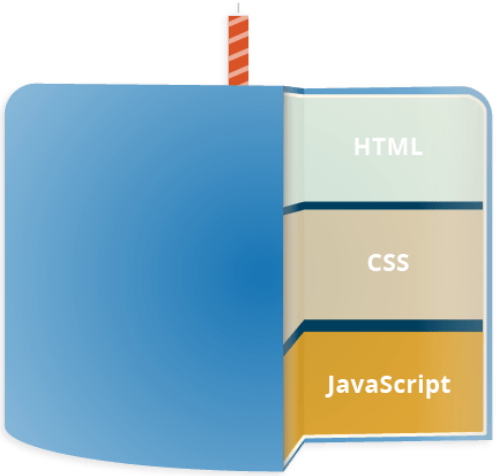
React Native adalah *framework JavaScript* untuk merancang aplikasi *mobile* untuk iOS dan Android. Basisnya adalah *React*, *library JavaScript* buatan Facebook untuk mengembangkan *user interface*, tetapi jika *React* menargetkan browser, *React-Native* menargetkan platform *mobile*. Dengan kata lain: developer web sekarang dapat mengembangkan aplikasi *mobile* yang terlihat dan terasa asli, semuanya dibantu dengan *library JavaScript* yang sudah luas dan umum digunakan.



Gambar 1 Rendering pada React dan React-Native

2.6. JavaScript

JavaScript adalah bahasa pemrograman yang memungkinkan kita untuk mengimplementasikan fitur-fitur kompleks pada sebuah *website*. *JavaScript* adalah lapis ke tiga dari lapisan standar teknologi web. Dua lainnya adalah HTML dan CSS. Gambar lapisan standar teknologi web ada di **Gambar 2** di bawah ini.



Gambar 2 Lapisan Standar Teknologi Web

2.7. Application Programming Interface (API)

Application Programming Interface atau biasa disebut API adalah penengah *software* yang memungkinkan dua aplikasi untuk berinteraksi satu sama lain. Setiap kali kita menggunakan aplikasi seperti WhatsApp untuk mengirim pesan, atau mengecek cuaca, kita menggunakan API.

2.8. Graph-QL

Graph-QL merupakan bahasa *server-side query* dan *runtime* untuk API yang memprioritaskan memberikan klien data persis seperti yang diminta[3]. Secara esensial, *Graph-QL* merupakan bahasa untuk *querying database* dari aplikasi sisi klien[4]. Di bagian - *back-end*, *Graph-QL* dapat mengspesifikasi kepada API bagaimana datanya akan dipresentasikan ke klien. *Graph-QL* juga didesain untuk membuat API lebih cepat, lebih fleksibel dan *developer friendly*[5].

2.9. Black Box Testing

Black Box Testing memperlakukan software sebagai sebuah "Kotak Hitam" – tanpa pengetahuan mengenai cara kerja internal dan hanya memeriksa aspek fundamental dari sistem. Selama melakukan Black Box Test, penguji harus mengetahui arsitektur sistem dan tidak akan mempunyai akses ke dalam source code[6].

Beberapa kelebihan dan kekurangan dari Black Box Testing adalah sebagai berikut[7]:

Kelebihan

- Efisien untuk segmen code besar.
- Persepsi penguji sangat sederhana.
- Perspektif pengguna sangat terpisah dari perspektif developer. (Programmer dan tester independent dari satu sama lain).
- Pengembangan test case lebih cepat.

Kekurangan

- Hanya beberapa skenario pengujian yang benar-benar dilakukan. Akibatnya, cakupannya menjadi terbatas.
- Tanpa spesifikasi yang jelas, test case sulit untuk didesain.
- Pengujian yang kurang efisien.

Beberapa nama lain dari Black Box Testing adalah Opaque Testing, Functional Testing, Close Box Testing, dan Behavioural Testing.

3. Hasil Percobaan

Berikut merupakan hasil percobaan aplikasi MOSTRANS Mobile Transporter menggunakan metode *Black Box Testing:*

Tabel 1 Hasil Pengujian Black Box Testing

No.	Input	Output	Status
1.	Mengisi form email dan password untuk login lalu klik Masuk	Masuk ke Dashboard	Valid
2.	Klik salah satu status trip pada Dashboard	Modal List Trip muncul	Valid
3.	Klik pada salah satu trip dari modal list trip dashboard	Detail trip muncul	Valid
4.	Klik icon sidebar pada navbar	Menu Sidebar beserta list menu lainnya muncul	Valid
5.	Klik menu Pengiriman SaaS	Masuk ke menu Pengiriman SaaS	Valid
6.	Klik salah satu trip di listing	Muncul detail pengiriman yang di klik	Valid
7.	Klik salah satu daftar order di detail pengiriman	Muncul detail order	Valid
8.	Klik kotak Nama Driver	Muncul modal assign Driver	Valid
9.	Klik salah satu nama Driver lalu klik Assign	Modal Tertutup, nama Driver terganti	Valid
10.	Klik kotak Plat Nomor	Muncul modal assign Driver	Valid
11.	Klik salah satu plat nomor lalu klik Assign	Modal tertutup, plat nomor terganti	Valid
12.	Klik Confirm	Muncul Alert dari sistem: ‘Success Confirmed Trip, Thankyou!’	Valid
12.	Klik Reject	Muncul Modal berisi daftar alasan reject	Valid
13.	Pilih salah satu alasan lalu klik reject	Muncul Alert dari sistem: ‘Succes Trip Berhasil direject!’	Valid
14.	Klik Partner SaaS pada *Sidebar*	Masuk ke menu Partner SaaS	Valid
15.	Klik icon telpon pada salah satu partner	Muncul menu telpon serta nomor kontak partner tersebut	Valid
16.	Klik icon GPS pada salah satu partner	Muncul alamat partner pada Google Maps	Valid
17.	Klik icon switch pada salah satu partner	Muncul Alert konfirmasi ‘Apakah anda yakin ingin mengaktivasi/meng non-aktifkan partner ini?’	Valid
18.	Klik Ya pada alert konfirmasi partner	Muncul Alert ‘Aktivasi/Deaktivasi berhasil!'	Valid
19.	Klik *icon add* partner	Muncul menu insert Partner	Valid
20.	Masukkan *token* partner lalu klik tombol Insert Partner	Muncul Alert: ‘Success Berhasil Insert Partner!	Valid
21.	Klik icon bell pada Navbar	Menu Notifikasi terbuka	Valid
22.	Klik icon kunci pada Navbar	Modal Generate Token terbuka	Valid
23.	Klik tombol Generate pada modal *Generate Token*	Muncul *token* yang berhasil *generate*	Valid
24.	Klik menu Laporan SaaS pada *Sidebar*	Menu Laporan SaaS terbuka	Valid
25.	Klik tombol tanggal pada menu Laporan	Modal tanggal (Kalender) terbuka	Valid
26.	Klik jarak tanggal yang diinginkan	Modal tertutup, tanggal dan data yang ditampilkan berubah	Valid
27.	Klik menu Daftar Pengguna pada *Sidebar*	Menu Daftar Pengguna terbuka	Valid
28.	Klik *icon* Add pengguna	Muncul menu tambahkan pengguna	Valid
29.	Masukkan nama pengguna, nomor *handphone*, akses,	Muncul Alert dari sistem: ‘Sukses tambah pengguna!’ lalu Kembali ke daftar pengguna	Valid

	email, status, dan password lalu klik tombol Tambah Pengguna		
30.	Klik *icon* pensil pada salah satu data pengguna	Muncul menu *edit* pengguna	Valid
31.	Ubah email lalu klik tombol Edit Pengguna	Muncul Alert dari sistem: 'Sukses edit pengguna!' lalu kembali ke daftar pengguna	Valid
32.	Klik menu Pengemudi pada *Sidebar*	Menu pengemudi terbuka	Valid
33.	Klik icon *Add* Pengemudi	Menu Tambah Pengemudi terbuka	Valid
34.	Masukkan nama pengemudi, nomor *handphone*, email, nomor KTP, nomor SIM, password, foto pengemudi, foto KTP, dan foto SIM, lalu klik tombol Tambah Pengemudi	Muncul Alert dari sistem: 'Sukses pambah pengemudi!' lalu kembali ke daftar pengemudi	Valid
35.	Klik *icon* pensil pada salah satu data pengemudi	Menu Edit Pengemudi terbuka	Valid
36.	Ubah nomor KTP dan nomor SIM lalu klik tombol Edit Pengemudi	Muncul Alert dari sistem: 'Sukses edit pengemudi!' lalu Kembali ke daftar pengemudi	Valid
37.	Klik menu Pool pada *Sidebar*	Menu Pool terbuka	Valid
38.	Klik *icon* Add *Pool*	Menu Tambah *Pool* terbuka	Valid
39.	Masukkan nama *Pool*, alamat, foto *pool*, kota, detail alamat, dan kode pos lalu klik tombol Tambah Pool	Muncul Alert dari sistem: 'Sukses tambah pool!' lalu Kembali ke daftar *Pool*	Valid
40.	Klik *icon* pensil pada satu salah satu data *Pool*	Menu edit *Pool* akan terbuka	Valid
41.	Ubah nomor kontak, alamat dan kota lalu klik tombol Edit Pool	Muncul Alert dari sistem: 'Sukses Edit Pool!' lalu kembali ke daftar *Pool*	Valid
42.	Klik menu Kendaraan pada *Sidebar*	Menu Daftar kendaraan terbuka	Valid
43.	Klik *icon* Add *Kendaraan*	Menu Tambah *Kendaraan* terbuka	Valid
44.	Masukkan tipe kendaraan, nomor KIR, plat nomor, pool, jatuh tempo KIR, dan tahun pembuatan lalu klik tombol Tambah Kendaraan	Muncul Alert dari sistem: 'Sukses tambah kendaraan!' lalu Kembali ke daftar kendaraan	Valid
45.	Klik *icon* pensil pada satu salah satu data kendaraan	Menu edit kendaraan akan terbuka	Valid

4. Kesimpulan

Berdasarkan hasil pembahasan pada bab sebelumnya terdapat beberapa poin ringkasan, yakni:

1. *Framework* React-Native serta bahasa pemrograman JavaScript telah berhasil diterapkan dalam pembuatan antarmuka aplikasi MOSTRANS *Mobile Transporter*.
2. Bahasa pemrograman GraphQL berhasil diimplementasikan untuk mengambil data dari server *database* dan untuk menampilkan data tersebut ke dalam aplikasi MOSTRANS *Mobile Transporter*.
3. Pembuatan aplikasi MOSTRANS *Mobile Transporter* yang akan membantu para *user* menjalankan kegiatan operasional perusahaannya berhasil dilakukan.
4. Berdasarkan hasil pengujian *Black Box Testing* dapat disimpulkan bahwa fungsionalitas aplikasi MOSTRANS *Mobile Transporter* sudah sesuai kebutuhan dan dapat berjalan dengan baik.

REFERENSI

[1] Kargo Tech. 2021. Siapa itu Shipper? Apa yang Bisa Mereka Lakukan di Kargo. Juni 23. Accessed September 9, 2021. https://kargo.tech/en/blog/siapa-itu-shipper/.

[2] Kargo Tech. 2021. Apa Itu Perusahaan Truk Transporter? April 8. Accessed September 9, 2021. https://kargo.tech/en/blog/apa-itu-perusahaan-transporter/.

[3] F. Mondaca, Philip Schildkamp, F. Rau. 2019. "Introducing Kosh, a Framework for Creating and Maintaining APIs for Lexical Data." Proceedings of Electronic Lexicography in the 21st Century Conference, 2019-October 907-921.
[4] Brito, Gleison, Thais Mombach, and Marco Tulio Valente. 2019. "Migrating to Graph-QL: A Practical Assessment." 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER) 140-150.
[5] K.S. Malakhov, A.P. Kurgaev, V.Yu. Velychko. 2018. "Modern restful api dls and frameworks for restful web services api schema modeling, documenting, visualizing." Problems in programming 59-68.
[6] Khan, Mohd. Ehmer. 2011. "Different Approaches to Black Box Testing." JSEA 31-40.
[7] Khan, Mohd. Ehmer. 2011. "Different Approaches to White Box Testing." IJSEIA 1-13.

Rubin Salim, saat ini sebagai mahasiswa program studi Teknik Informatika Universitas Tarumanagara.