SKRIPSI

PENINGKATAN EFEKTIVITAS PENGELOLAAN STATE MENGGUNAKAN POLA PROP DRILLING VUEJS PADA FITUR FINANSIAL JALA TECH

Disusun Oleh:

Nama : Albarra Naufala Erdanto

NIM : 18523158

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2021

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENINGKATAN EFEKTIVITAS PENGELOLAAN STATE MENGGUNAKAN POLA PROP DRILLING VUEJS PADA FITUR FINANSIAL JALA TECH

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

Nama : Albarra Naufala Erdanto

NIM : 18523158

Yogyakarta, 27 Desember 2021

Pembimbing,

Ari Sujarwo, S.Kom, M.I.T.

HALAMAN PENGESAHAN DOSEN PENGUJI

PENINGKATAN EFEKTIVITAS PENGELOLAAN STATE MENGGUNAKAN POLA PROP DRILLING VUEJS PADA FITUR FINANSIAL JALA TECH

TUGAS AKHIR JALUR MAGANG

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 27 Desember 2021

Tim Penguji

Ketua Penguji

Ari Sujarwo, S.Kom., M.I.T. _____

Anggota 1

Dr. Ahmad Luthfi, S.Kom., M.Kom. _____

Anggota 2

Andhik Budi Cahyono, S.T., M.T. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Albarra Naufala Erdanto

NIM : 18523158

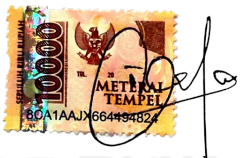
Tugas akhir dengan judul:

PENINGKATAN EFEKTIVITAS PENGELOLAAN STATE MENGGUNAKAN POLA PROP DRILLING VUEJS PADA FITUR FINANSIAL JALA TECH

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apa pun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 27 Desember 2021



Albarra Naufala Erdanto

HALAMAN PERSEMBAHAN

Laporan akhir ini merupakan persembahan istimewa untuk siapa pun yang telah mendukung saya dalam bentuk materi dan doa. Persembahan terbesar kepada orang tua, sahabat, dan teman saya karena telah memberikan dukungan sebesar-besarnya hingga akhirnya saya bisa tumbuh sampai sejauh ini dan menyelesaikan tanggung jawab ini. Terima kasih telah mengajarkan saya untuk berpikir kritis, berperilaku bijak, dan selalu bersyukur. Mohon maaf atas kekurangan dan kesalahan yang saya buat hingga akhirnya menyakiti perasaan kalian semua.

HALAMAN MOTO

"With great power comes great responsibility."

- Paman Ben (Spider-Man)

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Alhamdulillah, puji dan syukur kepada Allah SWT atas rahmat dan hidayah-Nya sehingga pemagang dapat menyelesaikan laporan akhir yang berjudul "Peningkatan Efektivitas Pengelolaan *State* Menggunakan *Prop Drilling* VueJS pada Fitur Finansial Jala Tech". Laporan ini disusun sebagai bukti pelaksanaan magang serta untuk memenuhi kewajiban penulis dalam memperoleh gelar sarjana pada Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.

Pada pelaksanaan magang di Jala Tech, banyak hambatan dan tantangan yang dihadapi. Namun, pada akhirnya dapat terselesaikan dengan baik karena adanya dukungan dari orang-orang di sekitar saya. Oleh karena itu, pemagang menyampaikan terima kasih sebesar-besarnya kepada:

1. Kedua orang tua saya karena selalu memberikan dukungan berupa materi dan doa serta ilmu dasar hingga menjadikan saya sejauh ini.
2. Bapak Ari Sujarwo, S.Kom, M.I.T. selaku dosen pembimbing saya yang telah memberikan waktu, tenaga, dan ilmu untuk menuntun saya hingga akhirnya dapat menyelesaikan laporan akhir ini.
3. Mas Farid Inawan, Mas Syauqy Nurul Azis, Mbak Geppy Octavianti, Mas Jihad Maulana, Mbak Arina, dan karyawan lain di Jala Tech karena telah memberikan kesempatan, ilmu, dan bantuan selama melaksanakan magang.
4. Segenap dosen informatika yang telah memberikan ilmu dengan baik dan bermanfaat selama masa perkuliahan.
5. Hersa Ajeng Priska yang telah memberikan semangat, nasihat, dukungan, dan doa selama menjalani perkuliahan.
6. Ika Rahmanita, Farrel Alfaiz, Ulil Albab Surya Negara, Muhammad Farhan, Reza Ali, dan Muhammad Prasetyo selaku teman dekat saya karena telah memberikan cerita dan keseruan selama menjalani kuliah.
7. Teman-teman tim Kolaborative dan Deafcare Indonesia yang telah memberikan kesempatan kepada saya untuk berkontribusi dalam mengembangkan produk dan proyek yang bermanfaat kepada sesama.
8. Seluruh pihak yang telah membantu dalam menyelesaikan laporan akhir.

Atas bantuan dari berbagai pihak, pemagang dapat menyelesaikan laporan akhir dengan sebaik mungkin. Namun, penulis menyadari bahwa laporan akhir ini masih jauh dari sempurna sehingga

pemagang sangat membutuhkan kritik dan saran untuk membuat laporan ini menjadi lebih baik lagi. Akhir kata, semoga laporan ini bermanfaat bagi kita semua.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 27 Desember 2021

Albarra Naufala Erdanto

SARI

Jala Tech merupakan perusahaan teknologi yang membantu petambak udang dalam meningkatkan kualitas tambak dengan memanfaatkan aplikasi berbasis web. Satu di antara fitur yang terdapat pada aplikasi Jala adalah fitur finansial. Pengelolaan *state* pada kode *frontend* fitur tersebut menggunakan pola *event bus*. Namun, terdapat permasalahan pada penerapan *event bus* yaitu ketidakmampuan Vue Devtools dalam melakukan *debugging* pada *state*. Ketidakmampuan tersebut mengakibatkan sulitnya pengembang dalam melacak alur perubahan *state* pada tampilan aplikasi. Selain itu, penempatan *state* yang tidak terpusat menyebabkan terjadinya redundansi *state* pada komponen-komponen yang memanfaatkan *state* yang sama. Berdasarkan hal tersebut, perlu dilakukan penulisan ulang kode pada fitur finansial untuk mengubah pengelolaan *state* dari menggunakan pola *event* bus menjadi pola prop *drilling*. Hasil dari penelitian ini adalah peningkatan efektivitas pengelolaan *state* pada fitur finansial yang telah menerapkan *prop drilling* pada kode *frontend*. Dengan menggunakan *prop drilling*, dapat dilakukan *debugging* pada *state* menggunakan Vue Devtools. Selain itu, penempatan *state* menjadi terpusat pada komponen induk utama fitur finansial aplikasi web Jala. Dengan demikian, pengembang mampu melacak perubahan *state* serta lebih mudah mengenali *state* yang dipakai pada fitur finansial aplikasi web Jala.

Kata kunci: *Prop Drilling, State, Vue Devtools*.

GLOSARIUM

Backend	pengembangan aplikasi yang fokus pada server dan basis data.
Debug	langkah untuk menelusuri kesalahan kode program.
Design pattern	arsitektur yang digunakan untuk menunjukkan solusi dari masalah yang berulang.
Event	kejadian yang dapat dipicu dengan kode pemrograman dan dapat ditangani untuk menjalani eksekusi yang berkaitan dengan kejadian yang dipicu.
Framework	kerangka kerja yang menyediakan fungsionalitas generik dan dapat dilakukan perubahan oleh pengembang untuk membuat suatu perangkat lunak.
Frontend	pengembangan aplikasi yang fokus pada tampilan aplikasi.
HTML	bahasa markah standar untuk menyusun tampilan pada situs web.
Javascript	satu di antara bahasa pemrograman tingkat tinggi yang dapat digunakan untuk membuat berbagai macam jenis aplikasi seperti web, *desktop*, dan *mobile*.
Library	modul yang telah dibuat oleh pemrogram dan dapat digunakan oleh pemrogram lain.
Props	argumen yang dapat dikirim kepada komponen anak melalui atribut komponen.
Reusable	dapat digunakan kembali.
Setup	tindakan membuat program siap untuk digunakan atau dikembangkan.
State	data atau variabel pada pengembangan *frontend* yang terikat pada tampilan dan dapat berubah seiring berjalannya waktu.
Tag	penanda awalan dan akhiran dari sebuah elemen di HTML yang ditulis dengan kurung siku (<...>) dan dapat memiliki konten maupun *tag* di dalamnya.
Vue Devtools	ekstensi peramban *chrome* yang digunakan pengembang *VueJS* untuk melakukan *debug* pada tampilan aplikasi web
VueJS	salah satu *framework* yang digunakan untuk mengembangkan frontend aplikasi web.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Di era yang serba digital saat ini, teknologi informasi dimanfaatkan oleh perusahaan untuk meningkatkan efektivitas berjalannya proses bisnis (Frestilia, 2013). Satu di antara perusahaan yang memanfaatkan teknologi informasi adalah Jala Tech. Jala Tech merupakan perusahaan yang membantu para petambak udang dalam meningkatkan produktivitas tambak dengan memanfaatkan teknologi. Satu di antara teknologi yang digunakan adalah berupa aplikasi berbasis web (Jala Tech, 2021).

Aplikasi web Jala dikembangkan menggunakan VueJS sebagai *framework* untuk sisi *frontend* aplikasi. VueJS merupakan sebuah *framework* progresif untuk membangun antarmuka aplikasi web (Vuejs, 2021). Dengan menggunakan VueJS, membangun tampilan aplikasi yang interaktif terhadap pengguna menjadi lebih mudah. VueJS juga menyediakan *tools* yang bernama Vue Devtools. Satu di antara keuntungan dalam menggunakan Vue Devtools adalah memudahkan pengembang melakukan debug dan memantau aktivitas perubahan *state* pada tampilan aplikasi (Cherckesova et al., 2021).

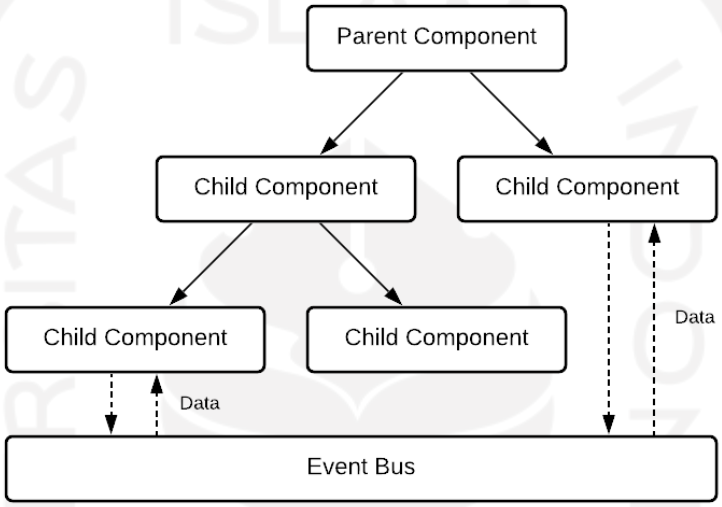
Pada pengembangan *frontend*, *state* adalah data atau variabel pada tampilan aplikasi yang dapat berubah seiring berjalannya waktu. *State* terikat pada komponen antarmuka aplikasi sehingga perubahan *state* menyebabkan perubahan pada antarmuka aplikasi. Seiring berkembangnya aplikasi, *state* yang digunakan menjadi lebih kompleks. Oleh karena itu, diperlukanlah pengelolaan *state* yang baik sesuai dengan kebutuhan aplikasi (Nelson, 2018).

Web Component atau yang disebut dengan komponen adalah komponen tampilan aplikasi yang bersifat *reusable* sehingga dapat digunakan di berbagai halaman yang berbeda (Yang et al., 2002). Komponen dapat terdiri dari beberapa komponen kecil penyusunnya. Komponen dapat memiliki *state*, fungsi, dan juga *props*. *Props* merupakan argumen yang dikirim dari komponen induk ke komponen anak melalui atribut komponen (Wohlgethan, 2018). Seperti *state*, *props* memengaruhi tampilan aplikasi, tetapi *props* tidak dapat diubah secara langsung oleh komponen itu sendiri.

Per tanggal 31 Oktober 2021, aplikasi web Jala terdapat sepuluh halaman utama (Jala Tech, 2021). Satu di antara halaman utama yang ada pada aplikasi web Jala adalah fitur finansial. Fitur

finansial merupakan fitur untuk mencatat pemasukan dan pengeluaran serta membuat laporan keuangan untuk para petambak udang.

Kode *frontend* fitur finansial aplikasi web Jala menggunakan pola *event bus* sebagai teknik pengelolaan *state*. *Event Bus* adalah sebuah medium yang memungkinkan beberapa *component* saling mengirim data (Freeman, 2018). Dengan *event bus*, *component* dapat mengirim data langsung melalui entitas *event bus* seperti pada Gambar 1.1.



Gambar 1.1 Ilustrasi pengiriman data menggunakan event bus

Berdasarkan dokumentasi VueJS (2022) dan hasil wawancara *Lead of Software Engineer* Jala Tech, Farid Inawan, penggunaan *event bus* pada fitur finansial memiliki beberapa kekurangan. Kekurangan pertama yaitu perubahan data atau *state* pada komponen tidak dapat dilacak menggunakan Vue Devtools. Ketidakmampuan tersebut mengakibatkan sulitnya investigasi aktivitas *state* yang tidak sesuai ekspektasi. Kekurangan kedua yaitu adanya *state* yang tidak saling sinkron. Hal tersebut dikarenakan e*vent bus* bukanlah sebuah medium untuk menyimpan *state*, melainkan medium untuk mengirim data. *State* yang berhasil dikirim oleh komponen melalui *event bus* harus disimpan sebagai *state* yang sama pada komponen penerimanya. Oleh karena itu, satu *state* yang memiliki fungsi yang sama dapat mengalami duplikasi *state* pada komponen lain. Hal tersebut dapat menjadi masalah jika *state* yang sama saling tidak sinkron atau memiliki nilai yang berbeda.

Berdasarkan permasalahan di atas, pelaksanaan magang bertujuan untuk meningkatkan efektivitas pengelolaan *state* dengan menulis ulang kode pengelolaan *state* dari menggunakan

event bus ke pola *prop drilling* pada fitur finansial aplikasi web Jala sehingga memberi kemampuan dalam men-*debug state* menggunakan Vue Devtools dan memusatkan *state* pada komponen induk fitur aplikasi web Jala sehingga pengembang dapat dengan mudah mengidentifikasi *state* yang digunakan untuk fitur tersebut.

1.2 Ruang Lingkup Magang

Magang dilaksanakan dalam durasi enam bulan dari Februari 2021 hingga Agustus 2021. Selama magang, dilakukan pengembangan pada aplikasi web Jala khususnya pada pengembangan *frontend*. Pengembangan dilakukan dengan menulis ulang kode *frontend* pada fitur finansial aplikasi web Jala dengan menerapkan *prop drilling* VueJS untuk pengelolaan *state*.

1.3 Tujuan

Tujuan dari pelaksanaan magang di Jala Tech adalah untuk meningkatkan efektivitas pengelolaan state dengan menggunakan pola *prop drilling* sehingga memberi kemampuan dalam men-*debug state* menggunakan Vue Devtools dan memusatkan *state* pada komponen induk fitur aplikasi web Jala sehingga pengembang dapat dengan mudah mengidentifikasi *state* yang digunakan untuk fitur finansial aplikasi web Jala.

1.4 Manfaat

Manfaat dari pengembangan kode *frontend* aplikasi web Jala dengan menerapkan *prop drilling* VueJS di fitur finansial adalah sebagai berikut

a. Mengurangi kemungkinan adanya dua atau lebih *state* yang tidak saling sinkron. Dikarenakan *event bus* menyimpan *state* secara desentralisasi pada banyak komponen, maka *state* yang seharusnya sama namun disimpan pada banyak komponen dapat terjadi redundansi pada *state* sehingga memungkinkan adanya *state* tidak saling sinkron.
b. Memudahkan pengembang dalam melakukan *debug* terhadap aktivitas *state* khususnya pada saat mengalami *error* atau perilaku *state* yang tidak sesuai ekspektasi.
c. Memudahkan pengembang dalam mengenali *state* yang digunakan pada fitur finansial karena telah disimpan secara terpusat di komponen utama fitur finansial aplikasi web Jala.

1.5 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam penyusunan laporan akhir ini adalah sebagai berikut:

a. BAB I – PENDAHULUAN

   Bab ini berisi mengenai latar belakang yang berisi masalah terkini dari aplikasi web Jala khususnya pada fitur finansial sehingga perlu dilakukannya peningkatan efektivitas pengelolaan *state* menggunakan *prop drilling*. Selain itu, bab ini juga berisi mengenai ruang lingkup magang, tujuan, manfaat, dan sistematika penulisan laporan.

b. BAB II – DASAR TEORI

   Bab ini membahas tentang teori-teori yang digunakan sebagai dasar penulisan laporan. Teori yang dibahas pada bab ini adalah membahas secara umum dari aplikasi web Jala, *web component*, *state*, *props*, dan *prop drilling*.

c. BAB III – PELAKSANAAN MAGANG

   Bab ini menguraikan pelaksanaan magang yang dilakukan di Jala Tech. Topik yang dibahas pada bab ini adalah pelaksanaan magang menggunakan metode *scrum* dalam pengembangan fitur finansial untuk meningkatkan efektivitas pengelolaan *state* menggunakan prop *drilling*.

d. BAB IV – REFLEKSI PELAKSANAAN MAGANG

   Bab ini berisi refleksi yang didapat pemagang dalam menjalani magang selama enam bulan di Jala Tech.

e. BAB V – KESIMPULAN DAN SARAN

   Bab ini berisi kesimpulan dari peningkatan efektivitas pengelolaan *state* menggunakan *prop drilling* VueJS pada fitur finansial aplikasi web Jala. Selain itu, adapun saran mengenai pengembangan di masa mendatang pada aplikasi web Jala.

f. DAFTAR PUSTAKA

   Daftar pustaka berisi mengenai sumber-sumber yang dijadikan sebagai referensi dari penulisan laporan.

g. LAMPIRAN

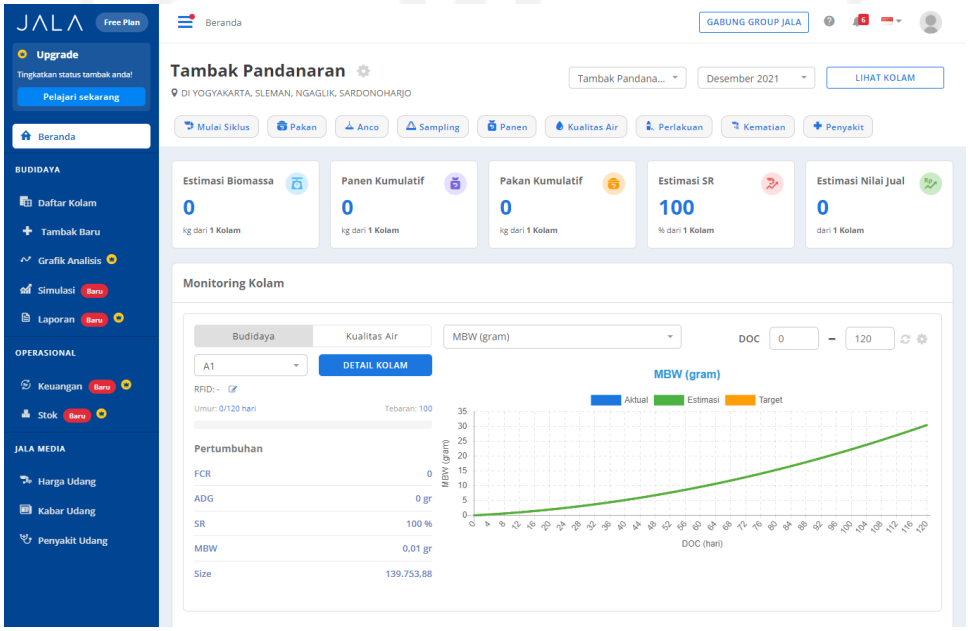
   Lampiran berisi gambar pendukung yang terkait dalam pelaksanaan magang di Jala Tech.

BAB II
DASAR TEORI

Dalam pelaksanaan magang, terdapat dasar teori yang mendasari adanya implementasi perubahan pengelolaan *state* dari menggunakan pola *event bus* menjadi pola *prop drilling*. Menurut Duldulao et al. (2021), terdapat elemen yang perlu dipahami dalam menerapkan pola *prop drilling*, yaitu *web component*, *state*, dan *props*. Dikarenakan objek dari pengembangan yang dilakukan adalah aplikasi web Jala, diperlukan juga pemahaman terkait aplikasi web Jala itu sendiri, seperti teknologi penyusunnya, dan kondisi sebelum dilaksanakannya magang.

2.1 Aplikasi Web Jala

Aplikasi Web Jala adalah perangkat lunak berbasis web yang membantu petambak udang dalam meningkatkan produktivitas tambak mereka (Jala Tech, 2021). Aplikasi web Jala terhubung dengan alat ukur kualitas air pintar sehingga petambak udang dapat dengan mudah memantau kualitas air pada tambak langsung dari gawai mereka. Oleh karena itu, penanganan terhadap tambak dapat lebih cepat dan terukur karena keputusan dibuat berdasarkan data yang ada. Bentuk dari aplikasi web Jala dapat dilihat pada Gambar 2.1.



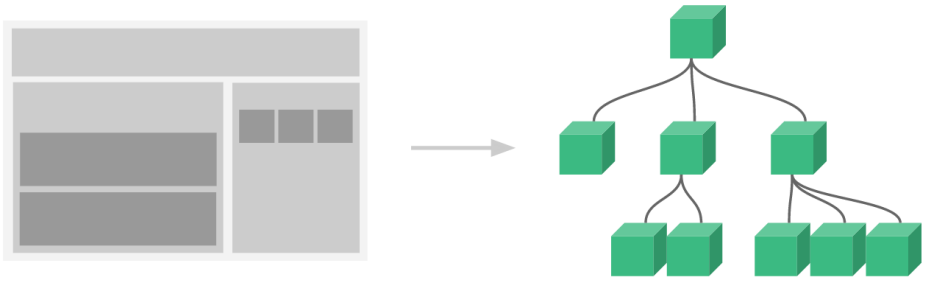
Gambar 2.1 Halaman dasbor utama aplikasi web Jala

Selain dapat memantau kualitas air pada tambak, aplikasi web Jala juga memiliki fitur lain seperti manajemen keuangan, manajemen stok, perdagangan udang, portal harga udang, portal kabar udang, dan buku digital penyakit udang. Pada laporan ini, fitur yang difokuskan adalah fitur manajemen keuangan atau fitur finansial.

Aplikasi web Jala dibangun menggunakan *framework* VueJS sebagai teknologi untuk mengembangkan *frontend* aplikasi (Jala Tech, 2021). Sebelum dilaksanakan magang, pengelolaan *state* pada aplikasi web Jala khususnya fitur finansial menggunakan pola *event bus*. Dengan *event bus*, pengiriman *event* yang dilakukan untuk memanipulasi *state* dikirim melalui medium yang disebut *event bus*.

2.2 *Web Component*

Web Component atau yang disebut dengan komponen adalah komponen tampilan aplikasi yang bersifat *reusable* sehingga dapat digunakan di berbagai halaman yang berbeda (Yang et al., 2002). Komponen dapat terdiri dari beberapa komponen kecil penyusunnya. Oleh karena itu, pengembang dapat menyusun tampilan aplikasi dengan cara menyusun atau mengorkestrasikan komponen kecil menjadi komponen yang lebih besar seperti pada Gambar 2.2.



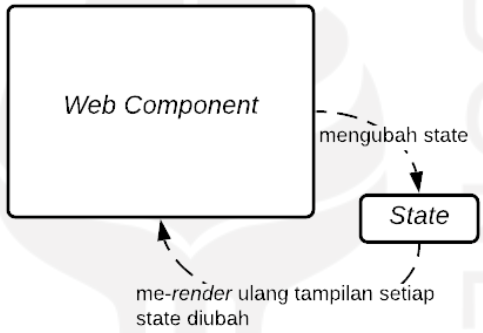
Gambar 2.2 Menyusun komponen menjadi antarmuka aplikasi
Sumber: VueJS (2021)

Sebuah komponen dapat menjadi komponen induk dan komponen anak tergantung sudut pandang dan keberadaan komponen itu sendiri. Sebuah komponen menjadi komponen induk jika memiliki komponen penyusun di dalamnya. Sedangkan komponen menjadi komponen anak jika komponen tersebut berada di dalam komponen induk.

Komponen dapat memiliki *state*, fungsi, dan juga *props*. Dengan *state*, komponen dapat menampilkan data yang dinamis serta dapat mengubah bentuk sesuai keadaan yang ada. Adapun fungsi yang memungkinkan komponen dapat melakukan aksi sesuai dengan perannya. *Props* juga berguna untuk memungkinkan komponen menampilkan data serta bentuk yang berbeda walaupun dipakai berulang-ulang.

2.3 *State*

State adalah data, variabel, atau keadaan pada tampilan aplikasi yang dapat berubah seiring berjalannya waktu (Nelson, 2018). *State* terikat pada komponen antarmuka aplikasi sehingga perubahan *state* menyebabkan perubahan pada antarmuka aplikasi. Hal tersebut dikarenakan *state* me-*render* ulang komponen setiap perubahan *state* seperti pada Gambar 2.3.



Gambar 2.3 Perubahan state pada komponen

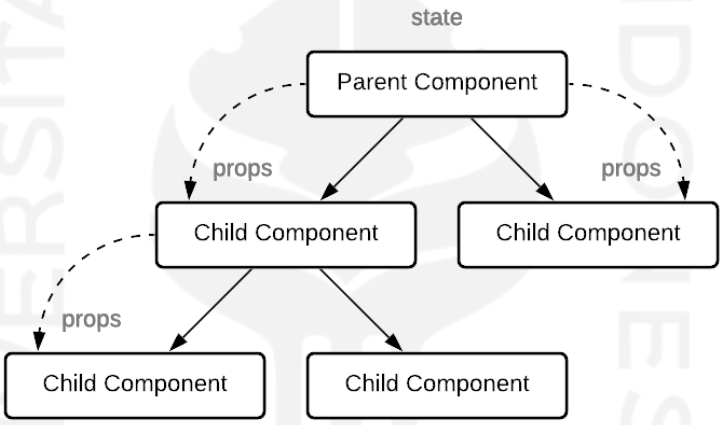
Seiring berkembangnya aplikasi, keberadaan *state* akan semakin kompleks. Komponen akan memiliki keadaan yang lebih beragam dan membuat halaman aplikasi memiliki tingkat keinteraktifan yang tinggi. Oleh karena itu, pemilihan cara pengelolaan *state* yang tepat diperlukan pada pengembangan *frontend* aplikasi (Nelson, 2018).

2.4 *Props*

Props adalah argumen yang dikirim dari komponen induk ke komponen anak melalui atribut komponen (Wohlgethan, 2018). Seperti state, *props* dapat memengaruhi komponen namun *props* tidak dapat diubah secara langsung oleh komponen itu sendiri. Dengan adanya *props*, komponen yang menampilkan data dinamis dapat dipakai berulang kali. Sama seperti fungsi yang memiliki parameter atau argumen agar melakukan kalkulasi sesuai argumen yang dikirim, komponen juga dapat menampilkan bentuk yang berbeda-beda tergantung *props* yang dikirim.

2.5 *Prop Drilling*

Prop drilling adalah pola pengelolaan *state* yang sangat mendasar pada pengembangan *frontend* aplikasi. *Prop drilling* mengirim *state* dengan memanfaatkan *props* dari komponen terluar secara terurut. *State* disimpan pada komponen induk utama sehingga penempatan *state* terpusat di satu komponen. Jika anak komponen atau bahkan komponen yang lebih dalam membutuhkan data dari *state*, maka komponen induk yang menyimpan *state* harus mengirim *state* tersebut melalui komponen anaknya secara berurutan (Kankaala, 2019). Alur pengiriman *state* dapat dilihat pada Gambar 2.4.



Gambar 2.4 Alur pengiriman *state* menggunakan *prop drilling*

Props yang dikirim dari komponen induk tidak dapat langsung diubah oleh komponen yang menerima *props*. Oleh karena itu, jika komponen anak perlu membuat perubahan pada *props*, maka komponen anak perlu mengirim *event* perubahan tersebut ke komponen yang lebih luar secara berurutan sampai komponen yang menyimpan *state* tersebut. Di VueJS, *event* dikirim menggunakan metode global *emit* (Vuejs, 2021).

Pemilihan pengelolaan *state* merupakan hal yang krusial pada pengembangan *frontend* aplikasi (Szymanek, 2021). Sebenarnya terdapat cara lain dalam pengelolaan *state* yang lebih umum digunakan pada pengembangan *frontend* khususnya saat menggunakan framework VueJS, yaitu dengan *library* bernama *Vuex*. *Vuex* adalah *library* pengelolaan *state* dengan menerapkan *flux* (Nelson, 2018). *Flux* adalah design pattern yang menyediakan pengelolaan *state* dengan aliran searah. Untuk menerapkan *design pattern flux*, dibutuhkan beberapa objek, di antaranya yaitu

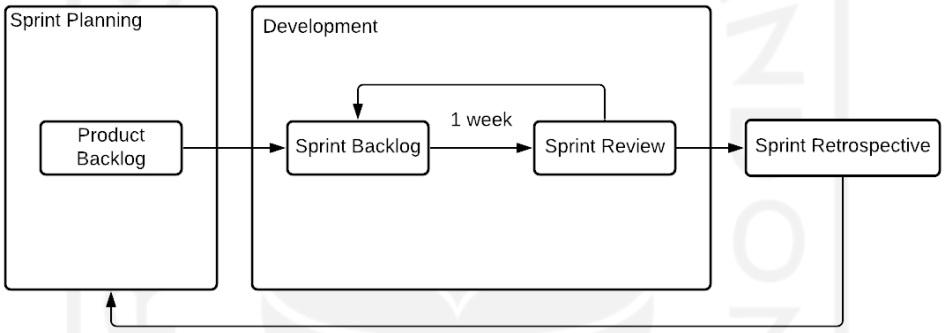
actions, *stores*, *dispatcher*, dan *views*. *Actions* merupakan objek untuk mendefinisikan aksi yang dibutuhkan untuk memutasi *state*. Stores merupakan objek untuk menyimpan *state*. *Dispatcher* merupakan objek untuk menghubungkan *actions* dengan *stores*. Sedangkan *views* adalah komponen antarmuka pada aplikasi. *Vuex* merupakan *library* yang memudahkan pengembang dalam mengelola *state* dengan menerapkan *flux*.

Lalu, mengapa pada pengembangan yang akan dilakukan pada fitur finansial tidak menerapkan *Vuex* untuk pengelolaan *state*? Menurut Duldulao et al. (2021), penggunaan *prop drilling* bukan merupakan pola pengelolaan *state* yang buruk untuk digunakan selama aplikasi hanya memiliki kedalaman komponen dua sampai tiga tingkat (Duldulao et al., 2021). Selain itu, penggunaan *prop drilling* mudah untuk diimplementasi karena tidak perlu melakukan *setup* pada aplikasi. Hal tersebut berbanding terbalik pada penggunaan *Vuex* untuk mengelola *state*. Penggunaan *Vuex* membutuhkan setup yang kompleks sebelum dapat digunakan. Pengembang juga harus memiliki pemahaman yang cukup dalam mengembangkan aplikasi menggunakan *Vuex*. Bahkan jika aplikasi hanya memiliki kedalaman komponen tidak lebih dari tiga, penggunaan *Vuex* merupakan tindakan yang berlebihan.

Berdasarkan dasar teori yang telah dipaparkan sebelumnya, dilaksanakanlah magang dengan menerapkan *prop drilling* dalam mengelola *state* pada fitur finansial aplikasi web Jala. Oleh karena itu, diharapkan pengembangan tersebut dapat bermanfaat bagi Jala Tech, pengembang, dan pengguna yang menggunakan aplikasi tersebut.

BAB III
PELAKSANAAN MAGANG

Metode pengembangan aplikasi yang digunakan di Jala Tech adalah *Scrum*. Metode *Scrum* merupakan satu di antara metode pengembangan perangkat lunak yang dilakukan secara *agile* dan sangat efektif dalam menangani proyek kompleks dengan jadwal yang padat (Pressman 2010). Ada beberapa tahap dalam pengembangan perangkat lunak menggunakan scrum di Jala Tech, yaitu *sprint planning*, *development*, *sprint review*, dan *sprint retrospective*.



Gambar 3.1 Ilustasi alur pengembangan perangkat lunak menggunakan *Scrum*

3.1 Sprint Planning

Di Jala Tech, *sprint planning* dilakukan dengan pertemuan yang dihadiri semua anggota tim yang terlibat dalam proyek. *Sprint planning* dipimpin oleh *product owner*. Tujuan dari *sprint planning* adalah untuk menentukan *product backlog*. *Product backlog* adalah daftar pekerjaan yang diprioritaskan oleh tim pengembang dalam implementasi proyek. Daftar *backlog* yang dikerjakan pada proyek pengembangan fitur finansial dapat dilihat pada Tabel 3.1.

Tabel 3.1 Daftar *backlog* proyek pengembangan fitur finansial

No.	***Backlog***
1.	Melakukan setup komponen *FinanceContainer*
2.	Menulis ulang kode komponen *Header*
3.	Menulis ulang kode komponen *TotalOverview*
4.	Menulis ulang kode komponen *IncomeOverview*
5.	Menulis ulang kode komponen *ExpenseOverview*

6.	Menulis ulang kode komponen *FinanceTabs*
7.	Menulis ulang kode *popup* modal tambah catatan keuangan

Selain menentukan *product backlog*, ditentukan juga estimasi durasi pengerjaan setiap *backlog*. Estimasi durasi ditentukan menyesuaikan kemampuan dan keputusan tim pengembang. Penentuan estimasi durasi ini bertujuan untuk mendapatkan estimasi waktu penyelesaian proyek. Di Jala Tech, estimasi durasi dihitung menggunakan poin. Satu poin sama dengan satu jam.

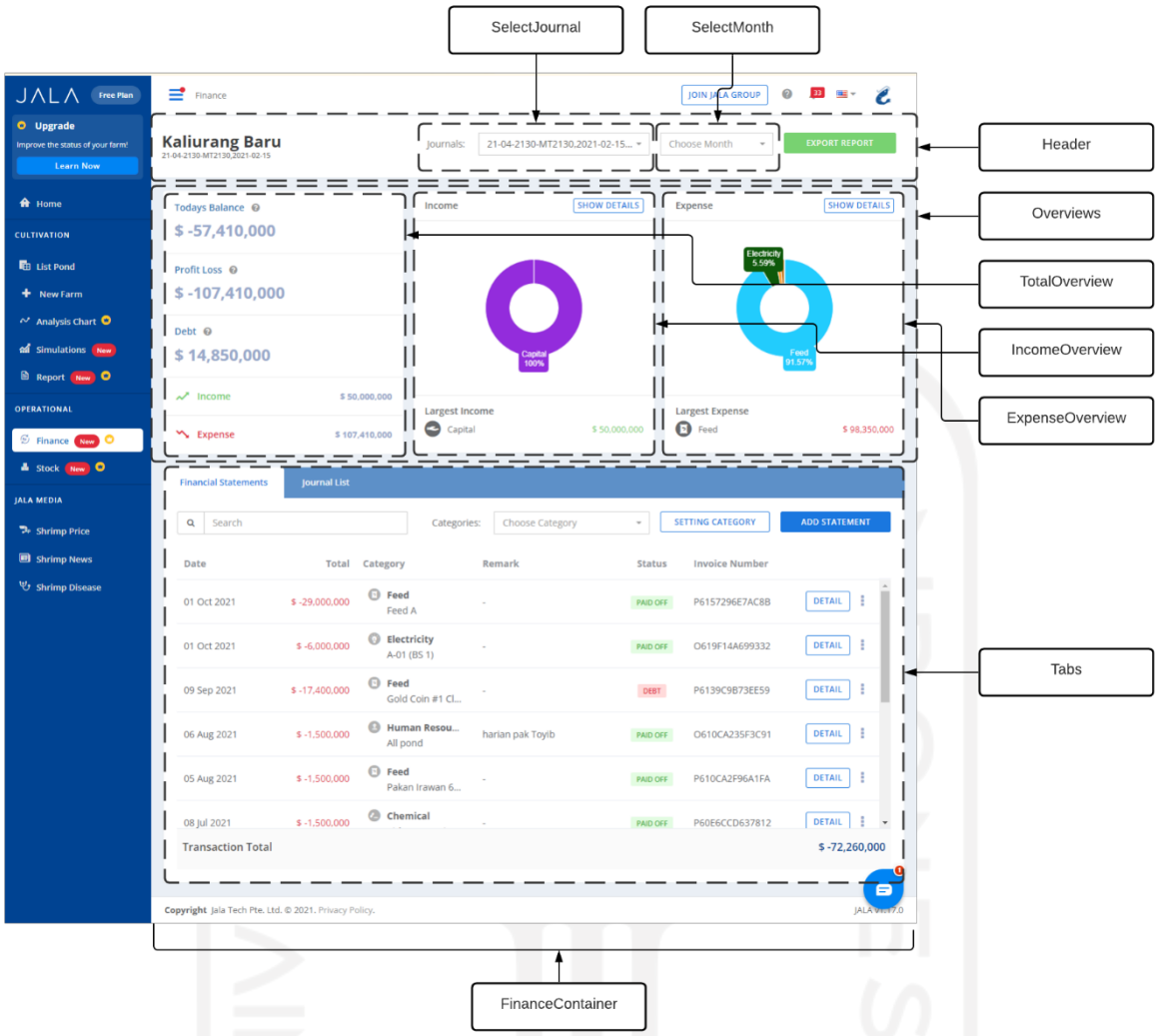
3.2 Development

Tahap *development* atau pengembangan merupakan tahap bagi tim pengembang untuk mengerjakan tugas yang telah didefinisikan sebelumnya pada tahap *sprint planning*. Namun, sebelum tim pengembang memulai pekerjaan, perlu dilakukan pendefinisian *sprint backlog* oleh *scrum master*. *Sprint backlog* merupakan sebagian dari *product backlog* yang akan dikerjakan pada *sprint* yang sedang dilakukan. Pendefinisian sprint *backlog* menyesuaikan estimasi waktu yang telah ditentukan pada setiap *backlog*.

Di Jala Tech, satu sprint dilakukan dalam satu minggu. Setiap sprintnya, pertemuan dilakukan sebanyak dua kali. Pertemuan pertama dilakukan untuk menentukan sprint *backlog*. Sedangkan pada pertemuan kedua, tim melakukan *sprint review*.

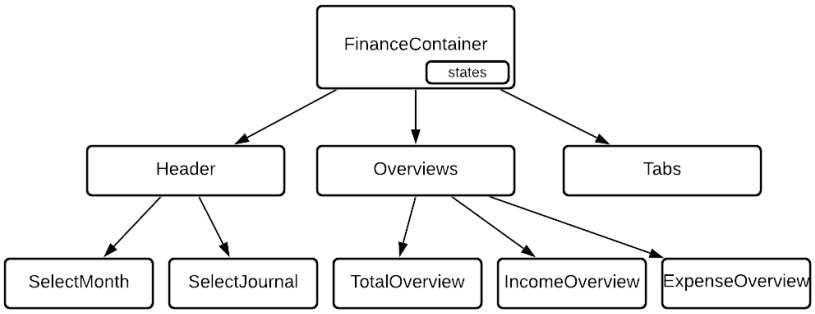
Sprint review dilakukan untuk memberikan *update* dari pekerjaan yang telah dikerjakan selama melakukan *sprint*. Hal tersebut bertujuan untuk mendapatkan masukan dari anggota tim lain dan menguji pekerjaan yang telah dikerjakan. Selain itu, *sprint review* dapat dijadikan wadah untuk melaporkan kendala atau masalah selama menjalani *sprint*. Dengan melaporkan kendala *sprint*, *scrum master* dapat melakukan improvisasi pada *backlog* sehingga *sprint backlog* pada *sprint* selanjutnya menyesuaikan improvisasi tersebut.

Sebelum melakukan pengembangan dengan mengimplementasi *prop drilling* pada fitur finansial aplikasi web Jala, pemagang menggambar desain struktur komponen dari fitur tersebut. Desain struktur komponen dibuat untuk dijadikan acuan dalam pembuatan dan penyusunan komponen serta penentuan hierarki komponen. Desain antarmuka dari fitur finansial yang sudah ada dijadikan acuan dalam pembentukan desain struktur komponen pada pengembangan yang akan dilakukan. Antarmuka serta komponen dari fitur finansial dapat dilihat pada Gambar 3.2.



Gambar 3.2 Antarmuka dan komponen di fitur finansial aplikasi web Jala

Setelah mengetahui komponen yang akan dibuat pada fitur finansial, maka dibuatlah desain struktur komponen. Dengan mengetahui struktur komponen, pemagang dapat mengetahui hierarki komponen dari fitur finansial. Hierarki komponen merupakan hal yang penting pada penggunaan pola *prop drilling* karena pengiriman *state* dilakukan oleh komponen utama kepada komponen-komponen di dalamnya menggunakan *props*. Desain struktur komponen fitur finansial aplikasi web Jala dapat dilihat pada Gambar 3.3.



Gambar 3.3 Desain struktur komponen fitur finansial aplikasi web Jala

Setelah mengetahui struktur komponen pada fitur finansial aplikasi web Jala, dilakukanlah pengembangan sesuai dengan *backlog* yang telah didefinisikan sebelumnya. Setiap *backlog* diberikan merupakan penulisan ulang kode setiap komponen pada fitur finansial aplikasi web Jala dengan menerapkan pengelolaan *state* menggunakan pola *prop drilling*.

3.2.1 Melakukan *Setup* pada Komponen *FinanceContainer*

Untuk menerapkan pola *prop drilling*, komponen induk utama diperlukan untuk menyimpan *state* yang digunakan pada sebuah fitur. Pada fitur finansial aplikasi web Jala, komponen induk utama dinamakan FinanceContainer. Sebelum membuat fail komponen, pemagang menentukan *state* yang digunakan pada fitur finansial. State yang digunakan pada fitur finansial adalah sebagai berikut:

Tabel 3.2 Daftar *state* yang digunakan pada fitur finansial

State	**Nilai Awal**
Profile	*Null*
Journal	*Null*
startDate	*‘’*
journalsPaginated	*{}*
financesPaginated	*{}*
financesTotal	*0*
currentCash	*0*
currentDebit	*0*
currentCredit	*0*
currentProfit	*0*

currentDebt	*0*
incomes	*[]*
expenses	*[]*

Setelah mengidentifikasi *state* yang digunakan pada fitur finansial, pemagang membuat fail "Container.vue" pada folder "finances_v2" untuk mendefinisikan komponen induk utama yang akan menyimpan semua *state* yang digunakan pada fitur finansial. Seperti pada Gambar 3.4, ditulis beberapa blok kode yaitu <template> dan <script>. Blok pada *tag* <template> akan berisi sintaksis HTML yang merupakan tempat pemagang membuat tampilan antarmuka. Pada *tag* tersebut pula komponen-komponen anak yang nantinya akan dibuat diletakkan.

```
<template>
  <!-- Anak Komponen di sini -->
</template>

<script>

export default {
  data => ({
    profile: null,
    journal: null,
    startDate: '',
    journalsPaginated: {},
    financesPaginated: {},
    financesTotal: 0,
    currentCash: 0,
    currentDebit: 0,
    currentCredit: 0,
    currentProfit: 0,
    currentDebt: 0,
    incomes: [],
    expenses: [],
  }),
}
</script>
```

Gambar 3.4 *Setup* pada komponen *FinanceContainer*

Dapat dilihat pada Gambar 3.4, semua *state* ditulis pada nilai pengembalian fungsi data pada objek *export default*. Penulisan tersebut merupakan cara pendefinisian *state* serta penetapan nilai awal *state* menggunakan *framework* VueJS. Semua properti dari pengembalian nilai fungsi data tersebut dapat digunakan sebagai *state*. *State* tersebut juga dapat dikirim kepada komponen anak menggunakan *props*. Jika nilai *state* diubah, maka komponen FinanceContainer akan di-*render* ulang dan tampilan antarmuka akan menyesuaikan berdasarkan *state* yang baru.

3.2.2 Menulis Ulang Kode Komponen *Header*

Komponen *Header* merupakan komponen yang ada di bagian atas halaman. Komponen *Header* berisi informasi terkait jurnal keuangan yang sedang dipilih serta terdapat dua komponen *Select* untuk mengganti jurnal dan bulan. Selain itu, komponen *Header* juga terdapat tombol untuk melakukan ekspor laporan keuangan.

Dengan adanya informasi jurnal dan bulan yang sedang dipilih, komponen *Header* memerlukan data untuk memberikan informasi tersebut. Dikarenakan *state* disimpan pada komponen FinanceContainer, maka komponen *Header* tidak membuat *state* baru untuk menampilkan data tersebut. Komponen *Header* memerlukan *props* untuk mendapatkan data dari komponen induknya. Maka terdapat dua *props* pada komponen *Header*, yaitu *journal* dan *startDate*. *Journal* adalah *props* yang dipakai untuk menyimpan data jurnal dipilih, sedangkan *startDate* adalah *props* yang dipakai untuk menyimpan data bulan yang dipilih. Bentuk implementasi *props* pada komponen *Header* dapat dilihat pada Gambar 3.5.

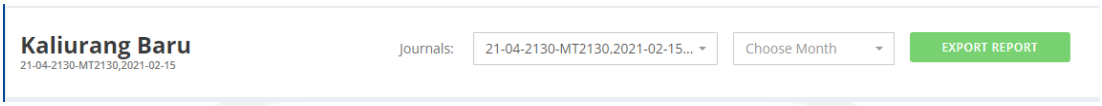
```
<template>
  <!— Kode HTML untuk tampilan Header -->
</template>

<script>
export default {
  props: {
    journal: {
      type: [Object, String], default: () => {},
    },
    startDate: {
      type: [String, String], default: () => null,
    },
  },
  methods: {
    goToReport() {
      if ('jalaAnalytic' in window) {
        jalaAnalytic.log('Finance: Export Report Clicked');
      }
      setTimeout(() => {
        window.location =
`/report?farm_id=${this.journal.farm_id}&journal_id=${this.journal.id}#finance`
      }, 300);
    }
  }
}
</script>
```

Gambar 3.5 Implementasi *props* pada komponen *Header*

Pada Gambar 3.5, pendefinisian *props* dapat dilakukan dengan menuliskan nama *props* pada blok kode di dalam properti *props*. Setiap *props* sendiri memiliki beberapa properti seperti *type* untuk menentukan tipe data dari *props* dan default untuk menentukan nilai *default* jika *props* tidak dikirim dari komponen induk. Terdapat properti baru yang berupa objek bernama *methods*.

Methods merupakan objek yang berisi metode dan fungsi yang dapat digunakan pada komponen. Terdapat metode *goToReport* yang berfungsi untuk mengarahkan halaman kepada halaman ekspor laporan keuangan. Hasil tampilan antarmuka dari komponen *Header* adalah seperti pada Gambar 3.6.



Gambar 3.6 Hasil tampilan antarmuka komponen *Header*

3.2.3 Menulis Ulang Kode Komponen *TotalOverview*

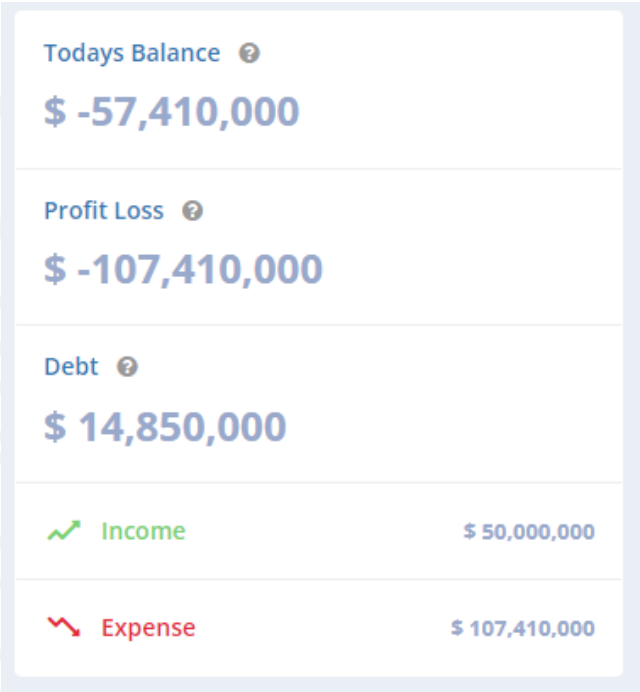
Komponen *TotalOverview* adalah komponen yang berfungsi untuk menampilkan rangkuman keuangan. Rangkuman keuangan tersebut antara lain saldo, laba rugi, utang, pemasukan, dan pengeluaran. Sama seperti komponen selain *FinanceContainer*, komponen TotalOverview tidak membuat *state* baru untuk menampilkan informasi rangkuman keuangan. Oleh karena itu, berdasarkan fungsi dari komponen *TotalOverview*, diperlukanlah *props* untuk menampilkan informasi tersebut. *Props* tersebut antara lain *currentCash, currentDebit, currentCredit, currentProfit*, dan *currentDebt*. Bentuk implementasi dari pendefinisian *props* dapat dilihat pada Gambar 3.7.

```
<template>
  <!-- Kode HTML untuk tampilan TotalOverview -->
</template>

<script>
export default {
  props: {
    currentCash: {
      type: Number, default: () => 0,
    },
    currentDebit: {
      type: Number, default: () => 0,
    },
    currentCredit: {
      type: Number, default: () => 0,
    },
    currentProfit: {
      type: Number, default: () => 0,
    },
    currentDebt: {
      type: Number, default: () => 0,
    },
  },
}
</script>
```

Gambar 3.7 Implementasi *props* pada komponen TotalOverview

Adapun hasil tampilan antarmuka dari komponen *TotalOverview* dapat dilihat pada Gambar 3.8. Terdapat lima baris tampilan yang menampilkan rangkuman informasi yang berasal dari *props* yang dikirim. Jika baris tersebut diklik, halaman akan diarahkan ke halaman detail setiap informasi.



Gambar 3.8 Hasil tampilan antarmuka komponen *TotalOverview*

3.2.4 Menulis Ulang Kode Komponen *IncomeOverview*

IncomeOverview merupakan komponen yang berfungsi untuk menampilkan informasi terkait pemasukan keuangan. Informasi yang ditampilkan berupa grafik *doughnut chart* dari jumlah pemasukan pada setiap kategori. Selain itu, komponen *IncomeOverview* juga menampilkan kategori dengan pemasukan terbesar. Untuk menampilkan informasi tersebut, komponen IncomeOverview memiliki satu *props*, yaitu *incomes*.

Untuk menampilkan grafik *doughnut chart*, pemagang menggunakan komponen yang telah dibuat sebelumnya. Komponen tersebut bernama *DoughnutChart*. Untuk menampilkan *DoughnutChart*, pemagang perlu melakukan impor agar dapat digunakan pada komponen *IncomeOverview*. Selain itu, *DoughnutChart* memerlukan data yang perlu dikirim melalui *props* antara lain *width, height, chartData*, dan *options*. Pada *props chartData*, objek yang dikirim harus memiliki *labels* dan *datasets*. Untuk memiliki data tersebut, mutasi terhadap data *incomes* perlu dilakukan. Mutasi dilakukan pada blok kode *computed*. *Computed* berisi fungsi-fungsi yang akan

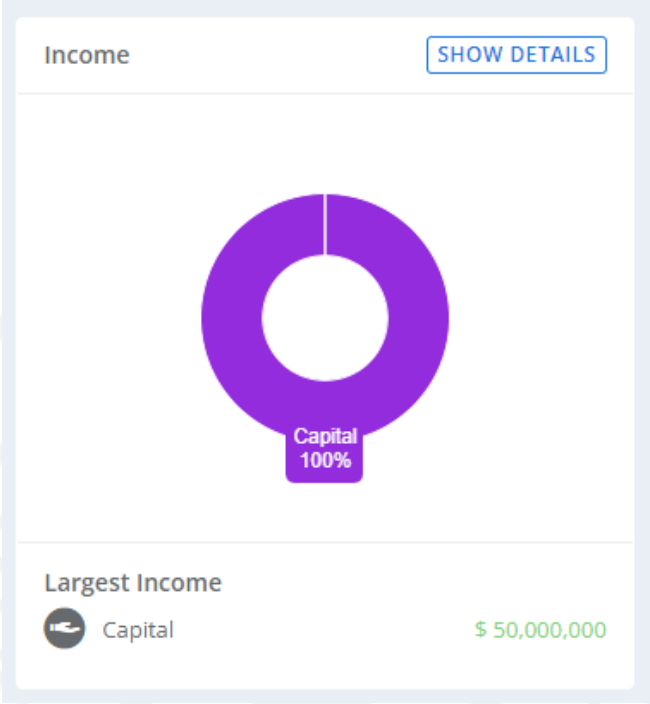
menjadi data yang dapat dipakai pada antarmuka aplikasi. *Computed* digunakan pada saat ingin memunyai sebuah data yang bergantung pada data yang lain. Contoh implementasi *DoughnutChart* dapat dilihat pada Gambar 3.9.

Untuk menampilkan pemasukan terbesar, maka pemasukan yang diambil adalah data pertama dari data *incomes*. Hal tersebut dikarenakan data *incomes* sudah diurutkan dari pemasukan yang terbesar hingga yang terkecil. Untuk mendapatkan kategori dari data *incomes* yang pertama, maka diperlukan metode *category* yang memiliki parameter *income*. Pembuatan metode tersebut dikarenakan butuh beberapa baris kode untuk mendapatkan kategori tersebut.

```
<template>
  <div>
    <doughnut-chart
      :chart-data="datacollection"
      :options="options"
    />
    <h4 class="m-b-none m-t-none"> {{ $t('finances.largest_income' }} </h4>
    <i :class="category(incomes[0]).icon"></i>
    <span class="category-label m-l-sm"> {{ category(incomes[0]).name }} </span>
    <span class="text-success"> Rp {{ incomes[0].debit }} </span>
  </div>
</template>
<script>
import DoughnutChart from '../../DoughnutChart';
export default {
  props: {
    incomes: {
      type: Array, default: () => [],
    },
  },
  data: () => ({
    options: {
      responsive: true,
      legend: false,
      layout: {
        padding: 24
      },
    },
  }),
  computed: {
    datacollection () {
      // Kode di sini
    },
  },
  methods: {
    category (income) {
      // Kode di sini
    },
  },
}
</script>
```

Gambar 3.9 Implementasi komponen *IncomeOverview*

Adapun hasil tampilan antarmuka dari komponen *IncomeOverview* dapat dilihat pada Gambar 3.10.



Gambar 3.10 Hasil tampilan antarmuka komponen *IncomeOverview*

3.2.5 Menulis Ulang Kode Komponen *ExpenseOverview*

ExpenseOverview adalah komponen yang berfungsi untuk menampilkan informasi terkait pengeluaran keuangan. Sama seperti komponen *IncomeOverview*, *ExpenseOverview* menampilkan grafik dari pemasukan oleh setiap kategori menggunakan *DoughnutChart*. Selain itu, *ExpenseOverview* juga menampilkan kategori dengan pengeluaran terbesar. Pada komponen *ExpenseOverview*, *props* yang dimiliki adalah *expenses*.

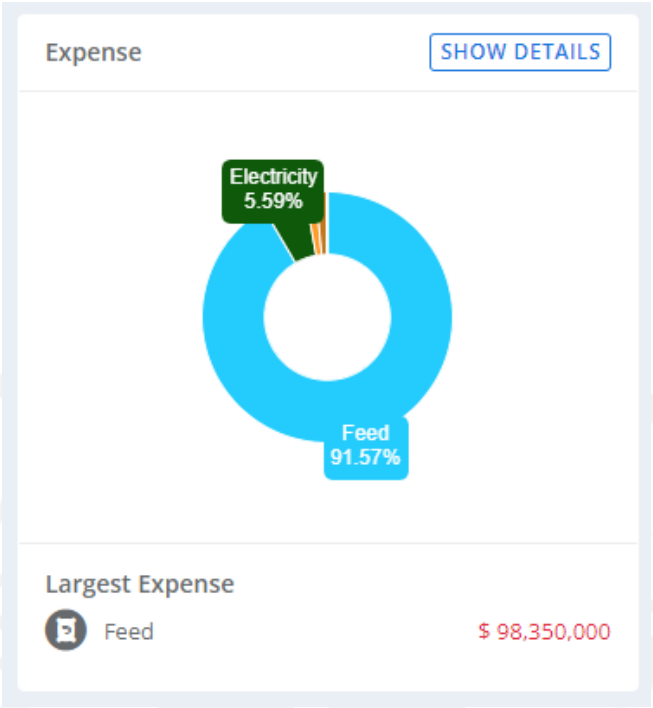
Untuk menampilkan *DoughnutChart*, pemagang perlu melakukan impor agar dapat digunakan pada komponen *ExpenseOverview*. Selain itu, *DoughnutChart* memerlukan data yang perlu dikirim melalui *props* antara lain *width, height, chartData*, dan *options*. Pada *props chartData*, objek yang dikirim harus memiliki *labels* dan *datasets*. Untuk memiliki data tersebut, mutasi terhadap data *expenses* perlu dilakukan.

Untuk menampilkan pengeluaran terbesar, maka pengeluaran yang diambil adalah data pertama dari data *expenses*. Hal tersebut dikarenakan data *expenses* sudah diurutkan dari pemasukan yang terbesar hingga yang terkecil. Untuk mendapatkan kategori dari data *expense* yang pertama, maka diperlukan metode *category* yang memiliki parameter *expense*. Contoh implementasi untuk menampilkan pemasukan terbesar dapat dilihat pada Gambar 3.11.

```
<template>
  <div>
    <doughnut-chart
      :chart-data="datacollection"
      :options="options"
    />
    <h4 class="m-b-none m-t-none"> {{ $t('finances.largest_expense') }} </h4>
    <i :class="category(expenses[0]).icon"></i>
    <span class="category-label m-l-sm"> {{ category(expenses[0]).name }} </span>
    <span class="text-danger"> Rp {{ expenses[0].credit }} </span>
  </div>
</template>
<script>
import DoughnutChart from '../../DoughnutChart';
export default {
  props: {
    expenses: {
      type: Array, default: () => [],
    },
  },
  data: () => ({
    options: {
      responsive: true,
      legend: false,
      layout: {
        padding: 24
      },
    },
  }),
  computed: {
    datacollection () {
      // Kode di sini
    },
  },
  methods: {
    category (expense) {
      // Kode di sini
    },
  },
}
</script>
```

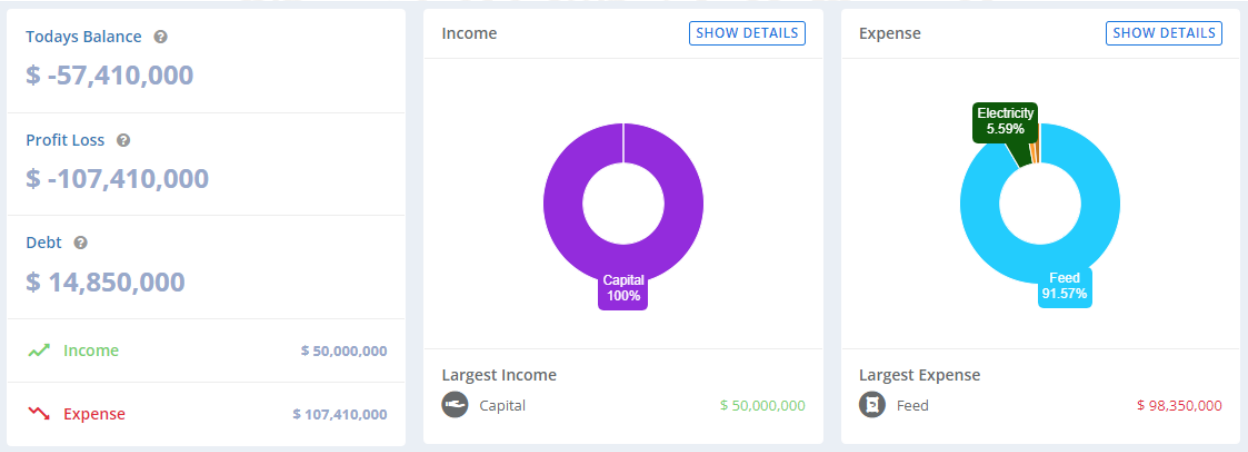
Gambar 3.11 Implementasi komponen *ExpenseOverview*

Adapun hasil tampilan antarmuka dari komponen *ExpenseOverview* dapat dilihat pada Gambar 3.12.



Gambar 3.12 Hasil tampilan antarmuka komponen *ExpenseOverview*

Setelah membuat ketiga komponen untuk menampilkan ikhtisar dari finansial tambak, ketiga komponen tersebut diorkestrasikan menjadi satu komponen yaitu *Overviews*. Komponen tersebut yang kemudian diletakkan di komponen induk FinanceContainer. Dengan menerapkan *props drilling*, *state* dikirim dari komponen yang terluar, yaitu *FinanceContainer* mengirim *state* melalui *props* kepada komponen *Overviews* terlebih dahulu, kemudian *Overviews* mengirim data kepada komponen *TotalOverview, IncomeOverview,* dan *ExpenseOverview* melalui *props*. Hasil tampilan antarmuka dari komponen *Overviews* dapat dilihat pada Gambar 3.13.



Gambar 3.13 Hasil tampilan antarmuka komponen *Overviews*

3.2.6 Menulis Ulang Kode *Tab* Catatan Keuangan

FinanceTabs adalah komponen yang berisi *tabs*. *Tabs* adalah komponen yang memungkinkan beberapa tampilan atau panel dimuat dalam satu *Tab* atau jendela. *FinanceTabs* menampilkan dua *Tab*, yaitu untuk menampilkan catatan keuangan dan menampilkan daftar jurnal keuangan yang ada.

Pada Tab catatan keuangan menampilkan tabel yang menggunakan *library ElementUI* untuk memudahkan membuat tampilan tabel yang responsif. Tabel catatan keuangan memiliki beberapa kolom di antaranya adalah tanggal, total, kategori, catatan, status, nomor *invoice*, dan aksi. Setiap baris data yang tampil memiliki tombol yang dapat mengeluarkan *popup* modal untuk melihat detail dari catatan keuangan. Terdapat pula total transaksi pada bagian bawah tabel. Data catatan keuangan pada tabel ditampilkan dengan paginasi sehingga tabel menampilkan data setiap seratus baris. Untuk menampilkan tabel, maka diperlukan beberapa *props* seperti *journal*, *financePaginated, financesTotal*, dan *financeKeyword*. Data catatan keuangan juga dapat disaring berdasarkan kata kunci dan kategori. Oleh karena itu, setiap menyaring dengan memasukkan kata kunci atau kategori, maka perlu dilakukan pengambilan data dari *backend*. Dikarenakan komponen *FinanceTabs* mendapatkan data dari *props*, maka komponen tidak dapat langsung mengubah data *props* secara langsung. Oleh karena itu, perlu dilakukan *emit* untuk melempar *event* kepada komponen induk dan memberi tahu bahwa perlu dilakukan pengambilan data kembali berdasarkan kata kunci atau kategori. Implementasi Tab catatan keuangan dapat dilihat pada Gambar 3.13.

Pada Tab daftar jurnal keuangan, data yang ditampilkan juga berupa tabel yang menggunakan *ElementUI*. Kolom yang ada di antaranya adalah nama, saldo, tanggal buka, tanggal tutup, tambak, orang yang membuat, batch, dan aksi. Untuk menampilkan data jurnal keuangan, diperlukan beberapa *props* yaitu *journalsPaginated* dan *journalKeyword*. Data jurnal keuangan juga dapat disaring berdasarkan kata kunci. Oleh karena itu, perlu dilakukan *emit* untuk melempar *event* kepada komponen induk untuk memberi tahu bahwa perlu dilakukan pengambilan data kembali berdasarkan kata kunci. Implementasi Tab daftar jurnal keuangan dapat dilihat pada Gambar 3.14.

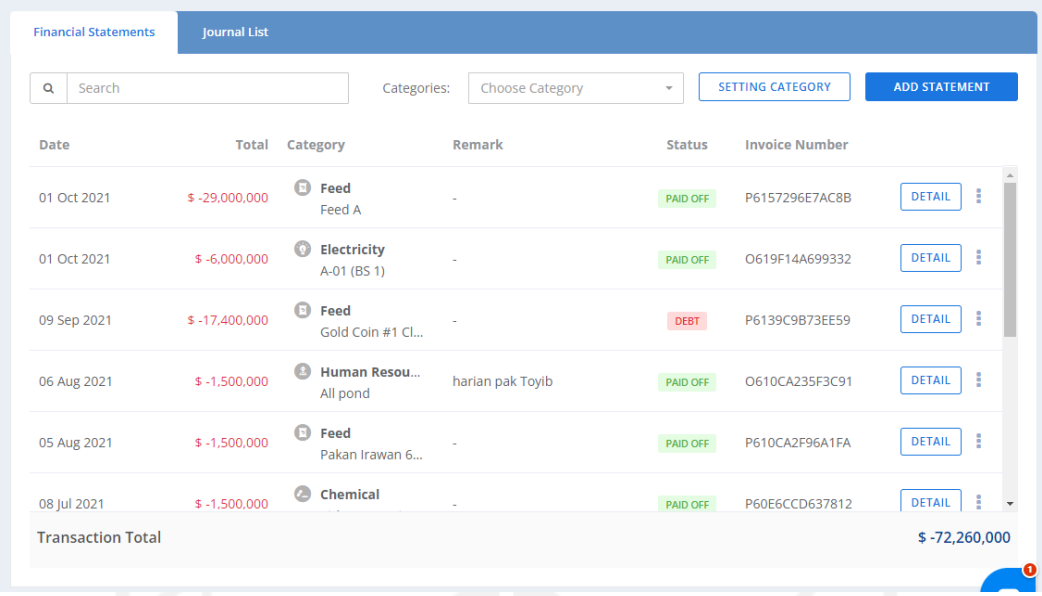
```
<template>
  <div class="tabs-container m-b-md">
    <el-table
      :data="financesPaginated.data || []"
    >
      <el-table-column
        prop="total"
        :label="$t('finances.total')"
```

```
      >
        Rp {{ row[column.property] }}
        </template>
      </el-table-column>
      <el-table-column
        prop="category"
        :label="$t('finances.category')"
      >
        <span>{{ row[column.property].name_translated }}</span>
        <span>{{ row.product_text }}</span>
      </el-table-column>
      <el-table-column
        prop="remark"
        :label="$t('finances.remark')"
      >
          <span>{{ row[column.property] || '-' }}</span>
      </el-table-column>
      <el-table-column
        prop="status"
        :label="$t('finances.status')"
      >
        <span>{{ $t(getFinanceStatus(row).value) }}</span>
      </el-table-column>
    </el-table>
  </div>
</template>

<script>
export default {
  props: {
    profile: {
      type: Object, default() { return {}; }
    },
    journal: {
      type: Object, default() { return {}; }
    },
    financesPaginated: {
      type: Object, default: () => ({}),
    },
    financesTotal: {
      type: Number, default: () => 0,
    },
    journalsPaginated: {
      type: Object, default: () => ({}),
    },
    financeKeyword: String,
    journalKeyword: String,
  },
};
</script>
```

Gambar 3.14 Implementasi komponen *FinanceTabs*

Adapun hasil tampilan antarmuka dari komponen *FinanceTabs* dapat dilihat pada Gambar 3.15.



Gambar 3.15 Hasil tampilan antarmuka komponen *FinanceTabs*

3.2.7 Menulis Ulang Kode Popup Modal Tambah Catatan Keuangan

Popup modal merupakan jendela atau panel yang terbuka di atas jendela utama. Komponen *popup* modal diletakkan pada komponen FinanceContainer. *Popup* modal yang berfungsi untuk menambah catatan keuangan dinamakan *CreateFinanceModal*.

CreateFinanceModal memiliki dua Tab di dalamnya, yaitu *ExpenseContainer* dan *IncomeContainer*. *ExpenseContainer* adalah formulir untuk menambah pengeluaran baru sedangkan *IncomeContainer* adalah formulir untuk menambah pemasukan baru. Implementasi komponen *CreateFinanceModal* dapat dilihat pada Gambar 3.16.

```
<template>
  <modal :value="value" @input="$emit('input', $event)">
    <expense-container v-if="journal" :journal="journal" @close="$emit('input',
false)" @expense-created="$emit('finance-created', $event)" />
    <income-container v-if="journal" :journal="journal" @close="$emit('input',
false)" @income-created="$emit('finance-created', $event)"  />
  </modal>
</template>

<script>
import ExpenseContainer from './ExpenseContainer';
import IncomeContainer from './IncomeContainer';
export default {
  props: {
    value: { type: Boolean, default: false },
    journal: { type: Object, default: null },
  },
}
</script>
```

Gambar 3.16 Implementasi *popup* modal tambah pengeluaran

Seperti yang terlihat pada Gambar 3.16, kode tampilan pada *CreateFinanceModal* dibungkus oleh komponen Modal. Komponen modal berfungsi untuk memungkinkan komponen di dalamnya muncul di dalam jendela *popup* modal. Pengembang dapat membuat tampilan secara fleksibel di dalam *tag* komponen Modal. Hal tersebut dikarenakan pada pembuatan komponen Modal, adanya penggunaan *tag slot*.

Tag slot digunakan untuk menentukan lokasi dari kode yang nantinya dimasukkan tampilan yang ditulis di dalam *tag* komponen. Pada saat pengembang mengisi tampilan di dalam tag komponen, *slot* akan digantikan oleh tampilan tersebut. Hal tersebut memberikan kefleksibilitasan dalam membuat tampilan yang berada di dalam sebuah komponen lain.

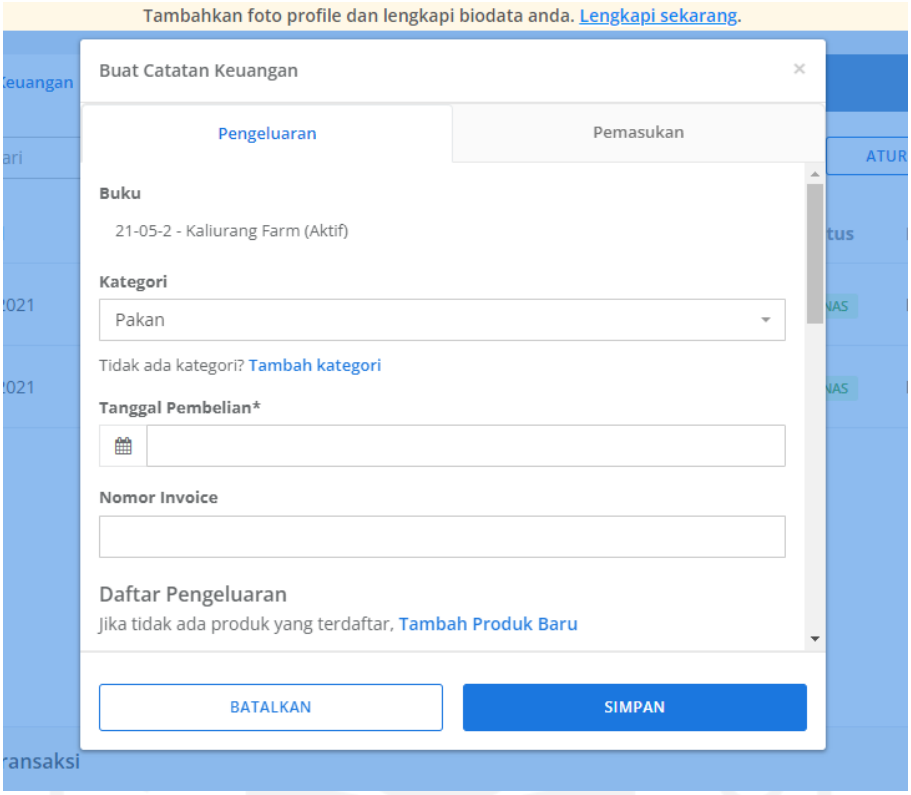
Penggunaan *tag slot* pada komponen Modal agar perilaku *popup* modal dapat digunakan oleh komponen modal yang lain sehingga tidak perlu menulis ulang komponen yang memiliki perilaku yang sama. Selain itu, penggunaan *tag slot* juga memberikan kefleksibilitasan dalam membuat tampilan di dalam modal. Implementasi komponen Modal dapat dilihat pada Gambar 3.17.

```
<template>
  <div class="modal__container" @click="$emit('close')">
    <slot name="title" />
    <slot />
  </div>
</template>

<script>
export default {
  props: {
    size: { type: String, default: () => 'medium' }
  },
  data: () => ({
    body: document.querySelector('body'),
  }),
}
</script>
```

Gambar 3.17 Implementasi komponen Modal

Adapun hasil tampilan antarmuka dari komponen *CreateFinanceModal* dapat dilihat pada Gambar 3.18.



Gambar 3.18 Hasil tampilan antarmuka komponen *CreateFinanceModal*

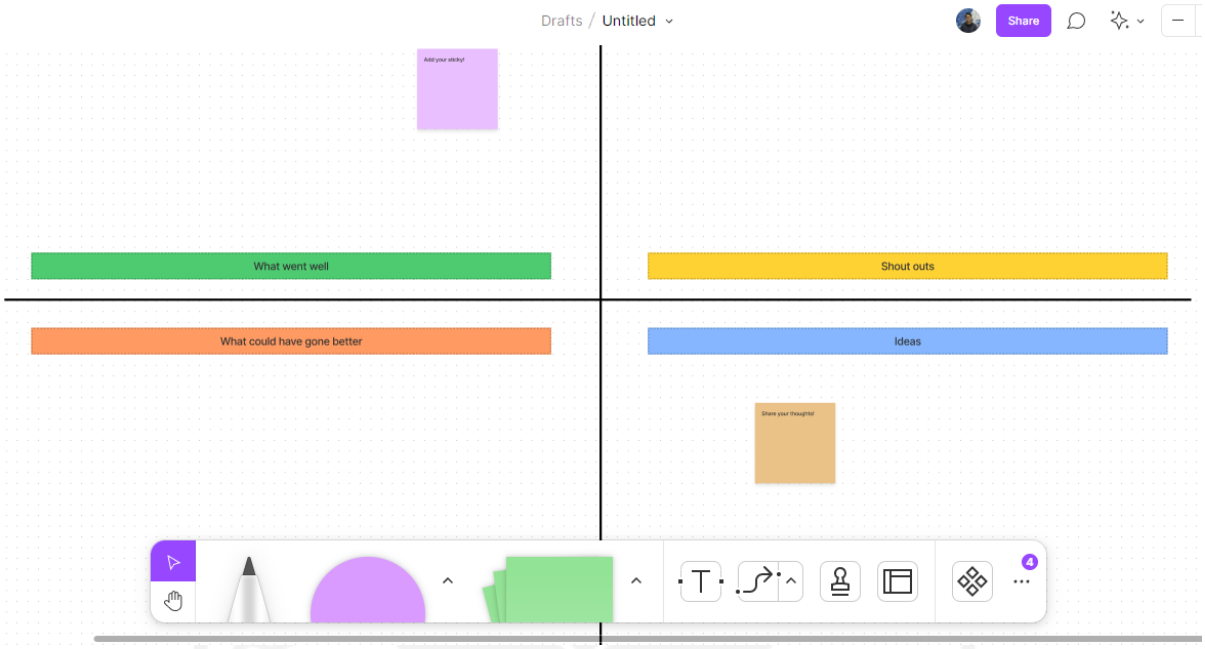
3.3 *Sprint Retrospective*

Setelah menyelesaikan semua *backlog*, maka dilakukanlah *sprint retrospective*. *Sprint retrospective* merupakan pertemuan yang dihadiri tim proyek untuk melakukan introspeksi kepada individu maupun tim. Hal tersebut bertujuan untuk meningkatkan kinerja pada *sprint* di masa mendatang.

Pada *sprint retrospective*, tim menentukan beberapa hal untuk meningkatkan kinerja pada *sprint* mendatang. Hal pertama adalah menentukan hal yang dilakukan pada *sprint* sekarang dan akan sangat membantu jika hal tersebut dilakukan kembali di masa mendatang. Penentuan tersebut bertujuan untuk mempertahankan hal yang baik dikerjakan di masa sekarang. Selain itu, *sprint retrospective* juga menentukan hal yang dilakukan pada *sprint* sekarang akan tetapi memiliki dampak negatif karena telah melakukannya. Tindakan tersebut akan dijadikan bahan evaluasi dan tidak diulang kembali pada *sprint* di masa mendatang. *Sprint retrospective* juga menentukan hal yang belum pernah dilakukan sebelumnya namun akan berpotensi meningkatkan kinerja di *sprint* mendatang.

Di Jala, *sprint retrospective* dilakukan secara daring dengan memanfaatkan *Figjam*. *Figjam* adalah satu di antara fitur yang terdapat pada aplikasi web *Figma*. Dengan *Figjam*, anggota tim

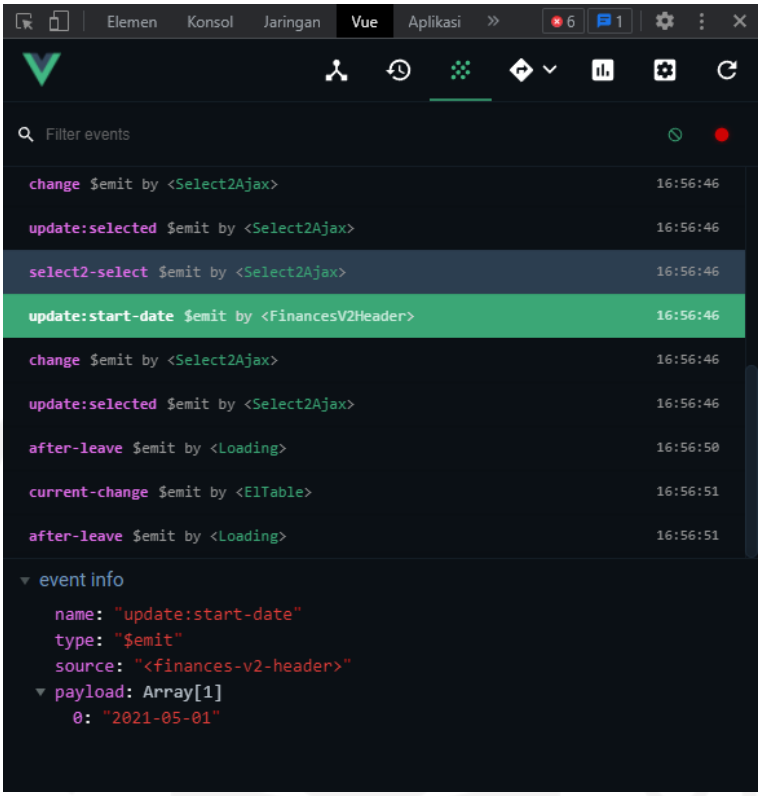
dapat memasang *sticky note* layaknya berdiskusi secara luring di papan tulis. Contoh tampilan antarmuka dari *Figjam* dapat dilihat pada Gambar 3.19.



Gambar 3.19 Tampilan antarmuka *Figjam*

3.4 Dampak Implementasi

Implementasi yang telah dikerjakan oleh pemagang masuk ke dalam *production*. Oleh karena itu, kontribusi yang dilakukan pemagang telah resmi aktif pada aplikasi web Jala saat ini. Menurut hasil wawancara dari *Lead of Software Engineer* Jala Tech, Farid Inawan, mengatakan bahwa penyimpanan *state* yang terpusat di komponen induk fitur finansial memudahkan pengembang dalam mengidentifikasi *state* yang digunakan pada fitur tersebut. Oleh karena itu, pengembangan yang dilakukan menjadi lebih cepat karena pengembang langsung mengetahui *state* dan alur perubahannya. Selain itu, hal tersebut juga dapat meminimalkan adanya *state* yang tidak saling sinkron.



Gambar 3.20 Pelacakan *event* menggunakan Vue Devtools

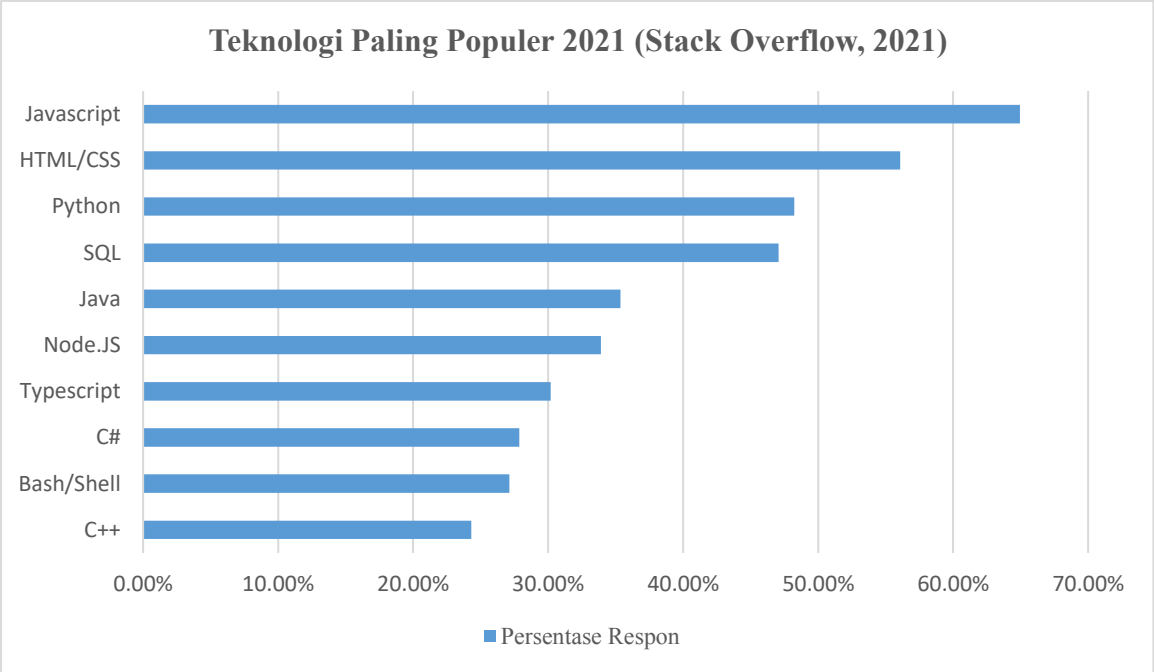
Berdasarkan Gambar 3.20, *event* pada *state* tercatat pada Vue Devtools. Data yang tercatat pada Vue Devtools antara lain nama *event*, komponen yang melakukan, dan waktu *event*. Data yang dikirim pada *event* juga tercatat pada Vue Devtools. Menurut Farid Inawan, kemampuan dalam melacak perubahan *state* juga memudahkan pengembang dalam mengidentifikasi perilaku *state* yang tidak sesuai ekspektasi sehingga mempercepat proses *debugging* dan penyelesaian masalah.

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Teknis

Selama melaksanakan magang, pemagang mendapatkan beberapa pelajaran dalam hal teknis yang tidak diajarkan di perkuliahan. Walaupun tidak diajarkan di perkuliahan, kemampuan tersebut sangat penting dalam perusahaan khususnya dalam pengembangan *frontend*.

Di perkuliahan, kemampuan memrogram dengan bahasa javascript sangat sedikit. Padahal, perusahaan yang memiliki peran pengembang *frontend* sangat membutuhkan kemampuan memrogram dengan bahasa javascript. Bahkan menurut Stack Overflow Developer Survey (2021), Javascript merupakan bahasa pemrograman paling populer peringkat pertama dengan persentase suara sebesar 64.96%.



Gambar 4.1 Peringkat teknologi paling populer 2021

Sumber: Stack Overflow (2021)

Di perkuliahan hanya diajarkan mengenai penggunaan javascript untuk menampilkan *alert* dan hal kecil lainnya yang sebetulnya tidak terlalu dipakai pada pengembangan aplikasi di perusahaan. Perusahaan lebih membutuhkan kemampuan pemrogram dalam penggunaan

framework javascript, manipulasi DOM (*Document Object Model*), dan pemahaman fundamental yang mendalam dalam memrogram javascript. Namun, pemagang merangkum beberapa pelajaran yang didapat selama magang di Jala Tech. Pelajaran yang didapat adalah berupa pemahaman terkait penggunaan *emit* pada VueJS, penggunaan *multiple promises* pada javascript, penggunaan *high order function* pada javascript, penggunaan *setter* dan *getter* pada objek javascript, dan penggunaan *spread operator* pada javacript.

4.1.1 Penggunaan Emit pada VueJS

Emit merupakan metode global pada VueJS yang berfungsi untuk mengirim event dari komponen anak ke komponen induk. Emit biasanya digunakan pada saat komponen anak melakukan aksi yang memengaruhi komponen induknya. Hal tersebut dapat berguna pada saat komponen anak yang menerima *state* melalui *props* ingin mengubah data tersebut. Dikarenakan tidak bisa mengubah secara langsung pada komponen anak tersebut, maka komponen anak perlu mengirim event ke komponen induknya lalu *state* tersebut diubah oleh komponen induk yang menyimpan *state* tersebut. Contoh penggunaan emit pada komponen anak dapat dilihat pada Gambar 4.2. Sedangkan contoh penanganan event yang dikirim melalui emit pada komponen induk dapat dilihat pada Gambar 4.3.

```
<template>
      <input :value=”name” @input=”$emit(‘update:name’, $event.target.value)” />
</template>

<script>
export default {
      name: ‘ChildComponent’,
      props: [‘name’],
}
</script>
```

Gambar 4.2 Contoh penggunaan emit pada komponen anak

```
<template>
       <child-component :name=”name” @update:name=”name = $event” />
</template>

<script>
export default {
       name: ‘ParentComponent’,
       data () {
              name: ‘Albarra’
       }
}
</script>
```

Gambar 4.3 Contoh penanganan emit pada komponen induk

4.1.2 Penggunaan Multiple Promises pada Javascript

Javascript dapat memiliki fitur yang dapat mengeksekusi kode tidak secara berurutan yang dinamakan *concurrency*. Salah satu kelas yang dapat menjalankan concurrency pada javascript adalah promise. Promise adalah kelas yang memungkinkan pengembang menjalankan eksekusi kode secara asynchronous. Biasanya promise digunakan untuk aksi yang perlu memakan waktu. Promise akan mengembalikan nilai setelah eksekusi selesai dijalankan.

Secara default, eksekusi kode yang dijalankan menggunakan promise akan dijalankan secara paralel sehingga kode di bawahnya langsung dijalankan walaupun eksekusi kode pada promise belum selesai dijalankan. Jika ingin menjalankan kode setelah kode promise selesai dijalankan, maka perlu memasukkan syntax khusus berupa await tepat sebelum kode promise. Penggunaan await akan memungkinkan kode setelahnya menunggu kode promise selesai sebelum mengeksekusi kode tersebut.

Selain itu, javascript juga memungkinkan eksekusi kode dijalankan secara paralel dan berurutan bersamaan. Fungsi yang memungkinkan hal tersebut terjadi adalah fungsi Promise.all(). Fungsi Promise.all() memiliki argumen berupa array yang setiap element array-nya merupakan fungsi promise. Promise.all() mengembalikan nilai jika semua promise di dalam argumen selesai dieksekusi. Oleh karena itu, semua kode promise di dalam argumen dieksekusi secara paralel. Namun, kode setelahnya akan dijalankan setelah kode Promise.all() selesai dijalankan dengan menambahkan await sebelum Promise.all(). Contoh penggunaan Promise.all() dapat dilihat pada Gambar 4.4.

```
await Promise.all([
       fetchFarms(),
       fetchPonds(),
       fetchFinances()
]);

console.log('Kode selesai dijalankan');
```

Gambar 4.4 Contoh penggunaan Promise.all() pada javascript

4.1.3 Penggunaan High Order Function pada Javascript

High order function adalah fungsi yang dapat menjadikan fungsi sebagai parameter ataupun hasil pengembalian. Javascript menyediakan beberapa high order function yang dapat langsung digunakan. Tiga di antara fungsi tersebut adalah map, filter, dan reduce.

Fungsi map berfungsi untuk memetakan atau memformat ulang array. Map memiliki parameter berupa fungsi yang harus mengembalikan nilai dengan tipe data apa pun. Fungsi tersebut nantinya akan dijalankan secara berulang pada setiap elemen dari array sehingga setiap pengembalian akan menjadi elemen pada array baru. Contoh penggunaan map dapat dilihat pada Gambar 4.5.

```
const names = ['albarra', 'hersa', 'ika', 'farhan'];

const namesWithIndex = names.map(function(element, index){
       return `${index}. ${element}`;
});

console.log(namesWithIndex);
// Hasil: ['0. albarra', '1. hersa', '2. ika', '3. farhan']
```

Gambar 4.5 Contoh penggunaan fungsi map pada javascript

Fungsi filter berfungsi untuk menyaring array sesuai kondisi tertentu. Filter memiliki parameter berupa fungsi yang harus mengembalikan nilai boolean. Pengembalian nilai false akan disaring dan tidak akan dimasukkan pada array baru yang dibuat pada fungsi filter. Contoh penggunaan fungsi filter dapat dilihat pada Gambar 4.6.

```
const names = [‘albarra’, ‘hersa’, ‘ika’, ‘farhan’];

const namesWithoutAlbarra = names.filter(function(element){
       return element !=== ‘albarra’;
});

console.log(namesWithoutAlbarra);
// Hasil: [‘hersa’, ‘ika’, ‘farhan’]
```

Gambar 4.6 Contoh penggunaan fungsi filter pada javascript

Fungsi reduce berfungsi untuk mengeksekusi setiap elemen pada array. Tidak seperti fungsi map dan filter, fungsi reduce hanya mengembalikan satu nilai saja. Kode yang dieksekusi pada setiap elemen ditulis di dalam parameter pada fungsi reduce. Contoh penggunaan fungsi filter dapat dilihat pada Gambar 4.7.

```
const numbers = [1, 2, 3, 4, 5];

const accumulation = numbers.reduce(function(acc, cur){
       return acc + cur;
});

console.log(accumulation);
// Hasil: 15
```

Gambar 4.7 Contoh penggunaan fungsi reduce pada javascript

4.1.4 Penggunaan Setter dan Getter pada Objek Javascript

Pada javascript, membuat sebuah objek tidak perlu membuat kelas terlebih dahulu seperti bahasa pemrograman berorientasi objek lainnya. Membuat objek di javascript dapat langsung didefinisikan melalui nilai yang memiliki properti. Oleh karena itu, pengembang dapat sangat fleksibel menaruh properti apa pun pada sebuah objek karena tidak memiliki struktur pasti.

Seperti pada bahasa pemrograman berorientasi objek lainnya, sebuah objek dapat memiliki setter dan getter. Di javascript, pendefinisian setter dan getter dapat langsung ditulis pada saat membuat objek. Contoh penulisan setter dan getter dapat dilihat pada Gambar 4.8.

```
const albarra = {
       firstName: 'Albarra',
       lastName: 'Naufala',
       major: 'Informatika',
       set fullName (newFullName) {
              const fName = newFullName.split(' ')[0];
              const lName = newFullName.split(' ')[1];
              this.firstName = fName;
              this.lastName = lName;
       }
       get fullName () { return `${firstName} ${lastName}` },
};
```

Gambar 4.8 Contoh penggunaan setter dan getter pada javascript

4.1.5 Penggunaan Spread Operator pada Javascript

Spread operator adalah operator yang ditulis dengan tiga titik (...) yang dapat ditempatkan sebelum array dan objek. Spread operator dapat berfungsi untuk memecah nilai array dan objek. Hasil pecahan yang dihasilkan spread operator dapat menjadi variabel baru. Contoh penggunaan spread operator dapat dilihat pada Gambar 4.9.

```
function sum (x, y, z) {
       return x + y + z;
}

const numbers = [1, 2, 3];
const accumulation = sum(...numbers);
console.log(accumulation);

// Hasil: 6
```

Gambar 4.9 Contoh penggunaan spread operator pada javascript

4.2 Non Teknis

Selain mendapatkan pengalaman teknis, pemagang mendapatkan pengalaman non teknis yang bermanfaat untuk kehidupan sehari-hari terutama bekal untuk menjalani dunia pekerjaan di masa mendatang. Pengalaman tersebut di antaranya adalah pengalaman bekerja secara profesional, memanajemen diri, pengalaman beradaptasi, mendapatkan pandangan terhadap dinamika industri, mendapatkan wadah aktualisasi keilmuan, bekerja di bawah tekanan, pengalaman presentasi atau berbicara di depan umum, pengalaman belajar dari mentor, pandangan terhadap organisasi perusahaan, dan .

4.2.1 Pengalaman Bekerja Secara Profesional

Satu di antara kemampuan yang tidak didapatkan di bangku perkuliahan adalah pengalaman bekerja secara profesional. Dalam menjalani magang, seseorang harus bekerja secara profesional sesuai deskripsi pekerjaan yang telah diberikan. Dalam sebuah tim, kemampuan bekerja sama serta komunikasi bersama kolega dibutuhkan untuk mencapai tujuan bersama. Di Jala Tech, pemagang bekerja bersama pengembang lain, desainer produk, serta atasan. Oleh karena itu, Jala Tech membantu pemagang meningkatkan kemampuan tersebut.

4.2.2 Pengalaman Memanajemen Diri

Selama menjalani magang, pemagang mendapatkan tantangan untuk harus dapat memanajemen diri karena kegiatan magang dilakukan secara bersamaan dengan kuliah. Pemagang harus dapat memilih prioritas agar mengetahui pekerjaan yang harus dikerjakan terlebih dahulu. Hal tersebut bermanfaat untuk memudahkan pemagang dalam membagi waktu antara magang dan kuliah. Di magang sendiri pasti tidak hanya terdapat satu pekerjaan saja. Melainkan dalam sekali *sprint* atau satu minggu, pemagang mendapatkan lebih dari satu pekerjaan yang perlu diselesaikan. Oleh karena itu, manajemen diri yang baik sangat memengaruhi suksesnya magang dan kuliah.

4.2.3 Pengalaman Beradaptasi

Pada saat melaksanakan magang, lingkungan perusahaan merupakan sesuatu yang baru bagi pemagang. Setiap perusahaan dapat memiliki kultur kerja yang berbeda-beda. Oleh karena itu, menjadi sebuah tantangan pemagang untuk dapat beradaptasi dengan lingkungan perusahaan. Adaptasi merupakan satu di antara aspek penting dalam kelancaran magang. Selain untuk mempermudah komunikasi, pemagang dapat memiliki inisiatif yang sesuai dengan kultur perusahaan.

Adaptasi terhadap teknologi yang digunakan pada perusahaan juga merupakan aspek penting dalam kelancaran pelaksanaan magang. Hal tersebut dikarenakan teknologi yang dipelajari di kuliah dapat jauh berbeda dengan yang diterapkan perusahaan. Pemagang harus memiliki daya belajar yang tinggi untuk terus terbuka terhadap teknologi-teknologi yang digunakan perusahaan. Dengan keterbukaan tersebut, pemagang dapat lebih mudah untuk terus mempelajari teknologi baru.

4.2.4 Mendapatkan Pandangan terhadap Dinamika Industri

Keadaan lingkungan di perkuliahan dengan di dunia industri jauh berbeda. Di perkuliahan, fase kerja dalam mengerjakan proyek dilakukan secara terstruktur dan memiliki waktu yang relatif lama. Sedangkan di dunia industri, fase kerja sangat cepat untuk mengejar ketertinggalan terhadap pesaing dan memenuhi kebutuhan pengguna secepat-cepatnya. Selain itu, di dunia industri sering terjadi hal yang tidak sesuai ekspektasi yang membuat rusaknya *timeline* pengerjaan proyek. Hal tersebut membuat karyawan khususnya pemagang untuk dapat beradaptasi dengan hal yang tidak sesuai ekspektasi tersebut.

Selain fase kerja, orientasi mahasiswa atau pekerja industri berbeda. Mahasiswa dalam mengerjakan proyek biasanya berorientasi untuk menyelesaikan tugas. Mahasiswa termotivasi untuk mengerjakan proyek sebaik-baiknya untuk mendapatkan nilai yang tinggi. Sedangkan di dunia industri, pekerja mengerjakan proyek sekadar untuk mendapatkan gaji dan memenuhi kebutuhan. Pekerja melakukan pekerjaan hanya sesuai dengan apa yang telah diarahkan sebelumnya. Motivasi untuk melakukan pekerjaan sebaik-baiknya oleh pekerja dapat lebih sedikit dibandingkan mahasiswa.

Berdasarkan hal tersebut, menjadi sebuah tantangan sebagai pemagang untuk terus termotivasi dalam mengerjakan pekerjaan yang diberikan perusahaan. Selain melakukan pekerjaan sesuai arahan, pemagang harus terus memotivasi diri untuk selalu melakukan inisiatif yang dapat memberikan dampak positif bagi perusahaan.

4.2.5 Mendapatkan Wadah Aktualisasi Keilmuan

Satu di antara alasan dalam mengikuti magang adalah keinginan untuk menerapkan ilmu yang telah dipelajari selama kuliah. Suatu kepuasan tersendiri untuk memanfaatkan ilmu yang telah dipunya kepada lingkungan sekitar, khususnya di perusahaan. Pemagang merasa bersyukur karena telah merasakan bekerja sesuai dengan bidang ilmu yang dipelajari di perkuliahan. Dengan mengikuti magang, menjadi kesempatan untuk mengimplementasikan ilmu sebanyak-banyaknya kepada perusahaan. Harapannya, bentuk implementasi ilmu tersebut dapat berkembang hingga cakupan masyarakat yang lebih luas.

4.2.6 Bekerja di Bawah Tekanan

Satu di antara hal yang sering dijumpai sebagai syarat lowongan pekerjaan adalah dapat bekerja di bawah tekanan. Tidak dapat dipungkiri bahwa hal tersebut memang benar adanya.

Dalam melakukan pekerjaan, pekerja harus dapat bekerja semaksimal mungkin walaupun memiliki latar belakang masalah yang tidak diketahui pekerja lainnya.

Berdasarkan hal tersebut, menjadi sebuah tantangan bagi pemagang untuk dapat memanajemen emosi seiring berubahnya keadaan. Selain itu, pemagang juga harus pandai dalam memanajemen stres karena adanya masalah yang timbul baik di dalam pekerjaan maupun di luar pekerjaan. Manajemen pikiran juga harus dapat dilakukan dengan baik apalagi pekerjaan yang dilakukan pemagang adalah pekerjaan yang mengandalkan otak daripada otot untuk menyelesaikan pekerjaan.

4.2.7 Pengalaman Presentasi atau Berbicara di Depan Umum

Dalam melaksanakan magang, presentasi merupakan hal yang rutin dilakukan setiap minggunya. Pemagang harus dapat melaporkan pekerjaan yang dikerjakan pada seminggu ke belakang. Dengan pengalaman tersebut, pemagang menjadi lebih percaya diri dalam melakukan presentasi karena telah biasa melakukannya.

Biasanya, ketidakpercayaan diri dalam melakukan presentasi dikarenakan adanya kekhawatiran terhadap kesalahan yang mungkin timbul pada saat presentasi. Padahal, setelah melakukan beberapa presentasi, pemagang menjadi sadar bahwa pembicara maupun penyimak adalah manusia yang tidak luput dari kesalahan. Adanya komentar buruk pada saat presentasi dapat dijadikan sebagai pelajaran untuk terus memperbaiki diri dalam presentasi ke depannya. Hal yang bisa dilakukan untuk memaksimalkan presentasi adalah terus latihan dan menguasai materi yang dipresentasikan.

4.2.8 Pengalaman Belajar dari Mentor

Pada saat melaksanakan magang, pemagang mendapatkan mentor atau supervisor. Mentor memberikan pelajaran maupun arahan terkait pekerjaan. Selain terkait dengan pekerjaan, mentor juga memberikan pelajaran mengenai kemampuan non teknis seperti kultur perusahaan dan bagaimana seorang karyawan memiliki kepribadian yang baik.

Tidak hanya mengenai transfer ilmu, pemagang menjadikan mentor sebagai koneksi baru dalam berjejaring terutama dalam lingkup karier dan industri. Dengan adanya koneksi tersebut, pemagang dapat memanfaatkan koneksi untuk mendapatkan informasi yang mungkin akan dicari di kemudian hari.

Proses transfer ilmu oleh mentor kepada pemagang juga bermanfaat bagi pemagang. Hal tersebut bermanfaat untuk pemagang jika di kemudian hari diberi tugas untuk memberikan ilmu

kepada orang lain terutama transfer ilmu terhadap orang baru di perusahaan yang pemagang tempati di kemudian hari.

4.2.9 Pandangan terhadap Pengelolaan Perusahaan

Pemagang mempunyai cita-cita untuk memiliki perusahaan. Untuk mencapai cita-cita tersebut, pemagang perlu mempelajari bagaimana sebuah perusahaan mengelola semua hal yang terkait di dalamnya. Dengan mengikuti magang, pemagang mendapatkan gambaran bagaimana sebuah perusahaan terkelola.

Di Jala Tech, perusahaan merupakan rintisan bisnis teknologi yang bergerak dibidang akuakultur. Kultur kerja yang ada pada rintisan bisnis juga memiliki keunikan tersendiri dibandingkan perusahaan korporasi yang ada. Pada rintisan bisnis, hierarki jabatan bukan menjadi sesuatu yang menghalangi berjalannya komunikasi. Komunikasi dilakukan secara setara walaupun dilakukan oleh kedua orang yang memiliki jabatan yang berbeda. Hal tersebut sangat baik untuk membangun lingkungan pekerjaan yang terbuka dan cepat. Namun, yang menjadi catatan bahwa seorang pemimpin harus tetap bisa mempertahankan integritas yang ada walaupun memiliki komunikasi yang setara dengan karyawan lain.

Berdasarkan hal tersebut, pemagang menjadi sadar bahwa lingkungan dan cara komunikasi di suatu perusahaan dapat berubah seiring perubahan zaman. Hal tersebut memberikan wawasan bagi pemagang bahwa jika memiliki perusahaan nanti, pemagang harus dapat menjadikan perusahaan tersebut untuk terus beradaptasi dengan perubahan yang terjadi pada lingkungan sekitar.

Beberapa kali pemagang juga mendapatkan pertanyaan personal, tidak hanya persoalan pekerjaan. Hal tersebut untuk memastikan bahwa saya memiliki keadaan mental yang baik. Hal tersebut memberikan dampak positif bagi saya dan memberikan kesan bahwa perusahaan peduli dengan keadaan mental karyawannya. Hal tersebut memberikan pelajaran bagi pemagang bahwa jika memiliki perusahaan nanti, pastikan perusahaan tersebut peduli dengan kesehatan mental karyawannya. Perusahaan harus dapat memberikan kebutuhan karyawannya tidak hanya persoalan materi, tapi juga keadaan mental karyawannya.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pelaksanaan magang, implementasi *prop drilling* pada fitur finansial aplikasi web Jala meningkatkan efektivitas pengelolaan *state*. Beberapa manfaat terhadap adanya peningkatan efektivitas pengelolaan *state* menggunakan *prop drilling* adalah sebagai berikut:

a. Adanya kemampuan Vue Devtools dalam melacak perubahan *state*. Hal tersebut mempermudah pengembang VueJS melakukan debug pada kode *frontend* aplikasi. Oleh karena itu, jika terdapat *state* yang memiliki perilaku di luar ekspektasi, maka pengembang dapat melakukan investigasi masalah dibantu dengan Vue Devtools.
b. Penyimpanan *state* menjadi terpusat. Implementasi *prop drilling* yang membuat penyimpanan *state* menjadi terpusat komponen utama mempermudah dan mempercepat pengembang lain dalam melakukan identifikasi terdapat *state* yang digunakan pada fitur finansial. Hal tersebut dikarenakan *state* tidak disimpan di setiap komponen yang menggunakan *state*, namun *state* hanya disimpan pada komponen induk utama dan *state* dikirim melalui *props*.

5.2 Saran

Adapun saran yang dapat diberikan dalam pengembangan aplikasi web Jala di masa yang akan datang khususnya pada pengembangan *frontend* adalah sebagai berikut:

a. Perlu dilakukannya riset mendalam di kemudian hari mengenai pengelolaan *state* pada fitur lainnya yang bertujuan untuk mencari pengelolaan *state* yang lebih efektif khususnya pada penerapan jangka panjang. Hal tersebut dikarenakan teknologi pengembangan *frontend* yang dipakai pada aplikasi web Jala merupakan teknologi yang sudah usang khususnya pada versi VueJS yang digunakan. VueJS yang digunakan pada aplikasi web Jala adalah versi 2.6.6, sedangkan versi terkini (26 Desember 2021) adalah versi 3.2.20.
b. Perlu dilakukannya reviu kode pada kode yang dikerjakan oleh tim pengembang untuk menjaga kebersihan dan kualitas kode yang masuk pada aplikasi web Jala. Selain meningkatkan kualitas kode dari aplikasi web Jala, pengembang juga dapat berkembang menjadi lebih baik lagi sehingga menjadi pemrogram yang lebih berkualitas.

c. Perlu adanya dokumentasi terhadap komponen yang dibuat oleh pengembang. Hal tersebut akan bermanfaat jika pengembang lain akan mengerjakan fitur yang sama di kemudian hari dan menjadi lebih mudah mengenali fungsi dari komponen yang ditulis. Selain itu, hal tersebut bermanfaat pada saat pengembang lain ingin memakai ulang komponen yang telah dibuat sebelumnya karena telah mengetahui fungsi dari komponen tersebut.

DAFTAR PUSTAKA

Cherckesova, L., Boldyrikhin, N., Revyakina, E., Safaryan, O., & Yengibaryan, I. (2021). Development of a real-time document approval system. In E3S Web of Conferences (Vol. 273, p. 08047). EDP Sciences.

Duldulao, D. B., & Cabagnot, R. J. L. (2021). Managing *State* Using Redux with Redux Toolkit. In Practical Enterprise React (pp. 203-214). Apress, Berkeley, CA.

Freeman, A. (2018). Loosely Coupled Components. In Pro Vue. js 2 (pp. 451-477). Apress, Berkeley, CA.

Frestilia, N. (2013). Pengaruh Pemanfaatan Teknologi Informasi, Karakteristik Informasi Sistem Akuntansi Manajemen, Dan Ketidakpastian Lingkungan Terhadap Kinerja Manajerial. Jurnal akuntansi, 1(1).

Jala Tech. (2021). Jala Tech Web App. Diambil dari https://app.jala.tech

Jala Tech. (2021). Jala: Solusi Digital untuk Kesuksesan Tambak Udang Anda. Diambil dari https://jala.tech

Kankaala, M. (2019). Enhancing E-Commerce with Modern Web Technologies.

Nelson, B. (2018). State Management. In Getting to Know Vue. js (pp. 127-147). Apress, Berkeley, CA.

Pressman, R. S. (2005). Software engineering: a practitioner's approach. Palgrave macmillan.

Stack Overflow. (2021). Stack Overflow Developer Survey 2021. Diambil dari https://insights.stackoverflow.com/survey/2021#technology-most-popular-technologies

Szymanek, K., & Pańczyk, B. (2021). Comparison of web application *state* management tools. Journal of Computer Sciences Institute, 20, 183-188.

Vuejs. (2021). Event Bus, Events API. Diambil dari https://v3.vuejs.org/guide/migration/events-api.html#event-bus

Vuejs. (2021). Introduction. Diambil dari https://vuejs.org/v2/guide/index.html

Wohlgethan, E. (2018). SupportingWeb Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue. js (Doctoral dissertation, Hochschule für Angewandte Wissenschaften Hamburg).

Yang, J., & Papazoglou, M. P. (2002, May). Web component: A substrate for web service reuse and composition. In International Conference on Advanced Information Systems Engineering (pp. 21-36). Springer, Berlin, Heidelberg.

LAMPIRAN

Lampiran Sertifikat Magang

JALA

MAGANG:
010/VIII/JALA/2021

SERTIFIKAT

Albarra Naufala Erdanto

Mahasiswa Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia dengan NIM 18523158, telah menyelesaikan magang selama periode 03 Februari 2021 - 03 Agustus 2021 dengan hasil
SANGAT BAIK.
Yang bersangkutan telah memiliki kemampuan Web Development dengan VueJS dan Laravel, selain itu juga sudah sangat baik dalam melakukan tanggung jawabnya di dalam tim.

Demikian sertifikat ini dikeluarkan untuk dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 03 Agustus 2021

JALA
PT. Atnic Ekotekno Wicaksana

LIRIS MADUNINGTYAS
CHIEF EXECUTIVE OFFICER
JALA TECH

Head Office :
Jl. Kaliurang KM 16,3
Yogyakarta 55584, Indonesia
T. +62 274 2874515
F. +62 274 898269