KERJA PRAKTIK - EF234603

Pengembangan Sistem Automasi Surat RfQ dan Sistem Turnamen Pingpong menggunakan *Framework* Laravel, Blade *Templating Engine*, dan *Framework* Vue.js

PT. Max Solusindo Jaya
Jl. S. Parman Kav. 28 Lantai 19/Unit T7, Tanjung Duren Selatan, Grogol Petamburan, Jakarta Barat
Periode: 26 Juli 2024 – 26 Oktober 2024

Oleh:

Arya Widia Putra 5025201016

Pembimbing Jurusan
Rizky Januar Akbar, S.Kom., M.Eng.

Pembimbing Lapangan
Alfonso Ricky Kurnia A

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2025

[Halaman ini sengaja dikosongkan]

KERJA PRAKTIK – EF234603

Pengembangan Sistem Automasi Surat RfQ dan Sistem Turnamen Pingpong menggunakan Framework Laravel, Blade Templating Engine, dan Framework Vue.js

PT. Max Solusindo Jaya
JI. S. Parman Kav. 28 Lantai 19/Unit T7, Tanjung Duren Selatan, Grogol Petamburan, Jakarta Barat
Periode: 26 Juli 2024 – 26 Oktober 2024

Oleh:

Arya Widia Putra 5025201016

Pembimbing Jurusan
Rizky Januar Akbar, S.Kom., M.Eng.

Pembimbing Lapangan
Alfonso Ricky Kurnia A

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2025

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR POTONGAN KODE

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN
KERJA PRAKTIK

Pengembangan Sistem Automasi Surat RfQ dan Sistem Turnamen Pingpong menggunakan Framework Laravel, Blade Templating Engine, dan Framework Vue.js

Oleh:

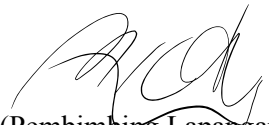
Arya Widia Putra 5025201016

Disetujui oleh Pembimbing Kerja Praktik:

1. Rizky Januar Akbar, S.Kom., M.Eng.
   NIP. 198701032014041001

(Pembimbing Departemen)

2. Alfonso Ricky Kurnia A



(Pembimbing Lapangan)

[Halaman ini sengaja dikosongkan]

Pengembangan Sistem Automasi Surat RfQ dan Sistem Turnamen Pingpong menggunakan Framework Laravel, Blade Templating Engine, dan Framework Vue.js

Nama Mahasiswa	: Arya Widia Putra
NRP	: 5025201016
Departemen	: Teknik Informatika FTEIC-ITS
Pembimbing Departemen	: Rizky Januar Akbar, S.Kom., M.Eng.
Pembimbing Lapangan	: Alfonso Ricky Kurnia A

ABSTRAK

PT Max Solusindo Jaya memiliki layanan manajemen pendataan request for Quotation, berupa automasi pencatatan penawaran RfQ ke dalam Google Sheets tracking. Namun, pembuatan surat penawaran RfQ untuk penawaran RfQ ini masih dilakukan secara manual oleh manusia. Selain layanan ini, PT. Max Solusindo Jaya juga mendapatkan permintaan sistem untuk mengelola data turnamen pingpong yang dapat membuat file daftar nilai berformat XLSX. Kegiatan magang dilakukan secara online dari 26 Juli 2024 sampai 26 Oktober 2024. Penulis merumuskan masalah dengan melihat existing system dan menggali kebutuhan sistem. Dari analisis, penulis mendapatkan bahwa kebutuhan dari Sistem Automasi Surat RfQ adalah integrasi dengan Google Sheets tracking, pengelolaan data penawaran vendor dan admin, dan pembuatan surat RfQ dengan Google Sheets API. Sedangkan, untuk Sistem Turnamen Pingpong, penulis mendapatkan kebutuhannya adalah manajemen data-data turnamen dan pembuat lembar nilai turnamen berformat .xlsx. Kedua sistem dibangun menggunakan framework Laravel. Sistem Automasi Surat RfQ menggunakan Blade Templating Engine sebagai pembangun front-end, SQLite sebagai basis data

sederhana, dan Google Workspace API untuk mengakses dan membuat Google Sheets. Di lain sisi, Sistem Turnamen Pingpong menggunakan Vue.js untuk pendukung pembangunan front-end yang lebih kompleks dan MySQL untuk penyimpanan dat di basis data. Dari hasil pengujian dengan cara membandingkan sistem yang telah dibuat dengan kebutuhan yang sudah ditentukan, penulis memperoleh hasil bahwa kedua sistem sudah berhasil diimplementasikan dengan baik. Di masa depan, penulis berharap sistem notifikasi pada Sistem Automasi Surat RfQ dapat dikembangkan menjadi lebih baik dan fitur bracketing dan proses manajemen turnamen lainnya dapat diintegrasikan ke Sistem Turnamen Pingpong.

Kata Kunci: Web Development, Laravel, Vue.js, Docker, Google Workspace API, PHPSpreadsheet

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena atas karunia-Nya yang memperbolehkan penulis menyelesaikan salah satu kewajiban sebagai mahasiswa Departemen Teknik Informatika, yaitu Kerja Praktik dengan judul: Pengembangan Sistem Automasi Surat RfQ dan Sistem Turnamen Pingpong menggunakan *Framework* Laravel, Blade *Templating Engine*, dan *Framework* Vue.js.

Penulis menyadari masih memiliki banyak kekurangan dalam pelaksanaan kerja praktik maupun penyusunan buku laporan kerja praktik. Kendati demikian, penulis berharap buku laporan ini dapat memberi manfaat bagi pembaca ke depannya.

Melalui kata pengantar ini, penulis juga ini menyampaikan terima kasih kepada orang-orang yang telah membantu penyusunan laporan kerja praktik ini, yaitu:

1. Orang tua penulis yang memberikan dukungan.
2. Bapak Rizky Januar Akbar, S.Kom., M.Eng. selaku dosen pembimbing kerja praktik.
3. Bapak Alfonso Ricky Kurnia A selaku pembimbing lapangan selama pelaksanaan kerja praktik.
4. Teman-teman, terutama Kresnanta dan Fadhil yang selalu memberikan semangat dan memberikan bantuan ketika penulis melaksanakan KP.

Taichung, 8 Januari 2025
Arya Widia Putra

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

PT. Max Solusindo Jaya menyediakan layanan bantuan manajemen data *request for quotation* (RfQ) bagi vendor penyedia barang. Sistem yang sudah ada memasukkan data yang telah diolah dari email yang dikirim klien ke dalam Google Sheets *tracking* secara otomatis. Langkah ini merupakan langkah yang cukup baik karena data menjadi lebih tersentralisasi dan tercatat dengan baik. Namun, ketika vendor ingin mengirimkan harga penawaran, admin masih membuat surat secara manual berdasarkan data yang diberikan oleh vendor dalam bentuk *file* Excel. Lalu, pembaruan status pengajuan ini juga dilakukan secara manual oleh admin ke dalam Google Sheets *tracking*. Sistem yang dapat membuat surat RfQ dan memperbarui status secara otomatis dibutuhkan untuk meningkatkan efisiensi dan juga mengurangi *human error* yang dapat terjadi saat pengerjaan manual surat RfQ.

Selain menyediakan layanan manajemen data RfQ, PT Max Solusindo Jaya menerima permintaan untuk pembuatan manajemen lomba atau turnamen pingpong. Sistem ini diharapkan dapat mengeluarkan *output* berupa daftar nilai yang berisi detail organisasi penyelenggara, detail turnamen, detail pertandingan, dan daftar peserta. Sistem juga diharapkan dapat memberikan *preview* daftar nilai saat pengguna sedang mengisi formulir pendataan peserta pertandingan. Dalam kegiatan kerja praktik ini, penulis ingin membuat sistem meningkatkan kinerja pembuatan surat *request for quotation* (RfQ) dan organisasi turnamen pingpong.

1.2. Tujuan

Kerja praktik ini memiliki tujuan sebagai berikut:

1. Mengembangkan suatu sistem yang menggabungkan *workflow* manajemen *request of quotation* yang sudah ada sebelumnya yang menggunakan Google Sheets dan pembuatan surat RfQ manual menjadi sistem informasi yang terintegrasi dan lebih terautomasi.
2. Mengembangkan suatu sistem yang dapat mengelola data-data turnamen pingpong, termasuk detail turnamen, data pertandingan, dan data peserta, dengan kemampuan tambahan berupa *preview* data peserta sebagai daftar nilai dan *export* data turnamen menjadi daftar nilai berformat XLSX.

1.3. Manfaat

Hasil yang diharapkan dari kerja praktik ini adalah sistem yang meningkatkan efisiensi kerja dari pengerjaan surat RfQ dan sistem turnamen pingpong yang terintegrasi dan dapat mempermudah pembuatan daftar nilai turnamen.

1.4. Rumusan Masalah

Rumusan masalah yang dibahas pada buku kerja praktik ini adalah

1. Bagaimana proses pembuatan aplikasi web yang mengintegrasikan Google Sheets *tracking* dan membuat surat RfQ yang terautomasi?
2. Bagaimana proses pengembangan dan implementasi aplikasi web untuk sistem turnamen pingpong dengan fitur *preview* dan *export* daftar nilai ke *file* XLSX?

1.5. Lokasi dan Waktu Kerja Praktik

Kerja praktik dilaksanakan dari tanggal 26 Juli 2024 hingga 26 Oktober 2024. Kerja praktik dilaksanakan secara *online* dengan skema *Work From Home* (WFH).

1.6. Metodologi Kerja Praktik

Metodologi yang dilakukan dalam pembuatan buku kerja praktik adalah sebagai berikut:

1.6.1. Perumusan Masalah

Sistem manajemen RfQ yang sudah ada sebelumnya masih mengandung bagian pengerjaan manual yang rawan akan *human error* sehingga dibutuhkan sistem yang menjembatani antara Google Sheets *tracking* dengan modul yang membuat surat RfQ secara otomatis. Selanjutnya, untuk sistem yang kedua, klien membutuhkan sebuah sistem turnamen pingpong yang dapat menyimpan data turnamen, pertandingan, dan peserta, serta membuat daftar nilai yang berbentuk Excel dan berisi tambahan logo panitia.

1.6.2. Studi Literatur

Setelah mendapatkan penjelasan dari keperluan dari kedua sistem, penulis menganalisis topik-topik yang perlu ditinjau. Topik-topik yang ditinjau untuk kedua sistem, antara lain kerangka kerja pengembangan web, kontainer dan kontainerisasi, dan *RESTful* API.

1.6.3. Analisis dan Perancangan Sistem

Berdasarkan tinjauan pustaka yang telah dilakukan penulis, penulis menganalisis dan merancang sistem yang baik. Kedua sistem dirancang dengan gaya arsitektur *model-view-controller*.

1.6.4. Implementasi Sistem

Dari rancangan yang dibuat, sistem diimplementasi menggunakan *tools* dan *framework* yang sudah ditentukan. Pada tahap ini, *deployment* sistem dilakukan yang akan menunjukkan aplikasi hasil dari *source code*.

1.6.5. Pengujian dan Evaluasi

Sistem yang telah dibangun perlu diuji dan dievaluasi untuk menentukan apakah sistem sudah sesuai dengan *user requirement*.

1.6.6. Kesimpulan dan Saran

Dalam kesimpulan dan saran, penulis mengambil keputusan dalam menjawab rumusan masalah dan menjabarkan pengembangan yang dapat dilakukan di masa depan.

1.7. Sistematika Laporan

1.7.1. Bab I Pendahuluan

Bab ini berisi latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

Bab ini

1.7.2. Bab II Profil Perusahaan

Bab ini berisi selayang pandang dari PT. Max Solusindo Jaya, berupa profil dan lokasi perusahaan.

1.7.3. Bab III Tinjauan Pustaka

Bab ini berisi dasar teori dari teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

1.7.4. Bab IV Analisis dan Perancangan Infrastruktur Sistem

Bab ini berisi mengenai tahap analisis sistem aplikasi dalam menyelesaikan proyek kerja praktik.

1.7.5. Bab V Implementasi Sistem

Bab ini berisi uraian tahap - tahap yang dilakukan untuk proses implementasi aplikasi.

1.7.6. Bab VI Pengujian dan Evaluasi

Bab ini berisi hasil uji coba dan evaluasi dari aplikasi yang telah dikembangkan selama pelaksanaan kerja praktik.

1.7.7. Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dan saran yang didapat dari proses pelaksanaan kerja praktik.

BAB II
PROFIL PERUSAHAAN

2.1. Profil PT. Max Solusindo Jaya

PT. Max Solusindo Jaya adalah perusahaan yang menawarkan solusi digital untuk membantu perusahaan, organisasi, atau individu mencapai tujuan mereka. Perusahaan ini menawarkan layanan seperti perancangan dan pengembangan *website*, pengembangan aplikasi perangkat bergerak, data *analytics*, pemasaran digital, *search engine optimization* (SEO), layanan komputasi awan, keamanan siber, dan solusi *e-commerce*. PT. Max Solusindo Jaya membantu klien mengidentifikasi kebutuhan mereka dan membuat sistem yang sesuai dengan kebutuhan tersebut. PT. Max Solusindo Jaya memberikan solusi yang paling efektif dan inovatif dengan perkembangan teknologi terkini.

2.2. Lokasi

PT. Max Solusindo Jaya berlokasi di Jalan S. Parman Kav. 28 Lantai 19/Unit T7, Tanjung Duren Selatan, Grogol Petamburan, Jakarta Barat.

[Halaman ini sengaja dikosongkan]

BAB III
TINJAUAN PUSTAKA

3.1. Kerangka Kerja (*Framework*) Pengembangan Web

Kerangka kerja atau *framework* adalah komponen-komponen yang dapat digunakan dalam berbagai proyek dan sistem yang dapat mempermudah pengembangan suatu aplikasi. Gagasan tentang kerangka kerja muncul dari komponen-komponen yang memiliki fungsi serupa di tiap aplikasi. Komponen-komponen ini memberikan nilai struktural pada suatu sistem, tetapi tidak memberikan nilai dalam pengembangan fitur baru atau proses bisnis. Dengan adanya kerangka kerja, pengembang aplikasi dapat fokus pada penambahan nilai bisnis dan pengembangan fitur, daripada menghabiskan waktu untuk pengembangan komponen struktural seperti *session control*, koneksi basis data, dan keamanan.

Penggunaan kerangka kerja dalam pengembangan suatu aplikasi memiliki kelebihan dan kekurangan. Penggunaan kerangka kerja mempercepat pengembangan suatu aplikasi karena komponen struktural tidak perlu lagi dibuat dari awal oleh pengembang aplikasi. Penggunaan kerangka kerja juga mempermudah pemeliharaan karena tanggung jawab pemeliharaan komponen struktural diambil oleh tim pemelihara *framework*, sementara tim pengembang aplikasi fokus pada pemeliharaan fitur aplikasi, terutama proses bisnis. Lalu, pengembang juga dapat membuat basis kode untuk fitur-fitur dengan struktur yang terstandarisasi menggunakan *command-line tools*. Kendati demikian, penggunaan kerangka kerja dapat menyebabkan masalah kinerja aplikasi karena banyaknya paket (*package*) yang di-*bootstrap* ke dalam bundel kerangka kerja. Masalah lain yang dapat muncul adalah pengembangan dan pemeliharaan kerangka kerja yang tidak memadai. Terakhir, pengembang mungkin akan menghadapi *learning curve* saat mulai menggunakan kerangka kerja.

Laravel adalah kerangka kerja yang memiliki karakteristik *full stack-based* dan *full-fledged*. Suatu kerangka kerja dianggap *full stack-based* jika pengembangan *back end* dan *front end* dapat dilakukan dalam satu kerangka kerja tersebut. Di lain sisi, Laravel dapat diklasifikasikan sebagai kerangka kerja *full-fledged* karena sebuah bundel pengembangan awal Laravel standar dilengkapi dengan banyak paket (*package*). Dengan paket-paket ini, Laravel mempermudah pengembangan aplikasi dengan memberikan pola dan gaya yang mapan. Akan tetapi, sering kali, banyak paket yang pada akhirnya tidak diperlukan dan menyebabkan proyek memiliki ukuran *file* yang besar.

Vue.js 3 adalah sebuah *view layer framework* yang digunakan untuk mengembangkan sisi front-end dari sebuah aplikasi web. Utamanya, *framework* ini adalah sebuah *declarative framework*, yang berarti pengembang yang menggunakan *framework* ini mengetikkan hasil yang diinginkan dari program pada *source code*, alih-alih menjabarkan setiap langkah untuk mencapai hasil tersebut seperti dalam *imperative framework* seperti jQuery (Yang, 2023). Meskipun demikian, Vue.js tetap menggunakan metode imperatif secara internal. Keuntungan menggunakan metode deklaratif adalah tingkat pemeliharaan yang lebih tinggi karena fokus utamanya adalah menunjukkan hasil yang mana lebih mudah dipahami daripada rantai fungsi yang digunakan pada metode imperatif. Di lain sisi, kelemahan *declarative framework* adalah kinerjanya tidak dapat melampaui *imperative framework* karena adanya tambahan *cost* untuk mencari tahu perubahan yang terjadi ketika kita ingin mengubah komponen dalam dokumen HTML.

3.2. Kontainer dan Kontainerisasi

Kontainerisasi (*containerization*) adalah teknik virtualisasi tingkat sistem operasi. Perangkat virtual yang dibuat menggunakan kontainerisasi disebut kontainer (*container*). Kontainer bekerja mirip dengan mesin virtual atau *virtual machine* (VM) tetapi lebih ringan dalam penggunaan sumber daya (Zammetti, 2022). VM

mengemulasi komputer fisik, termasuk perangkat kerasnya. VM dapat berjalan pada komputer fisik, yang disebut komputer *host*, dengan bantuan *hypervisor*. *Hypervisor* dapat berjalan di atas OS *host* (disebut *hosted hypervisor*) atau di bawah OS *host* (disebut *hardware hypervisor*), dimana *hardware hypervisor* memberikan kinerja yang lebih baik. Apa pun jenis *hypervisor* yang digunakan VM, VM tetap berat untuk dijalankan karena meniru mesin nyata. Untuk mengatasi masalah ini, kontainer berbagi sumber daya *kernel* dengan OS *host*.

Kontainer memiliki beberapa karakteristik utama. Pertama, kontainer dapat digunakan untuk mengemas aplikasi dan *runtime environment*, seperti distribusi JDK, Node, dan Linux, dalam suatu paket (Zammetti, 2022). Paket ini disimpan pada suatu *file* cetak biru yang diberi nama *image*. Lalu, kontainer berfungsi layaknya satu-satunya OS yang berjalan pada komputer. Terakhir, kontainer dapat dimulai dengan cepat sehingga lebih *easy to scale*.

Docker adalah alat yang mempermudah pengembang aplikasi untuk membangun dan mengelola *image* dan kontainer lewat *command-line* (Zammetti, 2022). Docker awam digunakan untuk membuat kontainer, bahkan dapat dikatakan sebagai alat standar industri untuk membuat kontainer (Engebreth & Sahu, 2023). Dengan kontainer Docker, pengembang dapat membuat sistem yang menyerupai web server tanpa harus membuat *virtual machine* yang benar-benar terpisah dari *host environment*. Kontainer ini juga dapat dibuat untuk menyerupai komputer dengan pengaturan yang spesifik. Dengan fitur ini, pengembang dapat menggunakan fungsi dan keuntungan dari penggunaan web server tanpa harus membuat, menginstal, dan memelihara server riil atau virtual.

3.3. *Application Programming Interface* dengan *Representational State Transfer Architecture* (*RESTful* API)

Application programming interface (API) adalah sebuah alat yang memberikan seseorang atau sistem kemampuan untuk

berinteraksi dengan perangkat lunak atau keras melalui operasi-operasi yang diperbolehkan (Dominte, 2023). API memberikan abstraksi dari *application layer* dan menyembunyikan detail rumit dalam suatu proses (Jain, 2022). API berurusan dengan *backend* dan bertindak sebagai *middle tier* dalam suatu aplikasi, menghubungkan *frontend* dan *backend*.

API biasanya digunakan dengan memanggil metode atau mengatur nilai pada *properties* yang terhubung dengan bagian kontrol sistem (Dominte, 2023). Pengguna dalam konteks API biasanya mengacu pada sistem atau pengembang aplikasi. Hal ini membedakan API dengan aplikasi pada umumnya, di mana aplikasi pada umumnya memiliki tombol atau elemen interaktif yang bertujuan mempermudah penggunaannya bagi sebagian besar orang.

API memiliki beragam jenis dan bentuk. *Push/Stream* API adalah API yang mengirimkan notifikasi secara *real-time* pada klien, *event driven*, dan sering digunakan pada sistem yang memerlukan pemrosesan *real-time* atau mengutamakan waktu . *Native* API adalah antarmuka untuk sebuah perangkat atau alat tertentu. *Software development kit* (SDK) adalah seperangkat alat yang membantu pengembang untuk membuat aplikasi. *Remote procedure call* (RPC) API memiliki ciri pemanggilan suatu metode yang terdapat di komputer lain yang dilakukan melalui jaringan tampak seperti pemanggilan suatu metode dalam aplikasi atau *scope* yang sama, yang bertujuan untuk menampilkan sistem yang terlihat monolitik. *RESTful* API adalah jenis API yang paling banyak digunakan, yang dibangun mengikuti gaya arsitektur *representational state transfer* (REST), sebuah konsep yang pertama kali dicetuskan oleh Roy Fielding.

Gaya arsitektur *representational state transfer* (REST) adalah gaya arsitektur yang awam diterapkan dalam pembuatan layanan web dan API (Ravi Kumar et al., 2024). Gaya arsitektur REST memiliki beberapa kaidah utama yang dijadikan acuan apakah suatu layanan atau API dapat dikatakan *RESTful*. Kaidah-kaidah

ini memberikan manfaat yang menguntungkan kinerja suatu aplikasi atau API.

1. Penggunaan arsitektur klien-server (*client-server architecture*): Kaidah membagi tanggung jawab suatu aplikasi ke *role* klien dan server. Klien bertanggung jawab menampilkan antarmuka pengguna yang ramah, sedangkan server bertanggung jawab menjalankan logika bisnis. Dengan demikian, sistem menjadi lebih modular, sehingga lebih mudah dipelihara, lebih fleksibel, dan lebih aman.
2. Interaksi yang *stateless* antara klien dan server (*stateless interaction*): Kaidah ini menjadikan server lebih sederhana, mudah diskalakan, dan fleksibel karena tidak perlu mengingat sesi dari suatu *client*.
3. *Cacheability*: *Cacheability* berarti kemampuan bagi klien atau sistem perantara untuk menyimpan respons untuk sementara. Dengan adanya penyimpanan sementara atau *cache*, kinerja dapat ditingkatkan dan beban pada server dapat dikurangi karena sumber daya dapat diambil dari *cache* terlebih dahulu daripada diambil dari server secara langsung.
4. Sistem yang berlapis (*layered system*): Sistem yang berlapis berarti tanggung jawab fungsi dan tugas pada suatu aplikasi dibagi kepada lapisan-lapisan (*layers*) yang berbeda. Pembagian ini meningkatkan *modularity*, menciptakan komunikasi yang dapat diprediksi dan terstandardisasi, dan mempermudah kolaborasi dalam pengerjaan sistem karena tim yang berbeda dapat mengerjakan lapisan yang berbeda secara bersamaan.
5. Antarmuka yang seragam (*uniform interface*): Kaidah ini mendukung kesederhanaan dalam pemakaian suatu layanan atau API. Penerapan kaidah ini terlihat pada identifikasi sumber daya yang seragam, metode HTTP dan kode status yang terstandardisasi, dan pesan yang deskriptif. Keuntungan dari penerapan kaidah ini adalah mengurangi *learning curve*, membuat lebih perilaku API lebih mudah diprediksi, dan mempercepat pengembangan aplikasi.

Google API adalah *application programming interface* yang memberikan pengembang kemampuan untuk mengakses aplikasi-aplikasi keluaran Google menggunakan pemrograman (Google for Developers, n.d.). Salah satu bagian dari aplikasi Google adalah sejumlah aplikasi yang termasuk dalam Google Workspace, seperti Google Drive, Docs, Sheets, dan yang lainnya (Chun, 2023). Aplikasi-aplikasi ini didukung dengan *RESTful* API di belakangnya. Dengan demikian, pengembang aplikasi dapat mengintegrasikan alur kerja pada aplikasi yang dibuat dengan aplikasi-aplikasi Google Workspace.

BAB IV
ANALISIS DAN PERANCANGAN INFRASTRUKTUR SISTEM

4.1. Analisis Sistem

Bab ini berisi penjelasan tentang tahap-tahap yang dilakukan untuk melakukan implementasi aplikasi Sistem Automasi Surat RfQ dan Sistem Turnamen Pingpong. Hal ini dijelaskan dalam dua bagian untuk tiap sistem, yaitu definisi umum dan analisis kebutuhan.

4.1.1. Definisi Umum Aplikasi Sistem Automasi Surat RfQ

Aplikasi Sistem Automasi Surat RfQ adalah aplikasi web yang memiliki fungsi untuk mempermudah proses pemantauan *request for quotation* (RfQ) dan pembuatan surat RfQ bagi vendor. Aplikasi ini digunakan untuk memantau *request for quotation* (RfQ) dari konsumen untuk vendor dengan menampilkan RfQ yang sudah masuk ke dalam dokumen Google Sheet *tracking*. Aplikasi Sistem Automasi Surat RfQ juga melakukan automasi dalam pembuatan surat RfQ untuk beberapa nomor RfQ sekaligus yang diolah dari dokumen format .xlsx sederhana.

4.1.2. Analisis Kebutuhan Aplikasi Sistem Automasi Surat RfQ

Setelah mendengarkan kebutuhan bisnis dari Sistem Automasi Surat RfQ, penulis menyimpulkan bahwa sistem memiliki beberapa kebutuhan sebagai berikut:

1. Vendor dapat melihat daftar RfQ yang terintegrasi dengan *file* Google Sheets *tracking* RfQ. Sistem yang sebelumnya digunakan adalah *file* Google Sheets yang terintegrasi dengan *email*, di mana *email* masuk tentang RfQ akan tercatat otomatis ke dalam *file* Google Sheets.
2. Vendor dapat mengunggah *file* daftar penawaran RfQ untuk konsumen dalam bentuk *file* .xlsx. *File* daftar penawaran ini berisi nomor RfQ, produk yang ditawarkan, dan harga

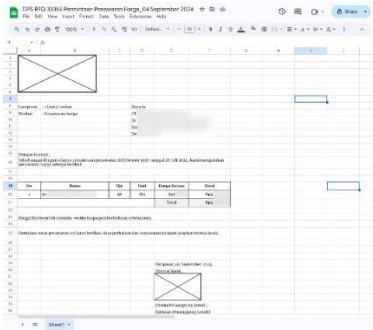
penawaran. Contoh *file* daftar penawaran dapat dilihat pada **Gambar 4.1**.

LIST NEGOISASI HARGA BARANG PT.
TANGGAL: 30 AGUSTUS 2024

NO	RFQ CODE	TANGGAL PENAWARAN	NAMA BARANG	QTY	VOLUME	HARGA (Rp)	JUMLAH HARGA (Rp)
1	RFQ 11111	20 Juli 2024	M	1.00	Unit	2.81	2,8
2	RFQ 22222	21 Juli 2024	Je	40.00	Pcs	7	3,1
3	RFQ 22223	21 Juli 2024	Je	160.00	Pcs	7	12,5
4	RFQ 33333	27 Juli 2024	Si	10.00	Pcs	50	5,0
5	RFQ 25252	27 Juli 2024	H	10.00	Meter	12	1,2
6	RFQ 25252	27 Juli 2024	Ba	10.00	Batang	7	7
7	RFQ 36363	28 Juli 2024	Ba	10.00	Batang	32	3,2
8	RFQ 33355	28 Juli 2024	La	20.00	Pcs	17	3,5
9	RFQ 28888	21 Juli 2024	K	1.00	Pcs	2.01	2,0
10	RFQ 28888	21 Juli 2024	K	2.00	Pcs	2.01	4,0
11	RFQ 26766	28 Juli 2024	Ri	5.00	Roll	35	1,7

Gambar 4.1. Tangkapan layar contoh *file* daftar penawaran

3. Admin dapat memonitor daftar penawaran RfQ yang diunggah oleh vendor. Lebih jelas lagi, admin dapat melakukan sub tugas berikut:
   a. Admin dapat memeriksa data apakah sudah dalam format yang tepat untuk mencegah *error* pada pembuatan surat RfQ dengan Google Sheets.
   b. Admin dapat melakukan *approval* pada data RfQ yang sudah sesuai format dan meneruskan perintah agar program memroses *file* daftar penawaran RfQ menjadi surat RfQ dengan format Google Sheets. Contoh hasil surat RfQ dengan Google Sheets dapat dilihat pada **Gambar 4.2**.



Gambar 4.2. Tangkapan layar contoh surat RfQ dalam format Google Sheets

4.1.3. Definisi Umum Aplikasi Sistem Turnamen Pingpong

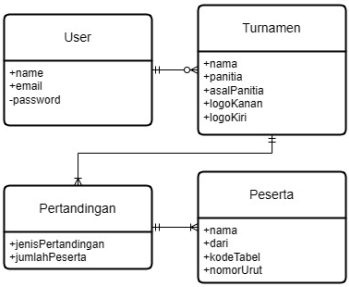
Aplikasi Sistem Turnamen Pingpong adalah aplikasi yang berfungsi untuk mengelola data turnamen pingpong atau tenis meja, yaitu data turnamen, data pertandingan (bagian lebih kecil dari turnamen), dan peserta pertandingan. Aplikasi Sistem Turnamen Pingpong memiliki fitur ekspor data turnamen, pertandingan, dan peserta ke dalam bentuk lembar penilaian dengan format .xlsx.

4.1.4. Analisis Kebutuhan Aplikasi Sistem Turnamen Pingpong

Kebutuhan dari Sistem Turnamen Pingpong adalah sebagai berikut.

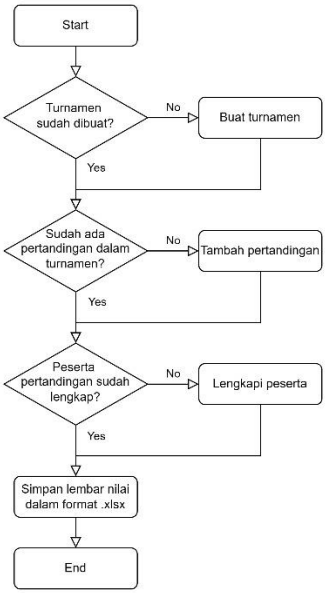
1. Menyimpan data-data turnamen, yang dapat direpresentasikan dengan entitas-entitas berikut:
    a. Turnamen, yang memiliki atribut nama turnamen, nama panitia, asal panitia, gambar logo kanan, dan gambar logo kiri.
    b. Pertandingan, yang memiliki atribut jenis pertandingan dan jumlah peserta.
    c. Peserta, yang memiliki atribut nama peserta, asal peserta (dari), kode tabel, dan nomor urut peserta
2. Membuat lembar nilai dari data turnamen, yang hanya bisa dibuat jika peserta sudah lengkap dimasukkan ke masing-masing pertandingan.

Hubungan antara entitas sistem ini dapat dilihat pada **Gambar 4.3**. Pengguna atau *user* akan dapat membuat nol sampai banyak turnamen. Turnamen akan memiliki beberapa pertandingan dan pertandingan hanya dimiliki satu turnamen saja. Pertandingan akan memiliki peserta sebanyak atribut “jumlahPeserta” dan setiap peserta milik satu pertandingan saja.



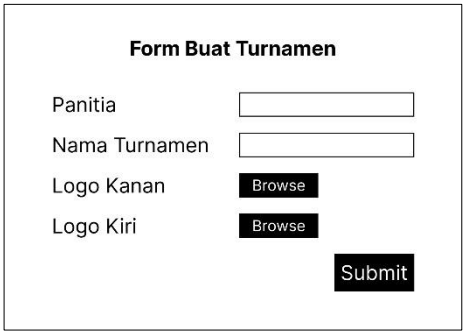
Gambar 4.3. *Entity relationship diagram* aplikasi Sistem Turnamen Pingpong

Secara umum, alur kerja dari Sistem Turnamen Pingpong dijelaskan dalam *flowchart* pada **Gambar 4.4**.



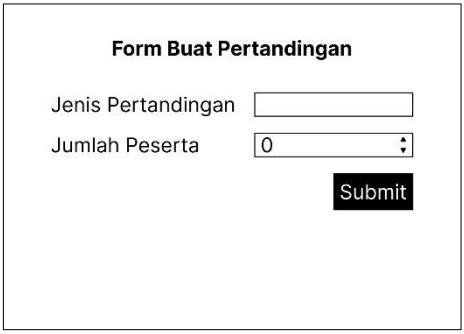
Gambar 4.4. *Flowchart* penggunaan aplikasi Sistem Turnamen Pingpong

Penulis juga diberikan gambaran umum atau wireframe dari halaman-halaman fungsi Sistem Turnamen Pingpong. **Gambar 4.5** memberikan gambaran umum untuk form pembuatan turnamen pingpong.



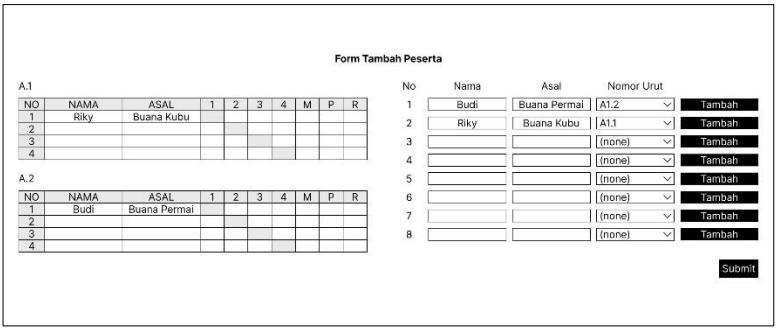
Gambar 4.5. Rancangan website (wireframe) dari aplikasi Sistem Turnamen Pingpong, halaman Form Buat Turnamen

Selanjutnya, **Gambar 4.6** memberikan gambaran umum untuk form pembuatan pertandingan pingpong.



Gambar 4.6. Rancangan website (wireframe) dari aplikasi Sistem Turnamen Pingpong, halaman *Form* Buat Pertandingan

Terakhir, **Gambar 4.7** memberikan gambaran umum untuk *form* pendaftaran peserta pertandingan. *User* mengisi *input* pada bagian kanan halaman, yaitu berupa nama, asal, dan nomor urut beserta tabel yang diinginkan. Ketika *user* menekan tombol tambah di samping *form* data peserta, data peserta akan ditampilkan pada bagian kiri halaman, layaknya *preview* dari lembar nilai yang dapat disimpan nantinya.



Gambar 4.7. Rancangan website (wireframe) dari aplikasi Sistem Turnamen Pingpong, halaman Form Tambah Peserta

4.2. Perancangan Infrastruktur Sistem

4.2.1. Sistem Automasi Surat RfQ

Bagian ini berisi penjelasan mengenai teknologi pengembangan aplikasi web yang digunakan pada Sistem Automasi Surat RfQ beserta alur pengerjaan sistem tersebut.

4.2.2.1 Teknologi yang Digunakan

Pengerjaan Sistem Automasi Surat RfQ menggunakan beberapa teknologi pengembangan aplikasi web. *Tech stack* yang digunakan dalam pengembangan sistem ini, antara lain:

1. PHP, merupakan bahasa pemrograman yang digunakan untuk pengembangan sistem secara umum, baik *back end* maupun *front end*.
2. Laravel, merupakan kerangka kerja pengembangan *full-stack web application* berbasis bahasa PHP. Aplikasi ini terutama

memanfaatkan fitur *background jobs* dan *queue* yang disediakan oleh Laravel.

3. Blade, *templating engine* yang membantu pembuatan *front end* dengan bahasa PHP.
4. SQLite, sebagai embedded database engine yang menjalankan basis data untuk menyimpan data-data Sistem Automasi Surat RfQ. SQLite biasanya menjadi sistem basis data bawaan Laravel, jika pengembang tidak mengubah konfigurasi basis data.
5. Google Workspace API, terutama Google Drive API dan Google Sheets API, untuk mencatat perubahan pada status RfQ dan membuat *file* surat RfQ untuk klien dari vendor.

Selain *tech stack* yang telah disebutkan, *development tools* juga digunakan untuk mendukung pengembangan aplikasi. *Development tools* yang digunakan adalah sebagai berikut:

1. Visual Studio Code, merupakan aplikasi *code editor* yang juga berfungsi untuk mengelola *file* kode dan integrasi Git.
2. XAMPP, untuk menjalankan *local server* yang melayani aplikasi yang dibuat.
3. Git, merupakan alat yang melacak perubahan pada *code base* dan menyimpan riwayat perubahan.
4. GitHub, merupakan repositori yang dapat digunakan dengan Git untuk menyimpan *code base*.

4.2.2.2 Alur Pengerjaan Sistem

Sistem Automasi Surat RfQ dikerjakan secara bertahap per fungsi. Fungsi pertama yang dibuat adalah menampilkan daftar RfQ yang diambil dari Google Sheets *tracking* RfQ. Fungsi selanjutnya yang diimplementasikan adalah pembuatan surat RfQ untuk konsumen dalam bentuk *file* Google Sheets. Lalu, fungsi *approval* oleh admin untuk memastikan data sudah dalam format yang tepat diimplementasikan. Terakhir, fungsi pembuatan surat RfQ sebagai *background process* dilakukan untuk memungkinkan

proses berjalan lebih lama daripada waktu respons yang diizinkan dan meningkatkan kinerja situs web secara keseluruhan.

4.2.2. Sistem Turnamen Pingpong

Bagian ini berisi penjelasan mengenai teknologi yang digunakan dan alur pengerjaan dari Sistem Turnamen Pingpong.

4.2.2.1 Teknologi yang Digunakan

Beberapa teknologi pengembangan aplikasi web digunakan dalam pengerjaan Sistem Turnamen Pingpong. *Tech stack* yang digunakan, yaitu:

1. PHP, sebagai bahasa pemrograman untuk pengembangan aplikasi bagian secara umum.
2. Laravel, sebagai kerangka kerja pengembangan *full-stack web application*.
3. Vue.js, sebagai kerangka kerja pendukung pengembangan aplikasi bagian *front end*.
4. MySQL, sebagai sistem basis data *relational* untuk penyimpanan data-data turnamen pingpong.

Selain *tech stack*, beberapa *development tools* juga digunakan, yaitu:

1. Visual Studio Code, sebagai aplikasi *code editor*, pengelolaan *file*, dan pembantu integrasi dengan Git dan GitHub.
2. Docker dan Docker Desktop, sebagai aplikasi pengelolaan *container* untuk melakukan testing dan mempersiapkan *deployment* pada perangkat lain.
3. Git, sebagai alat yang merekam perubahan pada *code* yang dikerjakan.
4. GitHub, sebagai repositori penyimpanan *code* dari aplikasi.

4.2.2.2 Alur Pengerjaan Sistem

Pengerjaan Sistem Turnamen Pingpong dilakukan secara bertahap. Tahap pertama adalah implementasi dasar aplikasi, yaitu implementasi fungsi tambah turnamen, tambah pertandingan, dan tambah peserta. Tahap kedua adalah implementasi fungsi simpan lembar nilai ke *file* .xlsx. Tahap terakhir adalah persiapan

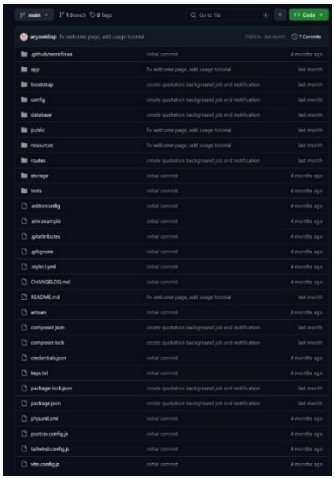
deployment menggunakan Docker. Di antara tahapan, pembimbing lapangan memberikan revisi untuk fitur yang sudah dibuat.

BAB V
IMPLEMENTASI SISTEM

Bab ini berisi pembahas terkait dua sistem yang dibuat oleh penulis. Sistem yang pertama dibahas adalah Sistem Automasi Surat RfQ. Lalu, sistem yang dibahas selanjutnya adalah Sistem Turnamen Pingpong.

5.1. Implementasi Sistem Automasi Surat RfQ

Sistem Automasi Surat RFQ adalah sebuah aplikasi web yang dirancang untuk membantu *tracking* data RfQ dan pembuatan surat RfQ secara otomatis oleh vendor. Sistem Automasi Surat RFQ dibangun mengikuti pola desain *model-view-controller* (MVC). Sistem ini menggunakan Google Workspace API untuk membuat surat RfQ yang berformat Google Sheets. Lalu, sistem ini juga menggunakan *background jobs* dan fitur notifikasi untuk meng-*handle* proses pembuatan surat yang memakan waktu lebih dari *response timeout* pada umumnya.



Gambar 5.1. Direktori repositori Sistem Automasi Surat RfQ

5.1.1. Model

Sistem Automasi Surat RfQ memiliki empat *model*, yaitu User, Penawaran, PenawaranChange, dan Notification. Secara garis besar, hubungan antar model adalah sebagai berikut: User memiliki beberapa Penawaran, Penawaran memiliki beberapa PenawaranChange, dan User menerima beberapa Notification. Setiap *model* memiliki *properties* dan *methods* masing-masing.

Model User adalah representasi dari pengguna. Atribut yang dimiliki oleh User adalah *name* (nama), *email*, dan *password*. *Model* User memiliki *method* penawarans() yang menerapkan HasMany untuk merepresentasikan hubungannya dengan penawaran secara terprogram.

```
<?php
namespace App\Models;
// uses directives
class User extends Authenticatable
{
    use HasFactory, Notifiable, HasRoles;
    protected $fillable = [
        'name', 'email', 'password',
    ];
    protected $hidden = [
        'password',
        'remember_token',
    ];
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }
    public function penawarans(): HasMany
    {
        return $this->hasMany(Penawaran::class);
    }
}
```

Potongan Kode 5.1. Model User (User.php)

Model Penawaran adalah representasi program dari daftar penawaran harga yang diunggah oleh suatu vendor. Model ini memiliki atribut *title* (judul), *filePenawaran*, dan *penawaranStatus*. Atribut *filePenawaran*, sesuai namanya, berfungsi untuk menyimpan *path* menuju *file* dengan daftar harga penawaran. Atribut *penawaranStatus* menunjukkan apakah suatu penawaran sedang menunggu konfirmasi dari admin, membutuhkan revisi, atau sudah dikonfirmasi oleh admin. *Method* user() menerapkan *class* BelongsTo berfungsi untuk menunjukkan hubungan Penawaran dengan User, yaitu penawaran dimiliki oleh suatu *user*. *Method* penawaranChanges() menerapkan *class* HasMany menunjukkan bahwa suatu Penawaran memiliki beberapa PenawaranChange.

```php
<?php
namespace App\Models;
// use directives
class Penawaran extends Model
{
    use HasFactory;
    protected $fillable = [
        'title',
        'filePenawaran',
        'penawaranStatus'
    ];
    public function user(): BelongsTo
    {
        return $this->belongsTo(User::class);
    }
    public function penawaranChanges(): HasMany
    {
        return $this->hasMany(PenawaranChange::class);
    }
}
```

Potongan Kode 5.2. Model Penawaran (Penawaran.php)

Model PenawaranChange merepresentasikan perubahan pada suatu penawaran. Oleh karena suatu penawaran dapat direvisi, sistem membutuhkan histori perubahan pada penawaran. Model ini

memiliki atribut *filePenawaran* dan *penawaranStatus*. Namun, migrasi untuk *model PenawaranChange* juga memiliki *timestamp* untuk mencatat kapan penawaran diubah. Atribut *filePenawaran* menyimpan *path* ke *file* yang dirujuk pada suatu perubahan. Sebagai contoh, vendor dapat mengunggah ulang *file* ke suatu penawaran jika admin meminta revisi pada suatu harga item. *File* lama dan baru akan tercatat sebagai suatu obyek PenawaranChange. Atribut *penawaranStatus* menyimpan status Penawaran saat diubah. *Method* penawaran() pada *model* PenawaranChange menunjukkan hubungan *model* ini ke *model* Penawaran, yaitu suatu PenawaranChange dimiliki oleh suatu Penawaran.

```
<?php
namespace App\Models;
// use directives
class PenawaranChange extends Model
{
    use HasFactory;
    protected $fillable = [
        'filePenawaran',
        'penawaranStatus'
    ];
    public function penawaran(): BelongsTo
    {
        return $this->belongsTo(Penawaran::class);
    }
}
```

Potongan Kode 5.3. Model PenawaranChange (PenawaranChange.php)

Model terakhir pada sistem ini adalah *model* Notification yang merepresentasikan notifikasi yang dikirim ke pengguna (0). Kode model Notification tidak menunjukkan banyak hal dan tidak menunjukkan atribut yang dimiliki oleh model ini karena semua atributnya diisi otomatis oleh sistem. Jika melihat migrasi untuk model Notification pada 0, suatu objek Notification akan memiliki atribut *type*, *notifiable*, *data*, dan *read_at*. Atribut *type* menunjukkan tipe dari notifikasi agar *front end* dapat memproses sesuai keperluan. Atribut *notifiable* menunjukkan target dari

notifikasi, yang mana dalam kasus ini adalah User atau pengguna. Atribut *notifiable* juga mencatat pengguna mana yang akan dikirimkan notifikasi. Atribut *data* menunjukkan data yang ingin dikirim dalam notifikasi ke pengguna (Contoh data dapat dilihat pada 0). Terakhir, atribut *read_at* menunjukkan waktu suatu notifikasi dibaca.

```
<?php
namespace App\Models;
// uses directives
class Notification extends Model
{
    use HasFactory;
}
```

Potongan Kode 5.4. Model Notification (Notification.php)

```
<?php
// use directives
return new class extends Migration
{
    public function up(): void
    {
        Schema::create('notifications', function (Blueprint
$table) {
            $table->uuid('id')->primary();
            $table->string('type');
            $table->morphs('notifiable');
            $table->text('data');
            $table->timestamp('read_at')->nullable();
            $table->timestamps();
        });
    }
    public function down(): void
    {
        Schema::dropIfExists('notifications');
    }
};
```

Potongan Kode 5.5. Migrasi untuk Notification (2024_08_28_063650_create_notifications_table.php)

```
// Pesan jika berhasil
{
    "noRfq": "26760",
    "duration": 12.59118103981018,
    "message": "Successfully created the quotation for RFQ
26760. Time elapsed: 12.59118103981"
}
// Pesan jika gagal
{
    "noRfq": "26760",
    "duration": 41.76051092147827,
    "message": "Failed to create the quotation for RFQ 26760.
Error message: "
}
```

Potongan Kode 5.6. Data untuk menunjukkan pembuatan surat RfQ gagal atau berhasil

5.1.2. *View*

Sistem Automasi Surat RfQ memiliki enam view utama, yaitu ‘penawarans.index’, ‘penawarans.edit’, dan ‘penawarans.changes.index’ untuk *role* vendor, kemudian ‘penawarans.index’, ‘penawarans.show’, dan ‘notification.index’ untuk *role* admin. Implementasi untuk *view* ‘penawarans.index’ dapat dilihat pada 0 sampai 0. View ini menampilkan daftar penawaran dan formulir untuk menambah penawaran baru.

```
<x-app-layout>
    <!-- Layout baris pertama -->
    <div>
        <!-- Daftar Penawaran Diterima -->
        <div>...</div>
        <!-- Daftar Penawaran Revisi -->
        <div>...</div>
    </div>
    <!-- Layout baris kedua -->
    <div>
        <!-- Daftar Semua Penawaran -->
        <div>...</div>
    </div>
    <!-- Layout baris ketiga | Formulir Penawaran Baru -->
```

```
    <div>...</div>
    <!-- Loading modal -->
    <div id="loader" hidden>...</div>
    <script>...</script>
</x-app-layout>
```

Potongan Kode 5.7. Struktur *view* 'penawarans.index' (vendor)

Daftar penawaran yang ditampilkan pada *view* ini dibagi menjadi tiga dengan tampilan yang kurang lebih sama. 0 menunjukkan kode untuk menampilkan semua penawaran. Untuk penawaran yang dikonfirmasi, kolom histori pada tabel dihilangkan. Di lain sisi, kolom histori diganti menjadi kolom tombol untuk melakukan *update* untuk penawaran yang perlu revisi (0).

```
<div>
    <h2>Semua Penawaran</h2>
    <div>
        <table>
            <thead>
                <tr>
                    <th scope="col">No</th>
                    <th scope="col">Judul Penawaran</th>
                    <th scope="col">Tanggal diupload</th>
                    <th scope="col">File</th>
                    <th scope="col">Histori</th>
                </tr>
            </thead>
            <tbody>
                @foreach ($penawarans as $penawaran)
                    <tr>
                        <th scope="row">
                            {{ $penawaran->id }}
                        </th>
                        <td>{{ $penawaran->title }}</td>
                        <td>{{ $penawaran->created_at }}</td>
                        <td>{{ $penawaran->filePenawaran }}</td>
                        <th>
                            <a href="{{ route(
                                'penawarans.history.index',
```

```
                                $penawaran
                            ) }}">
                                <x-secondary-button>
                                    Lihat
                                </x-secondary-button>
                            </a>
                        </th>
                    </tr>
                @endforeach
            </tbody>
        </table>
        <div class="mt-3">
            {{ $penawarans->appends([
                'dikonfirmasi' =>
                    $penawaransDikonfirmasi->currentPage(),
                'revisi' => $penawaransRevisi->currentPage()
            ])->links() }}
        </div>
    </div>
</div>
```

Potongan Kode 5.8. Komponen yang menampilkan daftar semua penawaran

```
<th>
    <a href="{{ route('penawarans.edit', $penawaran) }}">
        <x-secondary-button>Update</x-secondary-button>
    </a>
</th>
```

Potongan Kode 5.9. Kolom terakhir pada tabel daftar penawaran revisi

Setelah daftar penawaran, view ‘penawaran.index’ berisi formulir *upload* penawaran baru (0). Saat pengguna memasukkan *file* ke kolom *input upload file* (sebelum formulir di-*submit*), data dari file tersebut akan muncul pada halaman sebagai *preview*.

```
<div>
    <h2>Upload Penawaran Baru</h2>
    <form
        method="POST" action="{{ route('penawarans.store') }}"
```

```
        enctype="multipart/form-data" id="fileForm">
        @csrf
        <!-- Upload file penawaran -->
        <div>
            <label for="filePenawaran">File Penawaran</label>
            <div>
                <input
                    type="file"
                    id="filePenawaran"
                    name="filePenawaran"/>
            </div>
        </div>
        @if ($errors->any())
            <div>
                <ul>
                    @foreach ($errors->all() as $error)
                        <li>{{ $error }}</li>
                    @endforeach
                </ul>
            </div>
        @endif
        <div>
            <h2>File preview</h2>
            <div id="dataPreview">
                <p id="dataPlaceholder">No file to show.</p>
            </div>
        </div>
        <div id="titleFormGroup">
            <label for="title">Judul Penawaran</label>
            <input type="text" id="title" name="title" />
        </div>
        <x-input-error :messages="$errors->get('message')"/>
        <x-primary-button>
            {{ __('Create request') }}
        </x-primary-button>
    </form>
</div>
```

Potongan Kode 5.10. Formulir untuk mengunggah penawaran baru pada *view* ‘penawarans.index’ (vendor)

Proses *preview* data dari *file* yang di-*upload* diatur menggunakan *script* (berbahasa JavaScript) yang tercantum pada 0 sampai 0. 0 menunjukkan pengambilan elemen dari halaman, seperti *input file*, bagian *data preview*, dan *data placeholder*, penambahan *event listener*, serta proses pembentukan elemen dengan appendChild(). Bagian memproses data dari *file* per baris akan dijelaskan pada paragraf berikutnya.

```
// Tampilkan preview dari file yang diunggah
const input = document.getElementById('filePenawaran');
const dataPreview = document.getElementById('dataPreview');
const dataPlaceholder = document.getElementById(
    'dataPlaceholder'
);
let titleValue;
input.addEventListener('change', function() {
    readXlsxFile(input.files[0]).then(function(data) {
        dataPlaceholder.setAttribute('class', 'hidden');
        const table = document.createElement('table');
        table.setAttribute('class', 'mt-2');
        const thead = document.createElement('thead');
        const tbody = document.createElement('tbody');
        // Memproses data dari file per baris
        data.map((row, index) => {
            ...
        });
        table.appendChild(thead);
        table.appendChild(tbody);
        dataPreview.appendChild(table);
        // Mengeset input judul secara otomatis
        const title = document.getElementById('title');
        title.setAttribute('value', titleValue);
        const titleFormGroup = document.getElementById(
            'titleFormGroup'
        );
        titleFormGroup.classList.remove("hidden");
    });
});
```

Potongan Kode 5.11. *Script* untuk menampilkan *preview* data dari *file* Excel *view* 'penawarans.index' (vendor)

Data dari *file* dibagi menjadi judul penawaran, detail penawaran, *header* tabel, dan daftar harga penawaran. 0 menunjukkan logika pemrosesan bagian judul penawaran, detail penawaran, dan *header* tabel. Bagian memproses daftar harga penawaran akan dijelaskan pada paragraf berikutnya.

```
if (index == 0) { // Judul penawaran
    const judulPenawaran = document.createElement('h2');
    judulPenawaran.setAttribute('class',
        'text-center text-xl font-semibold')
    titleValue = String(row[0] == null ? 'No title' : row[0]);
    judulPenawaran.appendChild(
        document.createTextNode(
            String(row[0] == null ? 'No title' : row[0])
    ));
    dataPreview.appendChild(judulPenawaran);
} else if (index == 1) { // Detail penawaran
    const detailPenawaran = document.createElement('p');
    detailPenawaran.setAttribute('class',
        'text-center text-lg text-slate-700')
    detailPenawaran.appendChild(
        document.createTextNode(
            String(row[0] == null ? 'No details' : row[0])
    ));
    dataPreview.appendChild(detailPenawaran);
} else if (index == 2) {
    // Pembatas antara detail file dan daftar penawaran
    // Lewati
} else if (index == 3) { // Header
    const trow = document.createElement('tr');
    trow.setAttribute('class',
        'h-9 bg-violet-700 text-violet-50');
    row.map((cell) => {
        const tcell = document.createElement('th');
        tcell.setAttribute('class', 'border-2 px-3')
        tcell.appendChild(document.createTextNode(
            String(cell == null ? '' : cell)
        ));
        trow.appendChild(tcell);
    });
    thead.appendChild(trow);
```

```
} else { // Daftar harga penawaran
    ...
}
```

Potongan Kode 5.12. *Script* untuk logika pemrosesan data pada Excel per baris pada *view* 'penawarans.index' (vendor)

0 menunjukkan detail pemrosesan bagian daftar harga penawaran dari *file* penawaran. Tiap baris dibaca per kolom. Beberapa kolom diberi perlakukan khusus, yaitu kolom indeks 0 yang adalah kolom nomor, kolom indeks 4 yang merupakan banyak kuantitas, dan kolom indeks 6 dan 7 yang merupakan harga per item dan harga dikali kuantitas,

```
const trow = document.createElement('tr');
trow.setAttribute('class', 'h-9 odd:bg-white even:bg-gray-50 border-b');
row.map((cell, index) => {
    const tcell = document.createElement('td');
    tcell.setAttribute('class', 'border-2 px-3');
    if (index == 0) {
        const th = document.createElement('th');
        th.setAttribute('class', 'border-2 px-3');
        th.appendChild(document.createTextNode(String(cell == null ?
            '' : cell)));
        trow.appendChild(th);
    } else if (index == 6 || index == 7) {
        const wrapper = document.createElement('div');
        wrapper.setAttribute('class',
            'text-right flex flex-row justify-between');
        const rupiah = document.createElement('span');
        rupiah.appendChild(document.createTextNode('Rp'));
        rupiah.setAttribute('class', 'font-bold mr-2')
        wrapper.appendChild(rupiah);
        const amount = document.createElement('span');
        const number = new Intl.NumberFormat('id-ID', {}).format(
            cell == null ? 0 : parseInt(cell), );
        amount.appendChild(document.createTextNode(number));
        wrapper.appendChild(amount);
        tcell.appendChild(wrapper);
        trow.appendChild(tcell);
    } else if (index == 4) {
```

```
        tcell.setAttribute('class', 'border-2 px-3 text-right');
        tcell.appendChild(document.createTextNode(String(cell ==
            null ? '' : cell)));
        trow.appendChild(tcell);
    } else {
        tcell.appendChild(document.createTextNode(String(cell ==
            null ? '' : cell)));
        trow.appendChild(tcell);
    }
});
tbody.appendChild(trow);
```

Potongan Kode 5.13. *Script* menampilkan *preview* bagian daftar harga penawaran pada 'penawarans.index' (vendor)

Selain *script* untuk menunjukkan *preview file* penawaran, *view* 'penawarans.index' juga memiliki *script* untuk menampilkan *loading* modal saat formulir di-*submit*. *Script* tersebut dapat dilihat pada 0.

```
// Tampilkan modal ketika formulir dikirim
const fileForm = document.getElementById('fileForm');
fileForm.addEventListener("submit", () => {
    console.log(`Loading...`);
    document.getElementById('loader').removeAttribute('hidden');
})
```

Potongan Kode 5.14. *Script view* 'penawarans.index' (vendor) untuk memunculkan *loading modal*

View 'penawarans.edit' adalah *view* yang menunjukkan formulir untuk memperbarui *suatu* penawaran. *View* ini berisi formulir, bagian *preview*, dan *script* yang mirip dengan *view* 'penawarans.index'. Perbedaannya terletak pada anotasi @method('patch') dan *route submit* dari formulir.

```
<x-app-layout>
    <x-slot name="header">
        <h2>{{ __('Update penawaran') }}</h2>
    </x-slot>
    <div>
        <div>
```

```
            <!-- Formulir upload file penawaran -->
            <!-- Mirip dengan pada penawarans.index -->
            <!-- Perbedaan: -->
            <!-- action="{{ route('penawarans.update',
                            $penawaran) }}" -->
            <!-- @method('patch') -->
        </div>
    </div>
    <div><!-- Bagian preview seperti penawarans.index --></div>
    <!-- Modal loader -->
    <!-- Script sama dengan pada penawarans.index -->
</x-app-layout>
```

Potongan Kode 5.15. View 'penawarans.edit' (vendor)

View terakhir untuk *role* vendor adalah 'penawarans.changes.index' (0). View ini menampilkan histori perubahan pada suatu penawaran. Detail dari suatu perubahan ditampilkan berupa judul penawaran, waktu *update*, status, dan *directory file*. Pengguna juga dapat mengunduh *file* penawaran dengan tombol unduh penawaran.

```
<x-app-layout>
    <div>
        <div>
            <h2>Riwayat perubahan</h2>
            <div>
                <ol>
                    @foreach ($penawaranChanges as
                    $penawaranChange)
                    <li>
                        <span><!-- Ikon bullet point --></span>
                        <h3>
                            {{ $penawaranChange->penawaran
                            ->title }}
                            @if ($loop->index == 0)
                            <span>Latest</span>
                            @endif
                        </h3>
                        <time>Updated on {{ $penawaranChange
                        ->created_at }}</time>
```

```
                        <p>{{ $penawaranChange
                        ->penawaranStatus }}</p>
                        <p>{{ $penawaranChange
                        ->filePenawaran }}</p>
                        <a href="{{ route(
                        'penawarans.history.download',
                        ['changeId' => $penawaranChange->id]
                        ) }}"><!-- Ikon unduh SVG -->
                            Download File</a>
                    </li>
                    @endforeach
                </ol>
            </div>
        </div>
    </div>
</x-app-layout>
```

Potongan Kode 5.16. *View* 'penawarans.changes.index' (vendor)

View untuk *role* admin yang pertama adalah 'admin.penawarans.index'. *View* ini berfungsi untuk menampilkan semua penawaran yang diajukan oleh vendor beserta statusnya. 0 menunjukkan struktur garis besar dari *view* 'admin.penawarans.index'. Implementasi tabel daftar penawaran yang diajukan dapat dilihat pada 0.

Selain itu, *view* ini juga berisi bagian untuk mengunggah penawaran yang gagal. Bentuknya serupa dengan 0, tetapi judulnya diganti menjadi Reupload Penawaran Gagal. *Script* yang digunakan pada *view* ini juga serupa dengan *script* pada 0 sampai 0.

```
<x-admin-layout>
    <x-slot name="header">
        <h2>{{ __('Ajuan Penawaran') }}</h2>
    </x-slot>
    <div>
        <h2>{{ __('Daftar Penawaran Terbaru') }}</h2>
        <!-- Tabel penawaran ajuan (0) -->
    </div>
    <!-- Formulir unggah penawaran
```

```
     Hampir sama dengan penawarans.index,
     tetapi judul section diganti menjadi
     Reupload Penawaran Gagal
     -->
    <!-- Loading modal -->
</x-admin-layout>
```

Potongan Kode 5.17. View ‘admin.penawarans.index’

```
<div id="penawarans">
    <table>
        <thead>
            <tr>
                <th scope="col">No</th>
                <th scope="col">Judul Penawaran</th>
                <th scope="col">Diupload oleh</th>
                <th scope="col">Tanggal diupload</th>
                <th scope="col">File</th>
                <th scope="col">Status</th>
                <th scope="col">Check</th>
            </tr>
        </thead>
        <tbody>
            @foreach ($penawarans as $penawaran)
                <tr>
                    <th scope="row">{{ $penawaran->id }}</th>
                    <td>{{ $penawaran->title }}</td>
                    <td>{{ $penawaran->user_id }}</td>
                    <td>{{ $penawaran->created_at }}</td>
                    <td>{{ $penawaran->filePenawaran }}</td>
                    <td>{{ $penawaran->penawaranStatus }}</td>
                    <td>
                        <a href="{{
                        route('admin.penawarans.show',
                        $penawaran)
                        }}">{{ __('Review and Approve') }}
                        </a>
                    </td>
                </tr>
            @endforeach
        </tbody>
```

```
    </table>
    <div class="mt-3">
        {{ $penawarans->links() }}
    </div>
</div>
```

Potongan Kode 5.18. Tabel daftar penawaran pada *view* 'admin.penawarans.index'

View 'admin.penawarans.show' (0) menampilkan detail dari suatu penawaran agar admin bisa me-*review* penawaran sebelum dikonfirmasi dan dibuat surat RfQ-nya (atau diminta revisi). *View* ini menampilkan detail, seperti judul, pengguna yang mengunggah data, dan waktu data diunggah. Lalu, data dari *file* Excel juga di-*preview* (0). Setelah me-*review* data penawaran, admin dapat memilih untuk mengonfirmasi atau meminta revisi dengan menekan tombol di bawah tabel.

```
<x-admin-layout>
    <x-slot name="header">
        <h2>
            {{ __('Detail penawaran') }}
        </h2>
    </x-slot>
    <div>
        <h2>File preview</h2>
        <div id="dataPreview">
            <h2>{{ $penawaran->title }}</h2>
            <p>{{ __('Uploaded by: ') }}
                {{ $penawaran->user->name }}</p>
            <p>{{ __('Upload date: ') }}
                {{ $penawaran->created_at }}</p>
            <table>
                <thead>
                    <tr>
                        @foreach ($file[3] as $header)
                            @if ($loop->index < 8)
                            <th>{{ $header }}</th>
                            @endif
                        @endforeach
                    </tr>
```

```
                </thead>
                <tbody>
                    <!-- Tampilkan baris data penawaran -->
                </tbody>
            </table>
        </div>
        <div>
            <form method="post"
            action="{{ route(
            'admin.penawarans.update', $penawaran
            ) }}">
                @csrf
                @method('patch')
                <x-primary-button
                    name="penawaranStatus" value="Dikonfirmasi"
                >{{ __('Terima') }}</x-primary-button>
                <x-primary-button
                    name="penawaranStatus" value="Revisi"
                >{{ __('Minta revisi') }}</x-primary-button>
                <x-secondary-button
                    name="backButton"
                >{{ __('Kembali') }}</x-secondary-button>
            </form>
        </div>
    </div>
</x-admin-layout>
```

Potongan Kode 5.19. *View* 'admin.penawarans.show'

```
@foreach ($file as $row)
    @if ($loop->index > 3)
    <tr>
        @foreach ($row as $cell)
            @if ($loop->index < 8)
                @if ($loop->index == 0)
                <th scope="col">{{ $cell }}</td>
                @else
                    @if ($loop->index === 6|$loop->index === 7)
                    <td>
                        <div>
                            <span>Rp</span>
```

```
                            <span>
                                {{ Number::format(
                                    $cell,
                                    locale: 'id') }}
                            </span>
                        </div>
                    </td>
                    @else
                    <td>{{ $cell }}</td>
                    @endif
                @endif
            @endif
        @endforeach
    </tr>
    @endif
@endforeach
```

Potongan Kode 5.20. Perulangan untuk menampilkan baris data penawaran pada *view* 'admin.penawarans.show'

View 'admin.notification.index' menampilkan semua notifikasi yang diterima oleh admin. Saat ini, ada dua tipe notifikasi, yaitu Penawaran.CreateQuotationSheets.success dan Penawaran.CreateQuotationSheets.error. Notifikasi dengan tipe berbeda akan muncul dengan tampilan yang sedikit berbeda. Notifikasi Penawaran.CreateQuotationSheets.success akan ditampilkan dengan *icon success* atau *info* dan Penawaran.CreateQuotationSheets.error akan muncul dengan *icon error*.

```
<x-admin-layout>
    <x-slot name="header">
        <h2>{{ __('Semua notifikasi') }}</h2>
    </x-slot>
    <div>
        <div>{{ $notifications->links() }}</div>
        <div>
            <div id="notification-drawer">
                @inject('carbon', 'Carbon\Carbon')
                @foreach ($notifications as $notification)
                    @php
```

```
                    $createdAt = $carbon::createFromFormat(
                        'Y-m-d H:i:s',
                        $notification->created_at, 'UTC'
                    );
                    $createdAt->setTimezone('Asia/Jakarta');
                    $notification->data = json_decode(
                    $notification->data,true);
                @endphp
                <div>
                    @if (str_contains(
                        (string) $notification->type,
                        'success'
                    ))
                        <div>
                        <!-- Success icon vector image -->
                        </div>
                        <div>
                            <div>
                                {{ $notification->type }}
                            </div>
                            <h3>
                                Quotation Sheets untuk RFQ
                                {{ $notification->data[
                                'noRfq'] }}
                                berhasil dibuat.
                            </h3>
                            <p>{{ $createdAt ?? '-' }}</p>
                        </div>
                    @elseif (str_contains(
                        (string) $notification->type,
                        'error'
                    ))
                        <div>
                        <!-- Failed icon vector image -->
                        </div>
                        <div>
                            <div>
                                {{ $notification->type }}
                            </div>
                            <h3>
                                Quotation Sheets untuk RFQ
```

```
                                    {{ $notification->data[
                                    'noRfq'] }}
                                    gagal dibuat.
                                </h3>
                                <p>{{ $createdAt ?? '-' }}</p>
                            </div>
                        @endif
                    </div>
                @endforeach
            </div>
        </div>
    </div>
</x-admin-layout>
```

Potongan Kode 5.21. View ‘admin.notification.index’

Di luar *view* yang sudah dijelaskan, *component* dari *view* yang penting untuk dibahas adalah *component* ‘admin.navigation’. *Component* ini ditambah dengan *button* baru untuk mengakses notifikasi yang masuk yang ditampilkan berupa *dropdown*. Implementasi dari bagian notifikasi ini dapat dilihat pada 0. Untuk membuat aplikasi dapat menampilkan notifikasi yang pernah diterima sebelumnya dan aplikasi yang diterima secara *real time*, dilakukan injeksi data dari objek PHP ke objek JavaScript. Hal ini dilakukan karena data notifikasi yang diambil dari basis data akan diterima sebagai objek bahasa PHP, sedangkan data notifikasi secara *real time* akan diterima sebagai objek Javascript.

```
<x-slot name="content">
    @php
        $user = Auth::user();
        $userId = Auth::user()->id;
    @endphp
    <script>
        // Injeksi variable userId ke JavaScript
        let userId = {{ $userId }};
        const notifications = [];
    </script>
    @if ($user->unreadNotifications->count() > 0)
        @foreach ($user->unreadNotifications as
```

```
        $notification)
            @php
                static $count = 0;
                // abaikan loop setelah lima notifikasi
                if ($count >= 5) { continue; }
                $count++;
                $noRfq = $notification->data['noRfq'];
                $duration = $notification->data['duration'];
                $dateFinished = $notification->created_at ?? 0;
                $id = $notification->id;
                // Injeksi data notifikasi ke JavaScript
                // diformat sesuai dengan data notifikasi
                // real time
                $dataString = "{id: '$id',
                noRfq: '$noRfq', duration: Number($duration),
                type: '$notification->type',
                dateFinished: '$dateFinished'}";
                echo "<script>
                    notifications.unshift($dataString);
                </script>";
            @endphp
        @endforeach
        <div id="notification-drawer"></div>
    @else
        <div id="notification-drawer">
            <div id="notification-drawer-empty-placeholder">
                Belum ada notifikasi yang belum dibaca.
            </div>
        </div>
    @endif
    <div id="open-notification">
        <a href="{{ route('admin.notifications.index') }}">
            {{ __('Lihat semua notifikasi') }}
        </a>
    </div>
</x-slot>
```

Potongan Kode 5.22. Bagian notifikasi pada *component* 'admin.notification'

0 merupakan bagian dari *script* yang digunakan pada *component* 'admin.notification'. *Script* ini berisi logika untuk

menerima notifikasi secara *real time* dengan window.Echo.private, *rendering* tiap notifikasi dari *array* notifications[] dengan fungsi bantuan renderNotification(), dan melakukan *toggle* muncul atau tidak terhadap indikator merah pada tombol notifikasi untuk menunjukkan ada notifikasi baru atau tidak dengan menggunakan fungsi activateNotificationBadge(). Selain itu, ada fungsi readNotification() yang dipanggil saat tombol notifikasi ditekan untuk menghilangkan indikator merah. Fungsi renderNotification() tidak dicantumkan implementasinya di potongan kode, tetapi fungsinya secara garis besar adalah membuat elemen HTML yang berisi data dari notifikasi. Fungsi activateNotificationBadge() juga tidak ditampilkan, tetapi fungsinya adalah untuk melakukan *toggle* pada visibilitas elemen indikator notifikasi.

```
let notificationIsRead = false;
window.addEventListener("load", (event) => {
    window.Echo.private('App.Models.User.' + userId)
        .notification((notification) => {
            console.log(notification.type);
            notifications.unshift(notification);
            if (notifications.length > 5) {
                notifications.pop();
                renderNotification(notification, true);
            } else {
                renderNotification(notification, false);
            }
            if (notifications.length > 0) {
                activateNotificationBadge(true);
            }
            notificationIsRead = false;
        });
    notifications.forEach(notification => {
        renderNotification(notification, false);
    });
    if (notifications.length > 0) {
        activateNotificationBadge(true);
    } else {
        activateNotificationBadge(false);
    }
```

```
    const readNotification = document.getElementById(
        'read-notification');
});
function readNotification() {
    notificationIsRead = true;
    activateNotificationBadge(false);
}
```

Potongan Kode 5.23. Fungsi untuk menerima notifikasi secara *real time*

5.1.3. *Controller*

Sistem Automasi Surat RfQ memiliki 4 *controller* utama untuk proses bisnis. *Controller-controller* tersebut adalah PenawaranController, PenawaranChangeController, FileDownloadController, dan NotificationController. *Controller-controller* ini menyambungkan *front end* dan *back end*, juga melakukan proses di *background*.

Controller PenawaranController berfungsi untuk mengatur proses yang menggunakan model Penawaran. *Controller* ini memiliki 5 *methods*, yaitu index(), show(), update(), store(), dan getLocalSpreadsheetData(). *Method* index() berfungsi untuk mengarahkan pengguna ke *view* daftar penawaran dengan data penawaran. View daftar penawaran memiliki sedikit perbedaan untuk *role* admin dan *role* vendor, sehingga data yang dikirimkan dari *controller* sedikit berbeda. Untuk vendor, data penawaran dipecah menjadi 3 bagian, yaitu semua penawaran, penawaran yang perlu direvisi, dan penawaran yang dikonfirmasi admin.

```
<?php
namespace App\Http\Controllers;
use App\Http\Controllers\Controller;
use App\Models\Penawaran;
use App\Services\PenawaranService;
use PhpOffice\PhpSpreadsheet\IOFactory;
// use directives lainnya...
class PenawaranControllerLaporan extends Controller
{
    ...
```

```
}
```

Potongan Kode 5.24. PenawaranController

```
public function index(Request $request)
{
    if ($request->user()->hasRole('admin')) {
        $user = Auth::user();
        $userId = Auth::user()->id;

        return view('admin.penawarans.index')->with(
            [
                'penawarans' => Penawaran::with('user')
                ->latest()
                ->paginate(5),
            ]
        );
    }

    if ($request->user()->hasRole('vendor')) {
        return view('penawarans.index')->with(
            [
                'penawarans' => Penawaran::with('user')
                    ->where('user_id', $request->user()->id)
                    ->latest()
                    ->paginate(2, ['*'], 'semua'),
                'penawaransRevisi' => Penawaran::with('user')
                    ->where('user_id', $request->user()->id)
                    ->where('penawaranStatus', 'Revisi')
                    ->latest()
                    ->paginate(2, ['*'], 'revisi'),
                'penawaransDikonfirmasi'
                => Penawaran::with('user')
                    ->where('user_id', $request->user()->id)
                    ->where('penawaranStatus', 'Dikonfirmasi')
                    ->latest()
                    ->paginate(2, ['*'], 'dikonfirmasi'),
            ]
        );
    }
}
```

Potongan Kode 5.25. *Method* index() pada PenawaranController

Method show() pada PenawaranController mengatur halaman detail dari penawaran yang diajukan oleh vendor. *Method* ini mengarahkan pengguna yang memiliki otoritas (dalam hal ini, *role* admin) menuju *view* detail penawaran dengan data penawaran dan daftar harga penawaran. *Method* show() dapat dilihat pada 0.

```
public function show(Penawaran $penawaran): View
{
    Gate::authorize('view', $penawaran);

    return view('admin.penawarans.show')->with(
        [
            'penawaran' => $penawaran,
            'file' => $this->getLocalSpreadsheetData(
                Storage::disk('local')
                ->path($penawaran->filePenawaran),
                'Xlsx'
            ),
        ]
    );
}
```

Potongan Kode 5.26. *Method* show() pada PenawaranController

Method update() pada PenawaranController berfungsi untuk memperbarui data penawaran. *Method* ini dapat dibagi menjadi dua bagian, yaitu logika untuk *role* vendor dan logika untuk *role* admin. Logika untuk *role* vendor yang ditunjukkan pada 0 menunjukkan bahwa sistem akan memvalidasi *input* dari formulir edit penawaran dari sisi vendor. Lalu, *file* disimpan ke *storage disk*, perubahan dimasukkan ke basis data, dan perubahan dicatat ke histori.

```
public function update(Request $request, Penawaran $penawaran,
PenawaranService $penawaranService)
{er
    // Simpan pesan kembalian fungsi dan services.
    $messages = [];
    ...
    if ($request->user()->hasRole('vendor')) {
```

```
        $messages = ["Default message."];
        // Logging file yg diupload, user, dan waktu upload.
        $request->validate([
            'filePenawaran' => [
                'required',
                File::types(['xlsx'])
                    // File::image()
                    ->min(1)
                    ->max(50),
            ],
        ]);
        if ($request->hasFile('filePenawaran')) {
            $filePenawaran = $request->file('filePenawaran');
            // Simpan ke storage disk
            $filePenawaranName =
            str()->uuid() . '.' . $filePenawaran->extension();
            $filePenawaran = Storage::disk()->putFileAs(
                "filePenawaran/{$request->user()->id}",
                $filePenawaran, $filePenawaranName
            );
            // Simpan perubahan ke basis data
            $penawaran->penawaranStatus =
            'Updated | Menunggu konfirmasi';
            $penawaran->filePenawaran = $filePenawaran ??= null;
            $penawaran->save();
            // Catat perubahan ke histori
            $penawaran->penawaranChanges()->create([
                'filePenawaran' => $filePenawaran,
                'penawaranStatus' =>
                'Updated | Menunggu konfirmasi'
            ]);
        } else {
            $messages = ["File not found"];
        }
        return redirect(route('penawarans.index'))
        ->with('messages', $messages);
    }
}
```

Potongan Kode 5.27. Logika untuk *role* vendor pada *method* update()

Role admin dapat mengubah status penawaran menjadi 'Dikonfirmasi' atau 'Revisi', tercermin pada 0. Jika status diubah menjadi 'Dikonfirmasi', fungsi update() akan mengambil daftar harga penawaran dari *file* penawaran dan membuat surat RfQ-nya melalui *method* createQuotationSheetsWithTemplateLoop() dari *class* PenawaranService. Terlepas dari jenis status, data penawaran di basis data akan diperbarui sesuai *input* admin.

```
public function update(Request $request, Penawaran $penawaran,
PenawaranService $penawaranService)
{
    // Collect messages returned from functions and services.
    $messages = [];
    ...
    if ($request->user()->hasRole('admin')) {
        $request->validate([
            'penawaranStatus' => ['required', 'string'],
        ]);
        // Update status sesuai input admin
        $penawaran->penawaranStatus = $request
        ->input('penawaranStatus');
        $penawaran->save();
        $messages[] = "Update status penawaran berhasil.";
        // File retrieval and input to Google Sheet
        if ($request
        ->input('penawaranStatus') === 'Dikonfirmasi') {
            $filePenawaranName = Storage::disk('local')
            ->path($penawaran->filePenawaran);
            $data = $this->getLocalSpreadsheetData(
                $filePenawaranName, 'Xlsx'
            );
            /**
             * @var array $rawData: Processing the data from
uploaded file. Skip title and headers.
             */
            $rawData = array_slice($data, 4, count($data));
            // Data for invoice
            $invoiceData = collect($rawData)
                // Group data by RFQ Code
                ->groupBy([1, 2])
```

```
                ->toArray();
            $index = 0;
            $driveService = new Drive($penawaranService
            ->createClient());
            $penawaranService
            ->createQuotationSheetsWithTemplateLoop(
                $invoiceData, Auth::user()->ide
            );
            return redirect(route('admin.penawarans.index'))
            ->with('messages', $messages);
        }
        return redirect(route('admin.penawarans.index'))
        ->with('messages', $messages);
    }
}
```

Potongan Kode 5.28. Logika untuk *role* admin pada *method* update()

Method selanjutnya pada PenawaranController adalah *method* store() (0). Logika pada *method* ini dapat dibagi untuk *role* vendor dan *role* admin. Logika untuk *role* vendor (0) dimulai dengan validasi data penawaran yang di-*input* oleh vendor. Lalu, *file* daftar penawaran disimpan ke *storage disk* dan data penawaran disimpan ke basis data. Terakhir, *input* data ini disimpan sebagai histori dengan model PenawaranChange.

```
public function store(Request $request): RedirectResponse
{
    $messages = ["Default message."];
....// Logika role vendor
    // Logika role admin
    return redirect(route('login'));
}
```

Potongan Kode 5.29. Struktur *method* store() secara garis besar

```
if ($request->user()->hasRole('vendor')) {
    $request->validate([
        'filePenawaran' => [
            'required',
            File::types(['xlsx'])
```

```
                ->min(1)->max(50),
        ],
        'title' => ['required', 'string', 'max:255']
    ]);
    if ($request->hasFile('filePenawaran')) {
        $filePenawaran = $request->file('filePenawaran');
        // Simpan ke storage disk
        $filePenawaranName = str()->uuid() . '.' .
$filePenawaran->extension();
        $filePenawaran = Storage::disk()->putFileAs(
            "filePenawaran/{$request->user()->id}",
            $filePenawaran, $filePenawaranName
        );
    } else {
        $messages = ["File not found"];
    }
    $title = $request->input('title');
    // Catat unggahan ke basis data
    $penawaran = $request->user()->penawarans()->create([
        'title' => $title,
        'filePenawaran' => $filePenawaran ??= null,
        'penawaranStatus' => 'Menunggu konfirmasi'
        // Menunggu konfirmasi | Revisi | Dikonfirmasi
    ]);
    // Tambah perubahan ke histori
    $penawaran->penawaranChanges()->create([
        'filePenawaran' => $filePenawaran,
        'penawaranStatus' => 'Menunggu konfirmasi'
    ]);
    return redirect(route('penawarans.index'))->with('messages',
$messages);
}
```

Potongan Kode 5.30. Logika untuk *role* vendor pada *method* store()

Tujuan utama dari proses penyimpanan data penawaran oleh admin adalah untuk mengunggah ulang data penawaran yang gagal dibuat. Oleh karena itu, selain menjalankan proses seperti logika untuk *role* vendor, *method* ini juga menjalankan logika pembuatan surat RfQ dengan Google Sheet. Selain itu, status

penawaran langsung dimasukkan menjadi ‘Reupload | Dikonfirmasi’. Logika untuk *role* admin dapat dilihat pada 0.

```
if ($request->user()->hasRole('admin')) {
    $request->validate([
        'filePenawaran' => [
            'required', File::types(['xlsx'])->min(1)->max(50),
        ],
        'title' => ['required', 'string', 'max:255']
    ]);
    if ($request->hasFile('filePenawaran')) {
        $filePenawaran = $request->file('filePenawaran');
        // Simpan ke storage disk
        $filePenawaranName = str()
        ->uuid() . '.' . $filePenawaran->extension();
        $filePenawaran = Storage::disk()->putFileAs(
            "filePenawaran/{$request->user()->id}",
            $filePenawaran, $filePenawaranName
        );
    } else {
        $messages = ["File not found"];
        return redirect(route('admin.penawarans.index'));
    }
    $title = $request->input('title');
    // Catat unggahan ke basis data
    $penawaran = $request->user()->penawarans()->create([
        'title' => $title,
        'filePenawaran' => $filePenawaran ??= null,
        'penawaranStatus' => 'Reupload | Dikonfirmasi'
    ]);
    // Tambah perubahan ke histori
    $penawaran->penawaranChanges()->create([
        'filePenawaran' => $filePenawaran,
        'penawaranStatus' => 'Reupload | Dikonfirmasi'
    ]);

    // Logika pembuatan surat RfQ
    $userId = $request->user()->id;
    $filePenawaranName = Storage::disk('local')->path(
        "filePenawaran/{$userId}/$filePenawaranName"
    );
```

```
    $data = $this->getLocalSpreadsheetData(
    $filePenawaranName, 'Xlsx');
    // Ambil data dari file unggahan. Abaikan judul dan header.
    $rawData = array_slice($data, 4, count($data));
    // Kumpulkan data berdasarkan kode RfQ
    $invoiceData = collect($rawData)
        ->groupBy([1, 2])
        ->toArray();
    // Buat surat RfQ di background
    $penawaranService = new PenawaranService;
    $penawaranService->createQuotationSheetsWithTemplateLoop(
        $invoiceData, Auth::user()->ide
    );
    return redirect(route('admin.penawarans.index'))
    ->with('messages', $messages);
}
```

Potongan Kode 5.31. Logika untuk *role* admin pada *method* store()

PenawaranController juga memiliki *method* pembantu getLocalSpreadsheetData(). Fungsi dari *method* ini adalah untuk membungkus kode untuk membuka *file spreadsheet* dengan IOFactory dari modul PhpSpreadsheet. Adanya *method* ini mengurangi redundansi pembukaan *spreadsheet* yang ada pada perangkat lokal. *Method* ini akan mengembalikan *array* yang berisi daftar penawaran harga.

```
// Muat data spreadsheet pada perangkat (lokal) ke dalam array
private function getLocalSpreadsheetData(string $filePath,
string $inputFileType): array
{
    return IOFactory::createReader($inputFileType)
        ->setReadDataOnly(true)
        ->load($filePath)
        ->getActiveSheet()
        ->toArray();
}
```

Potongan Kode 5.32. *Method* getLocalSpreadsheetData() dalam PenawaranController

Sebelum lanjut ke controller selanjutnya, penulis akan membahas PenawaranService. Class PenawaranService ini memberikan fungsi-fungsi atau metode-metode bantuan untuk PenawaranController. Tujuannya adalah untuk memfokuskan PenawaranController pada pengelolaan data dari *back end* atau *front end* dan *routing*, sementara PenawaranService mengurus logika bisnis yang lebih mendalam.

PenawaranService memiliki 11 *methods*: 3 *methods* utama dan 8 *methods "wrapper"*. Penulis menganggap 8 *methods* ini sebagai *methods* "wrapper" karena fungsi utamanya adalah untuk menyederhanakan kode pada metode yang menggunakannya sehingga logikanya lebih mudah untuk dibaca.

Method utama yang pertama adalah createClient(). Fungsi *method* ini adalah untuk membuat dan melakukan konfigurasi Google API Client. Implementasi dari *method* ini dapat dilihat pada 0.

```
public function createClient(string $idPenawaran = ""): Client
//dipake di own dan queue
{
    // URL redirect setelah otorisasi
    $redirectUrl = "https://redirectmeto.com/http://
    googlesheet.test/admin/penawarans/{$idPenawaran}";

    // Buat dan konfigurasi Google API Client
    $client = new Client();
    $client->setAuthConfig(base_path('credentials.json'));
    $client->setRedirectUri($redirectUrl);
    $client->addScope([
        'https://www.googleapis.com/auth/spreadsheets',
        'https://www.googleapis.com/auth/docs',
        'https://www.googleapis.com/auth/drive.file',
    ]);
    $httpClient = new GuzzleHttpClient([
        'connect_timeout' =>
env('GUZZLE_HTTP_CLIENT_CONNECT_TIMEOUT'),
        'timeout' => env('GUZZLE_HTTP_CLIENT_TIMEOUT'),
    ]);
    $client->setHttpClient($httpClient);
```

```
    return $client;
}
```

Potongan Kode 5.33. *Method* createClient() dalam PenawaranService

Method utama yang ketiga adalah updateTrackingSheets() yang tercantum pada 0. Fungsi *method* ini adalah untuk mencatat perubahan penawaran pada Google Sheets *tracker*. Perubahan dilakukan berdasarkan nomor RfQ dari harga penawaran.

```
public function updateTrackingSheets(array $rawData) {
    $messages = [];
    // Ambil data yang perlu
    $sum = collect($rawData)
        ->map(function (?array $row) {
            return array_merge(
                array_slice($row, 0, 3),array_slice($row, 7, 1),
            );
        })
        // Grup data berdasarkan RFQ Code,
        // lalu jumlahkan total harga
        ->groupBy([1, 2])
        ->map(function (Collection $dateGroup) {
            return $dateGroup->map(
                function (Collection $group) {
                    return $group->sum(3);
            });
        })->toArray();
    $dataToBeUpdated = [];
    foreach ($sum as $rfqCode => $dateAndTotalPrices) {
        if ($dateAndTotalPrices[
        array_key_first($dateAndTotalPrices)]) {
            $dataToBeUpdated[$rfqCode] = [
                'tanggal' =>
                    array_key_first($dateAndTotalPrices),
                'harga' => $dateAndTotalPrices[
                    array_key_first($dateAndTotalPrices)]
            ];
        } else { continue;}
    }
    // Ambil data dari Google Sheets tracker
```

```
    // untuk tahu baris mana yang diperbarui
    $sheetCollection = collect($this->getSheetData())
        ->map(function (?array $row, ?int $index) {
            $pattern = "/RFQ \d+/i";
            preg_match($pattern, $row[1], $matches);
            $code = $matches[0] ?? 'temp' . $index;
            return array(
                $code => [
                    'rowPosition' => $row['rowPosition'],
                    'rowLength' => count($row)
                ]
            );
        })->toArray();
    $codeAndPosition = [];
    foreach ($sheetCollection as $items) {
        if ($items[array_key_first($items)]) {
            $codeAndPosition[array_key_first($items)] =
            $items[array_key_first($items)]['rowPosition'];
        }
    }
    // Constraint: kolomnya harus berurutan
    $kolomNomorPenawaran = 'F';
    $kolomTanggalPenawaran = 'G';
    $kolomNominalPenawaran = 'H';
    $updateCells = [];
    foreach ($dataToBeUpdated as $rfqCode => $dataToUpdate) {
        if (array_key_exists($rfqCode, $codeAndPosition)) {
            // Update the cell in Google Sheet.
            $nomorBaris = $codeAndPosition[$rfqCode];
            $updateCells["{$kolomNomorPenawaran}{$nomorBaris}
            :{$kolomNominalPenawaran}{$nomorBaris}"] = [
                "DPS {$rfqCode}",
                $dataToUpdate['tanggal'],
                intval($dataToUpdate['harga']),
            ];
        }
    }
    $client = $this->createClient();
    $service = new Sheets($client);
    foreach ($updateCells as $cellRange => $updateCell) {
        $messages[] = "{$updateCell[0]} => {$this
```

```
        ->updatePenawaran($cellRange, $updateCell, $service)}";
    }
    return $messages;
}
```

Potongan Kode 5.34. *Method* updateTrackingSheets() dalam PenawaranService

Method ketiga adalah createQuotationSheetsWith Template() (0). Fungsi utama *method* ini adalah untuk mengunggah Google Sheets *template* ke Google Drive yang sudah ditentukan, mengganti data *placeholder* dengan data yang sebenarnya, dan memasukkan data harga penawaran. Hasil dari *method* ini adalah surat RfQ dalam bentuk Google Sheets untuk suatu nomor RfQ.

```
public function createQuotationSheetsWithTemplate( // referred
in queue job
    $quotationItems, $noRfq, $tanggalRfq, $totalHarga,
    $tanggalSekarang, $templateFilePath, $driveService,
    $sheetsService
) {
    $messages = [];
    // Step 1: Upload template to drive with RFQ as its name
    $file = $this->uploadSheetsTemplate($driveService,
        $noRfq, $tanggalSekarang, $templateFilePath, );
    // Step 2: Prepare the spreadsheet, get the sheet id
    $spreadsheetId = $file->id;
    $sheetId = $this->getSheetId($sheetsService,
        $spreadsheetId);
    // Step 3: Prepare batch request -> Replace and Add rows
    // 3.1 Replace requests
    $replaceVariables = [$noRfq, $tanggalRfq, "Rp$totalHarga",
        $tanggalSekarang,];
    $findVariables = ['$noRfq', '$tanggalRfq', '$totalHarga',
        '$tanggalSekarang',];
    $findReplaceRequests = $this->createReplaceRequest(
        $replaceVariables, $findVariables);
    // 3.2 Insert dimension requests
    $tableHeight = 19 + sizeof($quotationItems) + 1;
    $insertDimensionRequest = $this
```

```
        ->createInsertDimensionRequest($sheetId, $tableHeight);
    $batchUpdateRequests = [...$findReplaceRequests,
        $insertDimensionRequest,];
    // Replace variables and insert new rows to accommodate the
data
    $messages[] = $this->replaceAndInsertDimensionToSheets(
        $batchUpdateRequests, $sheetsService, $spreadsheetId
    );
    // Prepare data for the invoice
    $tabel = new ValueRange([
        "range" => "Sheet1!" . "A20:F" . ($tableHeight - 1),
        "majorDimension" => "ROWS",
        "values" => array_map(
            function ($invoiceItem) {
                static $index = 1;
                // $invoiceItem = array_splice($invoiceItem, 3,
5);
                $hargaPerUnit = 'Rp' . Number::format(
                    $invoiceItem[6], locale: 'id');
                $totalPerItem = 'Rp' . Number::format(
                    $invoiceItem[7], locale: 'id');
                return array_merge(
                    [$index++],
                    array_splice($invoiceItem, 3, 3),
                    [$hargaPerUnit], [$totalPerItem]
                );
            },
            $quotationItems
        ),
    ]);
    // Insert invoice items
    try {
        $params = ['valueInputOption' => 'USER_ENTERED'];
        //executing the request
        $result = $sheetsService->spreadsheets_values
        ->update(
            $spreadsheetId, $tabel->range, $tabel, $params
        );
        if ($result) {
            $messages[] = "Success message";
        } else {
```

```
            $messages[] = "Failed message";
        }
    } catch (Exception $e) {
        throw $e;
    }
    return $messages;
}
```

Potongan Kode 5.35. *Method* createQuotationSheets WithTemplate() dalam PenawaranService

Method wrapper tidak akan ditampilkan sebagai kode. Namun, berikut adalah penjelasan singkat dari methods tersebut:

1. createQuotationSheetsWithTemplateLoop(): Membuat perulangan yang memanggil background job CreateQuotationSheetJob untuk tiap nomor RfQ pada penawaran.
2. updatePenawaran(): memperbarui data penawaran (misalnya status) pada Google Sheets *tracker* (per item).
3. getSheetData(): mengambil data dari Google Sheets *tracker* untuk mencari baris data dengan nomor RfQ penawaran yang akan diperbarui.
4. uploadSheetsTemplate(): menambahkan Google Sheets *template* ke Google Drive surat RfQ.
5. getSheetId(): mendapatkan *identifier* dari *spreadsheet* pertama dari Google Sheets *template*.
6. createReplaceRequest(): membuat Sheets_Request dengan argumen 'findReplace' untuk mencari *placeholder* dalam *template* dan diganti dengan nilai yang sebenarnya.
7. createInsertDimensionRequest(): membuat Sheets_Request dengan argumen 'range' untuk menambahkan baris pada tabel dalam *template*.
8. replaceAndInsertDimensionToSheets(): membuat Batch UpdateSpreadsheetRequest dari 'findReplace' dan 'range' *request* yang telah disebutkan dan mengirimnya ke Google Sheets API.

Controller PenawaranChangeController mengatur proses yang menggunakan model PenawaranChange. *Controller* ini

memiliki satu *method*, yaitu index(). *Method* index() berfungsi untuk menunjukkan *view* histori perubahan pada penawaran. *Controller* ini mengirimkan data penawaran sesuai dengan *id* yang diinginkan dan perubahan-perubahan yang terjadi pada penawaran tersebut, yang direpresentasikan dengan PenawaranChange.

```
<?php
namespace App\Http\Controllers;
// use directives
class PenawaranChangeControllerLaporan extends Controller
{
    public function index(Request $request, string $idPenawaran)
    {
        if ($request->user()->hasRole('vendor')) {
            return view('penawarans.changes.index')->with(
                [
                    'penawaran' => Penawaran::where(
                        'id', $idPenawaran
                    )->latest()->get(),
                    'penawaranChanges' => PenawaranChange::with(
                        'penawaran'
                    )->where('penawaran_id', $idPenawaran)
                    ->latest()->get(),
                ]
            );
        }
    }
}
```

Potongan Kode 5.36. PenawaranChangeController

Controller FileDownloadController mengatur proses mengunduh *file* penawaran yang diunggah di sistem. *Controller* ini memiliki satu *method* yaitu downloadFilePenawaran(). *Method* ini mengambil data PenawaranChange sesuai dengan *id* yang dikirim, kemudian mengembalikan *file* sesuai dengan *path* yang ada pada atribut *file* penawaran.

```
<?php
namespace App\Http\Controllers;
// use directives
```

```
class FileDownloadController extends Controller
{
    public function downloadFilePenawaran(
        string $penawaranChangeId
    ) {
        $penawaranChange = PenawaranChange::with(
            'penawaran'
            )->where('id', $penawaranChangeId)
            ->first();
        $filepath = Storage::disk('local')
        ->path($penawaranChange->filePenawaran);
        return Storage::download($filepath);
    }
}
```

Potongan Kode 5.37. FileDownloadController

Controller terakhir adalah NotificationController. *Controller* ini mengatur *view* dan proses *back end* yang menggunakan model Notification. *Method* index() berfungsi untuk menampilkan *view* semua notifikasi yang diterima oleh pengguna dengan *role* admin. Implementasi *controller* ini terdapat pada 0.

```
<?php
namespace App\Http\Controllers;
// use directives
class NotificationController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index(Request $request): View
    {
        $userId = Auth::user()->id;
        $user = User::find($userId);
        if ($request->user()->hasRole('admin')) {
            // next time pikir pagination
            return view('admin.notification.index')->with(
                [
                    'notifications' => Notification::where(
                        'notifiable_type',
                         "=",
```

```
                        "App\Models\User"
                    )->where("notifiable_id", "=", $userId)
                    ->latest()->paginate(5),
                ]
            );
        }
        return view('dashboard');
    }
}
```

Potongan Kode 5.38. NotificationController

5.1.4. *Background Jobs* dan Notifikasi

Selain pembahasan arsitektur MVC, fitur *background jobs* dan *notifications* juga akan dibahas. Fitur *background jobs* dan *notifications* adalah dua fungsi yang dipermudah implementasinya dengan *framework* Laravel. Dengan background jobs, pengguna dapat melanjutkan kegiatan lain sementara suatu proses yang memakan waktu panjang berjalan. Selain itu, *background jobs* juga membuat proses tersebut dapat berjalan walaupun batasan *response timeout* dari suatu sistem web cukup rendah. Di sisi lain, notifikasi berfungsi untuk memberi tahu pengguna tentang hal yang terjadi pada sistem. Dalam sistem ini, notifikasi digunakan utamanya untuk memberi tahu pengguna bahwa *background jobs* berhasil atau gagal.

Background job CreateQuotationSheetJob berfungsi untuk menjalankan method createQuotationSheetsWithTemplate() di *background*. *Job* ini juga akan memanggil fungsi untuk mengirim notifikasi QuotationSheetsCreateSuccess ketika proses berhasil dilakukan. Jika proses gagal dilakukan, *job* ini akan memanggil *method* fail yang selanjutnya akan memanggil fungsi untuk mengirim notifikasi QuotationSheetsCreateFailed.

```
<?php
namespace App\Jobs;
// use directives
class CreateQuotationSheetJob implements ShouldQueue
{
    use Queueable;
```

```
    // Config attributes: ...
    public float $startTime;
    public $error;
    // Buat instansi job baru
    public function __construct(
        protected array $invoiceItems,
        public string $noRfq,
        protected string $tanggalRfq,
        protected string $totalHarga,
        protected string $tanggalSekarang,
        protected string $templateFilePath,
        public $userId,
    ){}
    // Jalankan job
    public function handle(
    ): void {
        $penawaranService = new PenawaranService();
        $client = $penawaranService->createClient();
        $driveService = new Drive($client);
        $sheetsService = new Sheets($client);
        $user = User::find($this->userId);
        $this->startTime = microtime(true);
        try {
            $penawaranService
            ->createQuotationSheetsWithTemplate(
                $this->invoiceItems,
                $this->noRfq,
                $this->tanggalRfq,
                $this->totalHarga,
                $this->tanggalSekarang,
                $this->templateFilePath,
                $driveService,
                $sheetsService
            );
            logger('Fungsi
            CreateQuotationSheetsWithTemplate tidak
            melempar exception.');
            $duration = microtime(true)-$this
            ->startTime;
            $user->notify(new QuotationSheetsCreateSuccess(
                $this->noRfq, $duration, $this->userId
```

```
            ));
        } catch (Exception $e) {
            $this->error = $e;
            logger('Logged from catch in
            CreateQuotationSheetJob' . $e->getMessage());
            $this->fail();
        }
    }
}
```

Potongan Kode 5.39. *Background job* CreateQuotationSheetJob

Selanjutnya, *class* notifikasi pertama yang akan dibahas adalah QuotationSheetsCreateSuccess. *Class* ini merepresentasikan notifikasi yang muncul saat pembuatan surat RfQ dengan *background job* berhasil. *Method* __construct() pada *class* ini melakukan inisialisasi atribut dari suatu objek dari *class* ini. *Method* via() mengatur notifikasi akan dikirim ke mana; dalam kasus ini, notifikasi dikirim ke basis data dan *broadcast (event* yang dapat ditangkap oleh *front end*). *Method* toArray() dan toBroadcast() mengatur struktur data notifikasi untuk tujuan yang sudah ditentukan. Terakhir, *method* broadcastType() dan databaseType() mengatur tipe dari notifikasi yang digunakan untuk membedakannya dari notifikasi lainnya.

```
<?php
namespace App\Notifications;
// use directives
class QuotationSheetsCreateSuccess extends Notification
{
    use Queueable;
    protected string $message;
    public function __construct(
        public string $noRfq, public float $duration,
        protected $userId
    ) {
        $this->message = "Successfully created the quotation for
        RFQ $this->noRfq. Time elapsed: $this->duration";
    }
    public function via(object $notifiable): array
```

```
    { return ['broadcast', 'database',]; }
    public function toArray(object $notifiable): array
    {
        return [
            'noRfq' => $this->noRfq,
            'duration' => $this->duration,
            'message' => $this->message,
            'dateFinished' => date('Y-m-d H:i:s'),
        ];
    }
    public function toBroadcast(
        object $notifiable): BroadcastMessage
    {
        return new BroadcastMessage([
            'noRfq' => $this->noRfq,
            'duration' => $this->duration,
            'message' => $this->message,
            'dateFinished' => date('Y-m-d H:i:s'),
        ]);
    }
    public function broadcastType(): string
    { return 'Penawaran.CreateQuotationSheets.success'; }
    public function databaseType(
        object $notifiable): string
    { return 'Penawaran.CreateQuotationSheets.success'; }
}
```

Potongan Kode 5.40. Notifikasi QuotationSheetsCreateSuccess

Class notifikasi selanjutnya adalah *QuotationSheetsCreateFailed* yang merepresentasikan notifikasi yang dikirim ketika pembuatan surat RfQ menggunakan *background job* gagal dilakukan. Isi dari *class* ini hampir sama dengan *QuotationSheetsCreateSuccess*; Perbedaannya terdapat pada nilai yang di-*assign* pada atribut *message* dan tipe notifikasinya, yang mana *class* ini akan bertipe 'Penawaran.CreateQuotationSheets.error'.

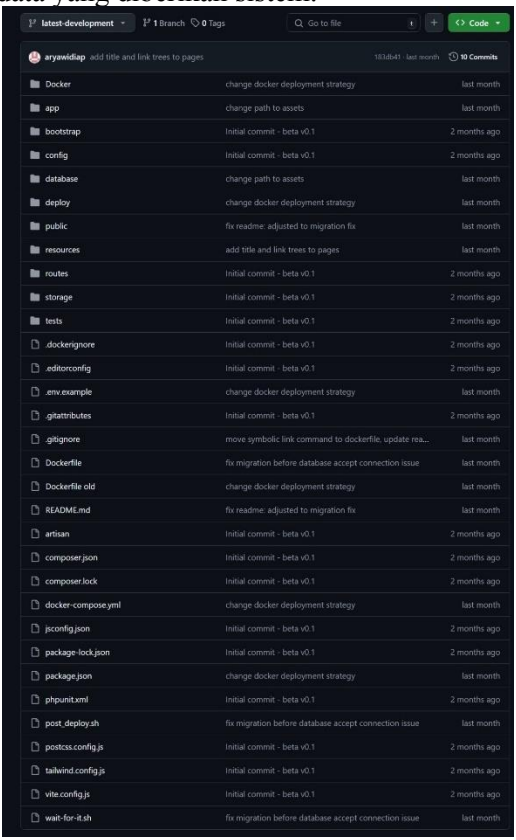
```
<?php
namespace App\Notifications;
// use directives
```

```
class QuotationSheetsCreateFailed extends Notification
{
    use Queueable;
    protected string $message;
    public function __construct(
        public string $noRfq, public float $duration,
        public $errorMessage, protected $userId,
    ) {
        $this->message = "Failed to create the quotation for RFQ
        $this->noRfq. Error message: $this->errorMessage";
    }
    public function via(object $notifiable): array
    { return ['broadcast', 'database',]; }
    public function toArray(object $notifiable): array
    {
        return [
            'noRfq' => $this->noRfq,
            'duration' => $this->duration,
            'message' => $this->message,
            'dateFinished' => date('Y-m-d H:i:s'),
        ];
    }
    public function toBroadcast(
        object $notifiable): BroadcastMessage
    {
        return new BroadcastMessage([
            'noRfq' => $this->noRfq,
            'duration' => $this->duration,
            'message' => $this->message,
            'dateFinished' => date('Y-m-d H:i:s'),
        ]);
    }
    public function broadcastType(): string
    { return 'Penawaran.CreateQuotationSheets.error'; }
    public function databaseType(object $notifiable): string
    { return 'Penawaran.CreateQuotationSheets.error'; }
}
```

Potongan Kode 5.41. Notifikasi QuotationSheetsCreateFailed

5.2. Implementasi Sistem Turnamen Pingpong

Sistem Turnamen Pingpong adalah sebuah sistem berbasis web yang berfungsi untuk membantu manajemen data turnamen pingpong atau tenis meja dengan tujuan pembuatan lembar nilai turnamen. Sistem ini dibangun dengan pola desain *model-view-controller* (MVC) dengan *framework* Laravel dan Vue.js. Sistem ini memanfaatkan *module* PHPSpreadsheet untuk membuat *file* Excel dari data yang diberikan sistem.



Gambar 5.2. Direktori dari repositori Sistem Turnamen Pingpong

5.2.1. *Model*

Sistem Turnamen Pingpong memiliki empat *model*, yaitu User, Turnamen, Pertandingan, dan Peserta. Relasi antar *model* dapat dilihat pada bab sebelumnya, pada Gambar 4.x. Model pertama adalah User, merepresentasikan pengguna dari sistem. User memiliki atribut *name*, *email*, dan *password*. User dapat memiliki beberapa Turnamen, ditunjukkan dengan *method* turnamens() yang meng-*extend* HasMany. Implementasi kode untuk model User dapat dilihat pada **0**.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Relations\HasMany;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;
    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];
    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
```

```
        'remember_token',
    ];
    /**
     * Get the attributes that should be cast.
     *
     * @return array<string, string>
     */
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }
    public function turnamens(): HasMany
    {
        return $this->hasMany(Turnamen::class);
    }
}
```

Potongan Kode 5.42. Model User (User.php)

Selanjutnya, sistem memiliki model Turnamen yang diimplementasikan dengan kode yang terdapat pada **0**. Turnamen memiliki atribut nama, panitia, asal panitia, logoKanan, dan logoKiri. Turnamen dapat memiliki beberapa pertandingan, direpresentasikan dengan metode pertandingans() yang meng-*extend* kelas HasMany. Lalu, metode user() yang meng-*extend* kelas BelongsTo menunjukkan User pemilik dari suatu Turnamen. Model ini juga memiliki metode cekKelengkapanPesertaPertandingan() yang berfungsi untuk memeriksa apakah peserta yang didaftarkan ke tiap pertandingan sudah sesuai banyaknya dengan jumlah peserta yang disebutkan saat pembuatan pertandingan.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
```

```
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Turnamen extends Model
{
    use HasFactory;
    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'nama',
        'panitia',
        'asalPanitia',
        'logoKanan',
        'logoKiri',
    ];
    public function pertandingans(): HasMany
    {
        return $this->hasMany(Pertandingan::class);
    }

    public function user(): BelongsTo
    {
        return $this->belongsTo(User::class);
    }

    public function cekKelengkapanPesertaPertandingan(): bool
    {
        if($this->pertandingans()->get()->count() === 0) {
            return false;
        }
        foreach ($this->pertandingans()->get() as $pertandingan)
{
            if($pertandingan->jumlahPeserta !==
count($pertandingan->pesertas()->get())){
                return false;
            }
        }
```

```
        return true;
    }
}
```

Potongan Kode 5.43. Model Turnamen (Turnamen.php)

Lalu, sistem memiliki model Pertandingan. Model pertandingan memiliki atribut jenisPertandingan dan jumlahPeserta. Suatu pertandingan memiliki beberapa peserta, yang direpresentasikan dengan metode pesertas() yang meng-*extend* kelas HasMany. Selain itu, suatu pertandingan dimiliki oleh satu turnamen, yang direpresentasikan dengan metode turnamen() yang meng-*extend* kelas BelongsTo. Implementasi kode model Pertandingan dapat dilihat pada 0.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Pertandingan extends Model
{
    use HasFactory;
    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'jenisPertandingan',
        'jumlahPeserta',
    ];
    public function turnamen(): BelongsTo
    {
        return $this->belongsTo(Turnamen::class);
    }
    public function pesertas(): HasMany
```

```
    {
        return $this->hasMany(Peserta::class);
    }
}
```

Potongan Kode 5.44. Model Pertandingan (Pertandingan.php)

Terakhir, sistem memiliki model Peserta. Model Peserta memiliki atribut nama, dari, kodeTabel, dan nomorUrut. Implementasi kode model Peserta dapat dilihat pada 0. Model Peserta memiliki metode pertandingan() yang meng-*extend* BelongsTo guna merepresentasikan pertandingan yang memiliki suatu peserta.

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Peserta extends Model
{
    use HasFactory;
    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'nama',
        'dari',
        'kodeTabel',
        'nomorUrut'
    ];
    public function pertandingan(): BelongsTo
    {
        return $this->belongsTo(Pertandingan::class);
    }
}
```

Potongan Kode 5.45. Model Peserta (Peserta.php)

5.2.2. *View*

View dalam implementasi Sistem Turnamen Pingpong dibuat menggunakan Vue.js. Format secara umum dari semua View yang dapat dilihat oleh pengguna atau User yang *authenticated* dapat dilihat pada potongan kode 0. Sesuai dengan kode tersebut, penulis akan merujuk pada bagian-bagian yang diubah atau ditambah pada masing-masing view dengan nama yang ditebalkan pada potongan kode, yaitu **bagian script** dan **konten halaman**. "Judul halaman" juga akan berubah untuk tiap *view*, tetapi dampaknya terhadap proses bisnis secara umum dapat diabaikan dan tidak akan dibahas secara mendetail. Penulis juga akan menghapus beberapa bagian pada kode (seperti atribut suatu *tag*) atau menggantinya dengan elipsis (...) jika bagian tersebut dianggap dapat disingkat pada kode dan tidak memengaruhi proses bisnis.

```
<!-- Main authenticated layout -->
<script setup>
// Bagian script
</script>
<template>
    <Head title="Judul Halaman" />

    <AuthenticatedLayout>
        <!-- Konten halaman -->
    </AuthenticatedLayout>

</template>
```

Potongan Kode 5.46. *Template* potongan kode dari *View* untuk pengguna yang *authenticated*

View pertama adalah Turnamens/Create.vue yang merupakan kode untuk halaman pembuatan turnamen. Pada bagian *script*, ada beberapa variabel dan fungsi yang dideklarasikan. Variabel *form* adalah variabel yang menampung data yang akan diambil dari formulir pembuatan turnamen. Fungsi *changeLogoKanan* dan *changeLogoKiri* adalah fungsi untuk

membantu pengunggahan *file* logo kanan dan logo kiri. Hal ini diperlukan mengingat pembangunan sistem menggunakan Vue.js dan bukan HTML murni. Bagian ini dapat dilihat pada **Potongan Kode 5.47**.

```
const form = useForm({
    nama: '',
    panitia: '',
    asalPanitia: '',
    logoKanan: '',
    logoKiri: '',
});

const changeLogoKanan = (e) => {
    form.logoKanan = e.target.files[0];
};

const changeLogoKiri = (e) => {
    form.logoKiri = e.target.files[0];
};
```

Potongan Kode 5.47. Bagian *script* dari view Turnamens/Create

Konten halaman view Turnamens/Create berisi dua bagian utama, yaitu judul halaman dan formulir yang dapat dilihat pada **0**. Atribut @submit.prevent dari elemen *form* menunjukkan bahwa formulir ini akan mengirim data dari variabel form, yang dijelaskan sebelumnya, dengan metode *post* ke *route* 'turnamen.store' dan jika sukses, variabel *form* akan diset ke awal mula.

```
<!-- Judul halaman -->
<div class="mb-4">
    <h2 class="text-2xl font-bold">Buat Turnamen</h2>
</div>
<!-- Formulir -->
<form @submit.prevent="form.post(route('turnamens.store'),
{ onSuccess: () => form.reset() })">
    <!-- Konten formulir -->
    <PrimaryButton class="mt-4">Buat Turnamen</PrimaryButton>
</form>
```

Potongan Kode 5.48. Konten halaman dari view

Turnamens/Create

Konten formulir tersebut memiliki beberapa input. **0** menampilkan *input* yang berupa teks, yaitu nama turnamen, panitia, dan asal panitia. Atribut v-model pada masing-masing *input* memungkinkan *input* teks menjadi terikat dengan elemen variabel *form* yang bersesuaian. Selanjutnya, *input* untuk logo kanan dan logo kiri adalah *file* gambar, di mana agar *file* terikat pada variabel *form*, *input* dibuat terikat dengan fungsi changeLogoKiri atau changeLogoKanan yang terpanggil setiap logo di-*input*.

```
<!-- Form Content: Text Inputs-->
<div class="form-group mt-4">
    <label for="nama">Nama Turnamen</label>
    <input v-model="form.nama" id="nama".../>
</div>
<InputError :message="form.errors.nama" class="mt-2"/>

<div class="form-group mt-4">
    <label for="panitia">Panitia</label>
    <input v-model="form.panitia" id="panitia".../>
</div>
<InputError :message="form.errors.panitia" class="mt-2"/>

<div class="form-group mt-4">
    <label for="asalPanitia">Asal Panitia</label>
    <input v-model="form.asalPanitia" id="asalPanitia".../>
</div>
<InputError :message="form.errors.asalPanitia" class="mt-2"/>

<!-- Form Content: Logo Kanan dan Logo Kiri-->
<div class="form-group mt-4">
    <label for="logoKanan">Logo Kanan</label>
    <input name="logoKanan" type="file" id="logoKanan"
@input="changeLogoKanan" .../>
</div>
<InputError :message="form.errors.logoKanan" class="mt-2"/>

<div class="form-group mt-4">
    <label for="logoKiri">Logo Kiri</label>
    <input name="logoKiri" type="file" id="logoKiri"
```

```
@input="changeLogoKiri" .../>
</div>
<InputError :message="form.errors.logoKiri" class="mt-2"/>
```

Potongan Kode 5.49. Konten formulir pada view Turnamens/Create

Selanjutnya, file Turnamens/Index.vue adalah kode yang membangun halaman daftar turnamen yang dibuat atau dimiliki oleh User. Bagian script Turnamens/Index.vue berisi kode pada **0**. Fungsi defineProps pada Vue.js berfungsi untuk mendeklarasikan props. Props adalah atribut yang dapat diteruskan ke *component* Vue, yang berisi data untuk ditampilkan dalam *component*. *Props* 'turnamens' akan terikat dengan data daftar turnamen yang dimiliki. Props ini dapat di-*loop* untuk tiap komponen yang dapat dilihat di **Potongan Kode 5.51**.

```
defineProps(['turnamens']);
```

Potongan Kode 5.50. Bagian *script* dalam *view* Turnamen/Index

Bagian selanjutnya dari *file* Turnamens/Index.vue adalah konten halaman yang berisi judul dan tabel daftar turnamen. Tabel ini akan menampilkan pesan "Belum ada turnamen yang dibuat di sistem." jika *user* belum membuat turnamen, berkat penggunaan atribut v-if. Jika ada turnamen yang dimiliki turnamen, props turnamens akan di-*loop*, di mana tiap elemennya akan diberi alias turnamen dan diteruskan ke komponen Turnamen.

```
<div class="mb-4">
    <h2 class="text-2xl font-bold">Daftar Turnamen</h2>
</div>
<a :href="route('turnamens.create')">
    <SecondaryButton class="mb-2">+ Tambah
turnamen</SecondaryButton>
</a>
<table class="table table-striped mx-auto w-full">
    <thead>
        <tr class="h-9 bg-green-700 text-green-50">
            <th scope="col"...>No</th>
            <th scope="col"...>Nama Turnamen</th>
```

```
            <th scope="col"...>Panitia</th>
            <th scope="col"...>Asal Panitia</th>
            <th scope="col"...></th>
        </tr>
    </thead>
    <tbody>
        <tr v-if="turnamens.length === 0"...>
            <td colspan="5"...>Belum ada turnamen yang dibuat di
sistem.</td>
        </tr>
        <Turnamen
        v-for="(turnamen, index) in turnamens"
        :key="turnamen.id"
        :turnamen="turnamen"
        :index="index"
        />
    </tbody>
</table>
```

Potongan Kode 5.51. Konten halaman dalam *view* Turnamen/Index

Component <Turnamen> yang tercantum pada 0 adalah *component* yang merepresentasikan baris data turnamen dalam tabel daftar turnamen. Data-data dikirim dari *view* Turnamens/Index.vue ke *component* ini untuk diolah menjadi baris data turnamen. Selain keterangan dari suatu turnamen, baris data juga mengandung tombol 'Lihat detail' yang berfungsi untuk membuka halaman detail dari suatu turnamen.

```
<script setup>
import SecondaryButton from '@/Components/SecondaryButton.vue';
defineProps(['turnamen', 'index']);
</script>

<template>
    <tr class="bg-gray-50">
        <th scope="row" ...>{{ props.index+1 }}</th>
        <td ...>{{ props.turnamen.nama }}</td>
        <td ...>{{ props.turnamen.panitia }}</td>
        <td ...>{{ props.turnamen.asalPanitia }}</td>
```

```
        <th ...>
            <a :href="route('turnamens.show', props.turnamen)">
                <SecondaryButton>Lihat detail</SecondaryButton>
            </a>
        </th>
    </tr>
</template>
```

Potongan Kode 5.52. *Component* <Turnamen>

Lalu, file Turnamens/Show.vue adalah kode untuk menampilkan detail dari turnamen yang dibuat atau dimiliki oleh User. Bagian script Turnamens/Show.vue dapat dilihat pada **0**. Serupa dengan Turnamens/Indeks.vue, bagian script juga mengandung defineProps, yaitu untuk data 'turnamen', 'pertandingans', 'isPesertaLengkap', dan 'exportMessage'.

```
defineProps(['turnamen', 'pertandingans', 'isPesertaLengkap',
'exportMessage',]);
```

Potongan Kode 5.53. Bagian script dalam view Turnamens/Show

Konten halaman Turnamens/Show terdiri dari deskripsi turnamen, tombol *export*, dan tabel pertandingan yang termasuk dalam suatu turnamen. Deskripsi turnamen didapat dari *props* 'turnamen'. Tombol export akan hanya muncul saat peserta lengkap, ditunjukkan oleh *props* 'isPesertaLengkap'. Serupa dengan tabel turnamen, jika belum ada pertandingan yang didaftarkan ke turnamen, tabel akan menunjukkan pesan "Belum ada pertandingan yang dimasukkan ke turnamen ini.". Setiap pertandingan direpresentasikan dengan *component* <Pertandingan>.

```
<div>
    <div>
        <h2>{{ turnamen.nama }}</h2>
        <p>Kode turnamen: {{ turnamen.id }}</p>
        <p>Panitia: {{ turnamen.panitia }}</p>
        <p>Asal Panitia: {{ turnamen.asalPanitia }}</p>
    </div>
```

```
    <div class="">
        <a
            v-if="isPesertaLengkap"
            :href="route('turnamens.export', turnamen)"
        >
            <SecondaryButton ...>
                Simpan ke file XLSX
            </SecondaryButton>
        </a>
    </div>
</div>
<!-- Tombol export -->
<p v-if="exportMessage">Export message: {{ exportMessage }}</p>
<!-- Tabel daftar pertandingan -->
<table ...>
    <thead>
        <tr ...>
            <th ...">No</th>
            <th ...>Jenis pertandingan</th>
            <th ...>Jumlah peserta</th>
            <th ...></th>
        </tr>
    </thead>
    <tbody>
        <tr v-if="pertandingans.length === 0" ...>
            <td colspan="4" ...>
                Belum ada pertandingan yang dimasukkan ke
turnamen ini.
            </td>
        </tr>
        <Pertandingan
        v-for="(pertandingan, index) in pertandingans"
        :key="pertandingan.id"
        :pertandingan="pertandingan"
        :turnamen="turnamen"
        :index="index"
        />
        <tr class="border-b border-slate-300 bg-gray-50">
            <td colspan="4" class="px-2 py-2">
                <a
:href="route('turnamens.pertandingans.create', turnamen)"
```

```
                >
                    <SecondaryButton ...>
                        + Tambah pertandingan
                    </SecondaryButton>
                </a>
            </td>
        </tr>
    </tbody>
</table>
```

Potongan Kode 5.54. Bagian script dalam view Turnamen/Show

Component <Pertandingan> adalah *component* yang merepresentasikan baris data pertandingan dalam tabel daftar pertandingan. *Component* ini menerima data pertandingan, turnamen, dan *index* (urutan pada tabel) dari kode yang menggunakannya, kemudian menjadikannya data-data pada suatu baris tabel. Selain data-data, baris pertandingan juga berisi tombol 'Lihat peserta' yang berfungsi untuk membuka detail pertandingan dan peserta yang terdaftar dalam pertandingan tersebut. Kode *component* ini tercantum pada **0**.

```
<script setup>
import SecondaryButton from '@/Components/SecondaryButton.vue';
defineProps(['pertandingan','turnamen', 'index']);
</script>

<template>
    <tr class="bg-gray-50">
        <th scope="row" ...>{{ index+1 }}</th>
        <td ...>{{ pertandingan.jenisPertandingan }}</td>
        <td ...>{{ pertandingan.jumlahPeserta }}</td>
        <th ...>
            <a
                :href="route('turnamens.pertandingans.pesertas.i
ndex',
                [turnamen,pertandingan,])"
            >
                <SecondaryButton>
                    Lihat peserta
                </SecondaryButton>
```

```
            </a>
        </th>
    </tr>
</template>
```

Potongan Kode 5.55. *Component* <Pertandingan>

View selanjutnya adalah *view* formulir pertandingan, yang diimplementasikan dengan *file* Pertandingans/Create. Pertama, bagian *script* berisi variabel form yang akan menyimpan data dari formulir *input* dan *props* yang berfungsi untuk menerima data dari server atau *parent component*. Variabel form memiliki dua elemen, yaitu jenisPertandingan dan jumlahPeserta. *Props* yang dideklarasikan adalah turnamen untuk menyimpan data turnamen yang dikirim dari server.

```
const form = useForm({
    jenisPertandingan: null,
    jumlahPeserta: null,
});

defineProps(['turnamen']);
```

Potongan Kode 5.56. Bagian script dalam view Pertandingans/Create

Konten halaman dari *view* formulir pertandingan berisi judul halaman dan formulir untuk pembuatan pertandingan. Formulir berisi dua *input*, yaitu *text input* untuk jenis pertandingan dan *number input* untuk jumlah peserta. *Number input* memiliki atribut min="4" dan step="4". Atribut min berfungsi untuk membatasi *input* agar bernilai minimal 4, sedangkan atribut step membatasi agar *input* selalu kelipatan 4.

```
<div>
    <!-- Judul Halaman -->
    <div>
        <h2>Buat Pertandingan</h2>
    </div>
    <!-- Formulir pembuatan pertandingan -->
    <form
```

```
    @submit.prevent="form.post(route
    ('turnamens.pertandingans.store', turnamen),
    { onSuccess: () => form.reset() })">
        <div>
            <label for="jenisPertandingan">
                Jenis Pertandingan
            </label>
            <input v-model="form.jenisPertandingan"
            id="jenisPertandingan" />
        </div>
        <InputError
        :message="form.errors.jenisPertandingan" />
        <div>
            <label for="jumlahPeserta">
                Jumlah Peserta
            </label>
            <input
            type="number" min="4" step="4"
            v-model="form.jumlahPeserta"
            id="jumlahPeserta" placeholder="0"/>
        </div>
        <InputError :message="form.errors.jumlahPeserta" />
        <p>Banyak tabel: {{ form.jumlahPeserta / 4 }}</p>
        <PrimaryButton>Buat Pertandingan</PrimaryButton>
    </form>
</div>
```

Potongan Kode 5.57. Konten halaman pada view Pertandingans/Create

View selanjutnya adalah *view* detail pertandingan dan daftar peserta yang diimplementasikan dengan Pesertas/Index.vue. Bagian *script* view ini (**0**) berisi props yang digunakan untuk menerima data dari *back end*. Props turnamen dan pertandingan digunakan untuk *keep track route* yang akan dituju. *Props* pesertas berisi data peserta-peserta yang terdaftar pada pertandingan tertentu.

```
const props = defineProps([
    'turnamen','pertandingan','pesertas',
]);
```

Potongan Kode 5.58. Bagian *script* pada *view* Pesertas/Index.

Konten halaman *view* Pesertas/Index terdiri dari deskripsi pertandingan dan tabel daftar peserta. Tabel daftar peserta menampilkan kode tabel, nomor urut, nama peserta, dan asal peserta. Jika belum ada peserta yang didaftarkan, pesan "Belum ada peserta yang didaftarkan" akan muncul. Selain itu, tombol "Tambah peserta" juga akan muncul saat belum ada peserta yang didaftarkan.

```
<div>
    <!-- Deskripsi turnamen -->
    <div>
        <h2>
            Pertandingan {{ pertandingan.jenisPertandingan}}
        </h2>
        <p>
            Jumlah Peserta: {{ pertandingan.jumlahPeserta }}
        </p>
        <p>Turnamen: {{ turnamen.nama }}</p>
    </div>

    <!-- Tabel daftar peserta -->
    <table>
        <thead>
            <tr>
                <th scope="col">Kode Tabel</th>
                <th scope="col">Nomor Urut</th>
                <th scope="col">Nama Peserta</th>
                <th scope="col">Asal</th>
            </tr>
        </thead>
        <tbody>
            <Peserta
            v-for="(peserta, index) in pesertas"
            :key="peserta.id"
            :peserta="peserta"
            :index="index"
            />
            <tr v-if="pesertas.length === 0">
                <td colspan="4">
```

```
                    Belum ada peserta yang
                    didaftarkan ke pertandingan ini.
                </td>
            </tr>
            <tr v-if="pertandingan.jumlahPeserta !== pesertas.length">
                <td colspan="4">
                    <a
                    :href="route('turnamens.pertandingans.pesertas.
                    create',
                    [turnamen,pertandingan,])">
                        <SecondaryButton>
                            + Tambah peserta
                        </SecondaryButton>
                    </a>
                </td>
            </tr>
        </tbody>
    </table>
</div>
```

Potongan Kode 5.59. Konten halaman pada *view* Pesertas/Index

Component <Peserta> adalah *component* yang merepresentasikan baris data suatu peserta dalam tabel daftar peserta pertandingan. Data suatu pertandingan, turnamen yang memiliki pertandingan tersebut, dan salah satu pesertanya diteruskan dari *parent component/code*, dalam hal ini Pertandingans/Show.vue. Lalu, data-data tersebut diproses menjadi baris data peserta yang berisi kode tabel, nomor urut, nama, dan dari (asal) peserta. Kode *component* dicantumkan dalam 0.

```
<script setup>
defineProps(['pertandingan','turnamen','peserta','index']);
</script>

<template>
    <tr ...>
        <th ...>
            {{ peserta.kodeTabel }}
        </th>
        <td ...>
```

```
            {{ peserta.nomorUrut }}
        </td>
        <td ...>
            {{ peserta.nama }}
        </td>
        <td ...>
            {{ peserta.dari }}
        </td>
    </tr>
</template>
```

Potongan Kode 5.60. *Component* <Peserta>

Halaman input data peserta merupakan halaman yang lebih kompleks dari halaman yang lain. Halaman input data peserta diimplementasikan dalam kode Pesertas/Create.vue. Bagian *script* untuk halaman ini terdiri dari beberapa bagian. Bagian pertama, dapat dilihat pada 0, adalah *class* Peserta. *Class* ini digunakan untuk enkapsulasi data-data peserta. Adanya *class* ini membuat data peserta terstandardisasi dan lebih mudah diakses dengan mengakses atribut. *Class* ini memiliki atribut ide, nama, dari, kodeTabel, nomorUrut, dan kodeTabelDanNomorUrut.

```
// Template for storing Peserta data temporarily
class Peserta {
    id;
    nama;
    dari;
    kodeTabel;
    nomorUrut;
    kodeTabelDanNomorUrut;

    constructor(id, nama, dari, kodeTabel, nomorUrut,
kodeTabelDanNomorUrut,) {
        this.id = id;
        this.nama = nama;
        this.dari = dari;
        this.kodeTabel = kodeTabel;
        this.nomorUrut = nomorUrut;
        this.kodeTabelDanNomorUrut = kodeTabelDanNomorUrut;
    }
```

```
    setKodeTabel_nomorUrut() {
        const kodeTabel_nomorUrutArray =
this.kodeTabelDanNomorUrut.split('-');
        if(kodeTabel_nomorUrutArray.length === 2) {
            this.kodeTabel = kodeTabel_nomorUrutArray[0];
            this.nomorUrut =
parseInt(kodeTabel_nomorUrutArray[1]);
        }
    }

    resetKodeTabel_nomorUrut() {
        this.kodeTabel = '';
        this.nomorUrut = null;
        this.kodeTabelDanNomorUrut = '';
    }

    gantiPosisi(kodeTabel, nomorUrut) {
        this.kodeTabel = kodeTabel;
        this.nomorUrut = nomorUrut;
        this.kodeTabelDanNomorUrut = `${kodeTabel}-
${nomorUrut}`;
    }
}
```

Potongan Kode 5.61. *Class* Peserta untuk enkapsulasi data peserta pada Pesertas/Create.vue

Class ini juga memiliki beberapa *methods*. *Method* setKodeTabel_nomorUrut berfungsi untuk mengeset kode tabel dan nomor urut secara terpisah karena pada baris input peserta digunakan atribut kodeTabelDanNomorUrut yang menggabungkan keduanya. Lalu, *method* resetKodeTabel_nomorUrut() digunakan untuk mengembalikan atribut kodeTabel, nomorUrut, dan kodeTabelDanNomorUrut ke nilai semua, yaitu *null* atau *empty string*. Terakhir, *method* gantiPosisi digunakan untuk mengubah kodeTabel, nomorUrut, dan kodeTabelDanNomorUrut saat yang diketahui adalah kode tabel dan nomor urut secara terpisah.

Setelah deklarasi *class* Peserta, variabel-variabel dibuat dan dilakukan inisialisasi untuk menampung data secara sementara, seperti terlihat pada 0. Sebelum deklarasi variabel, *props* turnamen dan pertandingan dideklarasikan. Variabel *array* pesertas berfungsi untuk menampung data peserta-peserta pada formulir *input*, sedangkan *array* tabelPesertas berfungsi untuk menampung data peserta-peserta yang telah dimasukkan ke dalam tabel penilaian.

```
// Variables for temporary Data storing, variabel initialization
const props = defineProps(['turnamen','pertandingan',]);
const pesertas = ref([]);
const tabelPesertas = ref([]);

for (let index = 0; index < props.pertandingan.jumlahPeserta/4;
index++) {
    tabelPesertas.value.push({
        kode: 'A.'+String(index+1),
        rows: Array.from({ length: 4 } , (value, index) => ({
            index: index+1,
            peserta: new Peserta(null, null, null, null, null,
            null,),
            isFilled: false,
        })),
    });
}

for (let index = 0; index < props.pertandingan.jumlahPeserta;
index++) {
    pesertas.value.push( new Peserta(index+1, null, null, null,
    null, null,) );
}
```

Potongan Kode 5.62. Props untuk menyimpan data yang diteruskan ke Pesertas/Create.vue, variabel-variabel untuk menyimpan data sementara, dan inisialisasi awal variabel-variabel tersebut

Selanjutnya, file Pesertas/Create.vue memiliki fungsi cekPesertaAdaDiTabel() yang dapat dilihat pada 0. Fungsi ini berfungsi untuk mengembalikan kode tabel (tabelPeserta.kode), peserta, dan nomor baris jika peserta yang di-*input* (pesertaBaru)

ditemukan pada salah satu tabel. Jika peserta tidak ditemukan, fungsi ini mengembalikan *array* dengan tiga elemen bernilai *null* untuk menunjukkan ketiadaan peserta.

```
// Cek peserta ada di tabel
const cekPesertaAdaDiTabel = (pesertaBaru) => {
    for (const tabelPeserta of tabelPesertas.value) {
        const row = tabelPeserta.rows.find(({peserta}) =>
peserta.id === pesertaBaru.id);
        console.log(row);
        if(row) {
            return [tabelPeserta.kode, row, row.index];
        }
    }
    return [null, null, null];
}
```

Potongan Kode 5.63. Fungsi *cekPesertaAdaDiTabel*

Selanjutnya, fungsi tambahPesertaKeTabel adalah fungsi untuk menambahkan peserta dari formulir *input* ke dalam tabel pratinjau, yang mana nantinya akan dikirim ke *back end* (0). Parameter pesertaBaru pada fungsi ini adalah data peserta yang di-*input* oleh *user*. Pertama, fungsi mengambil kode tabel dan nomor urut dari peserta. Lalu, fungsi memeriksa apakah peserta tersebut sudah pernah dimasukkan ke dalam tabel pratinjau sebelumnya. Jika kode tabel peserta yang baru adalah null, peserta tersebut akan dikembalikan nilai posisinya ke null yang berarti *user* ingin mengeluarkan peserta dari tabel pratinjau. Jika kode tabel baru tidak null, fungsi akan memeriksa apakah posisi yang baru sudah diisi oleh peserta lain, diindikasikan dengan properti isFilled. Jika iya, peserta yang di-*input* akan dikembalikan ke posisi sebelumnya. Jika tidak, peserta akan dipindah ke posisi baru, dan posisi peserta sebelumnya pada tabel dibersihkan dengan cara menggantinya dengan objek Peserta "*empty*" dan mengeset isFilled ke *false*.

```
// Tambah peserta ke tabel
const tambahPesertaKeTabel = (pesertaBaru) => {
    let kodeTabelBaru = null;
    let nomorUrutBaru = null;
```

```
    if(pesertaBaru.kodeTabelDanNomorUrut !== null) {
        // Set dari gabungan kodeTabel dan nomorUrut
        pesertaBaru.setKodeTabel_nomorUrut();
        kodeTabelBaru = pesertaBaru.kodeTabel;
        nomorUrutBaru = pesertaBaru.nomorUrut;
    }

    // Cek peserta ada atau tidak di tabel
    const [kodeTabelPesertaLama, rowLama, rowIndexLama] =
cekPesertaAdaDiTabel(pesertaBaru);

    // Jika tabel baru null
    if(kodeTabelBaru === null) { // TRUE, set peserta.kodeTabel
dan nomorUrut ke null
        pesertaBaru.gantiPosisi(null, null);
    } else { // FALSE, cek new position
        // Cek sudah ada orang belum di posisi baru
        console.log(kodeTabelBaru);
        const tabelPesertaBaru =
tabelPesertas.value.find(({kode}) => kode === kodeTabelBaru);
        const isFilled = tabelPesertaBaru.rows[nomorUrutBaru-
1].isFilled;
        if(isFilled) {
            console.log("Posisi sudah terisi. Balikkan ke posisi
awal.");
            pesertaBaru.gantiPosisi(kodeTabelPesertaLama,
rowIndexLama);
        } else {
            // Masukkan ke posisi baru
            tabelPesertaBaru.rows[nomorUrutBaru-1].peserta =
pesertaBaru;
            tabelPesertaBaru.rows[nomorUrutBaru-1].isFilled =
true;
        }
    }

    if(kodeTabelPesertaLama !== null &&
pesertaBaru.kodeTabelDanNomorUrut !== `${kodeTabelPesertaLama}-
${rowIndexLama}`) {
        // Bersihkan posisi lama
```

```
        rowLama.peserta = new
Peserta(null,null,null,null,null,null,);
        rowLama.isFilled = false;
    }
}
```

Potongan Kode 5.64. Fungsi *tambahPesertaKeTabel*

Bagian terakhir dari bagian script file Pesertas/Create berurusan dengan pengiriman data ke *back end*. Variabel *form* digunakan untuk mengirim data ke *back end*. Berbeda dari *form* pada umumnya yang mengikat elemen *input* ke elemen variabel *form*, fungsi *tambahPesertasKeForm*. Implementasi bagian ini dapat dilihat pada 0.

```
// Pengiriman data ke backend
const form = useForm({
    pesertas: [],
});
//-- Set variable form sebelum dikirim ke backend
const tambahPesertasKeForm = () => {
    const formPesertas = pesertas.value.map((peserta) => {
        const {...pesertaObject} = peserta;
        return pesertaObject;
    });
    form.pesertas = formPesertas;
    console.log(form.pesertas);
}
```

Potongan Kode 5.65. Variabel form dan fungsi tambahPesertasKeForm untuk proses pengiriman data *input* ke *back-end*

Konten halaman Pesertas/Create.vue, dapat dilihat pada **0**, terdiri dari judul halaman, tabel pratinjau, dan tabel input peserta.

```
<!-- Judul Halaman -->
<div>
    <div>
        <h2>
            Tambah Peserta
        </h2>
```

```
    </div>
</div>

<div>
    <!-- Pratinjau daftar nilai -->
    <!-- Tabel untuk user input peserta -->
</div>
```

Potongan Kode 5.66. Konten halaman Pesertas/Create.vue

Kode pratinjau daftar nilai dapat dilihat pada 0. Tabel-tabel pratinjau akan dibuat dari variabel tabelPesertas. Variabel tabelPesertas dihubungkan dengan *input* melalui fungsi tambahPesertaKeTabel. Pada bagian baris data, kolom 4 sampai 7 diberi *conditional* untuk membuat sel berwarna secara diagonal sesuai dengan *requirement*.

```
<div>
    <div v-for="tabelPeserta in tabelPesertas">
        <h2>{{ tabelPeserta.kode }}</h2>
        <table>
            <thead>
                <tr>
                    <th scope="col">No</th>
                    <th scope="col">Nama Peserta</th>
                    <th scope="col">Asal</th>
                    <th scope="col">1</th>
                    <th scope="col">2</th>
                    <th scope="col">3</th>
                    <th scope="col">4</th>
                    <th scope="col">M</th>
                    <th scope="col">P</th>
                    <th scope="col">R</th>
                </tr>
            </thead>
            <tbody>
                <tr v-for="(row, index) in tabelPeserta.rows">
                    <th scope="row">
                        {{ index+1 }}
                    </th>
                    <td>{{ row.peserta.nama }}</td>
```

```
                    <td>{{ row.peserta.dari }}</td>
                    <td :class="{'bg-red-700 text-red-50': index
                    == 0}"></td>
                    <td :class="{'bg-red-700 text-red-50': index
                    == 1}"></td>
                    <td :class="{'bg-red-700 text-red-50': index
                    == 2}"></td>
                    <td :class="{'bg-red-700 text-red-50': index
                    == 3}"></td>
                    <td></td>
                    <td></td>
                    <td></td>
                </tr>
            </tbody>
        </table>
    </div>
</div>
```

Potongan Kode 5.67. Tabel untuk *user input* Peserta

Formulir untuk melakukan *input* data peserta ditampilkan pada sisi kanan halaman dan implementasinya dapat dilihat pada 0. Formulir ini akan mengirim variabel *form* ke *route* ‘turnamens.pertandingans.pesertas.store’.

```
<div>
    <form
    @submit.prevent="form.post(
        route(
            'turnamens.pertandingans.pesertas.store',
            [turnamen,pertandingan]
        ), { onSuccess: () => {} }
    )">
        <table>
            <thead>
                <tr>
                    <th scope="col">No</th>
                    <th scope="col">Nama Peserta</th>
                    <th scope="col">Dari</th>
                    <th scope="col">Kode Tabel</th>
                    <th scope="col">Action</th>
                </tr>
```

```
            </thead>
            <tbody>
                <!-- Baris input data -->
            </tbody>
        </table>
        <InputError
        :message="form.errors.pesertas"/>
        <InputError
        v-for="peserta in form.errors.pesertas"
        :message="peserta"/>
        <PrimaryButton
        @click="tambahPesertasKeForm()">
            Submit Peserta
        </PrimaryButton>
    </form>
</div>
```

Potongan Kode 5.68. Tabel untuk *user input* Peserta

Kode lebih detail dari baris input data dapat dilihat pada 0. Tiap baris terdiri dari *text input* untuk nama peserta yang terikat dengan atribut nama dari salah satu elemen variabel *pesertas* (diberi alias peserta pada potongan kode), *text input* untuk asal peserta yang terikat dengan atribut *dari* dari elemen yang sama, *selection dropdown* untuk kode tabel dan nomor urut, dan tombol untuk menambah peserta ke pratinjau. Saat tombol diklik, fungsi tambahPesertaKeTabel untuk memroses data *input*.

```
<tr
v-for="(peserta, index) in pesertas"
:id="'form-group-'+index">
    <th scope="row">{{ index+1 }}</th>
    <td><input v-model="peserta.nama"/></td>
    <td><input v-model="peserta.dari"/></td>
    <td>
        <select
        name="kodeTabel"
        v-model="peserta.kodeTabelDanNomorUrut">
            <option
            :value="null"
            :selected="peserta.kodeTabelDanNomorUrut == '' ||
```

```
peserta.kodeTabelDanNomorUrut == null">
                (none)
            </option>
            <TabelDanNomorUrut
            v-for="tabelPeserta in tabelPesertas"
            :tabelPeserta="tabelPeserta"
            :peserta="peserta"
            />
        </select>
    </td>
    <td>
        <button
        type="button"
        @click="tambahPesertaKeTabel(peserta)">
            Tambah ke tabel
        </button>
    </td>
</tr>
```

Potongan Kode 5.69. Baris *input* untuk tiap peserta

Component <TabelDanNomorUrut> adalah component yang digunakan untuk menunjukkan nomor tabel dan nomor urut yang ada. Component ini pada dasarnya adalah *tag* <option> HTML. *Tag* <option> diberi *conditional* untuk atribut *selected* di mana atribut *selected* akan ditambahkan jika kode tabel dan nomor urut <option> sama dengan kode tabel dan nomor urut peserta. Selain itu, agar suatu <option> tidak dapat dipilih jika sudah ada peserta yang tercatat pada posisi tersebut, <option> diberi *conditional* pada atribut *disabled*-nya di mana atribut *disabled* akan ditambahkan jika kode tabel dan nomor urut pada <option> tidak sama dengan kode tabel dan nomor urut pada peserta *dan* tabelPesertaRow.isfilled bernilai *true*. Implementasi *component* ini dapat dilihat pada 0.

```
<script setup>
defineProps(['tabelPeserta', 'peserta']);
</script>

<template>
```

```
    <option
    v-for="tabelPesertaRow in tabelPeserta.rows"
    :value="`${tabelPeserta.kode}-${tabelPesertaRow.index}`"
    :selected="`${tabelPeserta.kode}-${tabelPesertaRow.index}` ==
peserta.kodeTabelDanNomorUrut"
    :disabled="`${tabelPeserta.kode}-${tabelPesertaRow.index}` !==
peserta.kodeTabelDanNomorUrut && tabelPesertaRow.isFilled"
    :class="{ 'bg-red-100': `${tabelPeserta.kode}-${tabelPesertaRow.index}` !==
peserta.kodeTabelDanNomorUrut && tabelPesertaRow.isFilled }"
    >
        {{ tabelPeserta.kode }}-{{tabelPesertaRow.index}}
    </option>
</template>
```

Potongan Kode 5.70. *Component* <TabelDanNomorUrut>

5.2.3. *Controller*

Sistem Turnamen Pingpong memiliki beberapa *controller*. Namun, tiga controller menjadi controller utama dalam proses bisnis, yaitu TurnamenController, PertandinganController, dan PesertaController. Controller-controller ini memiliki fungsi utama untuk menyambungkan *front end* (*view*) dan *back end* (model dan basis data) dan melakukan proses lainnya yang berjalan di *back end*.

Controller TurnamenController memiliki fungsi utama untuk menghubungkan bagian-bagian yang menggunakan model Turnamen. 0 menunjukkan fungsi-fungsi yang ada pada TurnamenController, yaitu index(), show(), create(), store(), dan export(). Potongan kode ini juga menunjukkan detail implementasi fungsi-fungsi tersebut, kecuali store() dan export() yang akan ditunjukkan di potongan kode berikutnya. Tiga fungsi pertama ini mengembalikan *render view* ke *client* yang sudah disertai dengan data yang diperlukan. Fungsi index berfungsi untuk mengembalikan *render view* daftar turnamen, fungsi show berfungsi untuk mengembalikan *render view* detail suatu turnamen, dan fungsi create berfungsi mengembalikan *render view* formulir pembuatan turnamen.

```
<?php
namespace App\Http\Controllers;
```

```
// use directives
class TurnamenController extends Controller
{
    public function index(): Response
    {
        $user = Auth::user();
        return Inertia::render('Turnamens/Index', [
            'turnamens' => $user->turnamens()
            ->latest()->get()
        ]);
    }
    public function show(Turnamen $turnamen)
    {
        // Periksa apakah turnamen dimiliki oleh user yang
        // sekarang authenticated
        if ($turnamen->user()->get()->first()
        ->id !== Auth::user()->id) {
            return redirect(route('turnamens.index'));
        }
        $semuaPertandinganTerisiPesertaLengkap = $turnamen->
        cekKelengkapanPesertaPertandingan();
        return Inertia::render(
            'Turnamens/Show', [
                'turnamen' => $turnamen,
                'pertandingans' => $turnamen->pertandingans()
                ->get(),
                'isPesertaLengkap' =>
                $semuaPertandinganTerisiPesertaLengkap,
            ]
        );
    }
    public function create(): Response
    {
        return Inertia::render(
            'Turnamens/Create', []
        );
    }
    public function store(Request $request): RedirectResponse
    {
        // Lihat 0
    }
```

```
    public function export(Turnamen $turnamen, TurnamenService
$turnamenService)
    {
        // Lihat 0
    }
}
```

Potongan Kode 5.71. TurnamenController dan detail implementasi fungsi index(), show(), dan create()

Selanjutnya, fungsi store memiliki fungsi untuk mengirim menyalurkan data dari view formulir pembuatan turnamen ke *back end*. Sebelum disimpan ke basis data, data divalidasi terlebih dahulu. Implementasi fungsi ini tercantum pada 0.

```
<?php
namespace App\Http\Controllers;
// use directives
class TurnamenController extends Controller
{
    ...
    public function store(Request $request): RedirectResponse
    {
        logger("Belum validasi!");
        $validated = $request->validate([
            'nama' => 'required|string|max:255',
            'panitia' => 'required|string|max:255',
            'asalPanitia' => 'required|string|max:255',
            'logoKanan' => [
                'required', 'image', Rule::dimensions()
                ->maxWidth(1000) ->maxHeight(1000),
            ],
            'logoKiri' => [
                'required','image', Rule::dimensions()
                ->maxWidth(1000)->maxHeight(1000),
            ],
        ]);
        logger("Input validated!");
        // Mulai proses file jika file ada
        if ($request->hasFile('logoKanan') && $request->
        hasFile('logoKiri')) {
            $logoKanan = $request->file('logoKanan');
```

```
            // Store to storage and disk. [DO NOT DELETE]
            $logoKananName = str()->uuid() .
            '.' . $logoKanan->extension();
            $logoKanan = Storage::disk()->putFileAs(
                "logoKanan/{$request->user()->id}",
                $logoKanan, $logoKananName
            );
            $logoKiri = $request->file('logoKiri');
            // Store to storage and disk. [DO NOT DELETE]
            $logoKiriName = str()->uuid() .
            '.' . $logoKiri->extension();
            $logoKiri = Storage::disk()->putFileAs(
                "logoKiri/{$request->user()->id}",
                $logoKiri, $logoKiriName
            );
            logger("Logo uploaded!");
        } else {
            return redirect(
                route('turnamens.create')
            );
        }
        $turnamen = $request->user()->turnamens()->create([
            'nama' => $validated['nama'],
            'panitia' => $validated['panitia'],
            'asalPanitia' => $validated['asalPanitia'],
            'logoKanan' => $logoKanan ??= null,
            'logoKiri' => $logoKiri ??= null,
        ]);
        logger("Upload success!");
        return redirect(route('turnamens.index'));
    }
    ...
}
```

Potongan Kode 5.72. Fungsi store pada TurnamenController

Fungsi terakhir pada TurnamenController adalah fungsi export(). Fungsi ini berfungsi untuk memastikan proses menyimpan data turnamen menjadi lembar nilai turnamen dalam format .xlsx dapat dilakukan dengan benar. Hal ini dilakukan dengan memeriksa kepemilikan Turnamen oleh *user* dan

kelengkapan peserta. Fungsi utama untuk membuat lembar nilai turnamen ditangani oleh fungsi export() dari *service* TurnamenService, yang dipanggil setelah kepemilikan turnamen dan kelengkapan peserta diperiksa.

```
<?php
namespace App\Http\Controllers;
// use directives
class TurnamenController extends Controller
{
    ...
    public function export(Turnamen $turnamen, TurnamenService
$turnamenService)
    {
        // See if turnamen belongs to authenticated user or not
        if ($turnamen->user()->get()
        ->first()->id !== Auth::user()->id) {
            return redirect(route('turnamens.index'));
        }
        $semuaPertandinganTerisiPesertaLengkap = $turnamen
        ->cekKelengkapanPesertaPertandingan();
        if (!$semuaPertandinganTerisiPesertaLengkap) {
            return Inertia::render('Turnamens/Show', [
                'turnamen' => $turnamen,
                'pertandingans' => $turnamen->pertandingans()
                ->latest()->get(),
                'isPesertaLengkap' =>
                $semuaPertandinganTerisiPesertaLengkap,
            ]);
        }
        try {
            $turnamenService->export($turnamen);
        } catch (\Throwable $th) {
            throw $th;
        }
    }
}
```

Potongan Kode 5.73. Fungsi export pada TurnamenController

Method export() pada TurnamenController memanggil salah satu *method* dari *service class* TurnamenService.

Implementasi *service* ini dapat dilihat pada 0, 0, dan 0. Service ini utamanya menggunakan *library* PhpSpreadsheet untuk membuat *file* lembar nilai turnamen yang berformat .xlsx. 0 menunjukkan bagian pertama dari TurnamenService, yaitu fungsi export yang membuka *file* templat, mengubah data-data *placeholder*, dan menambahkan gambar logo kanan dan kiri.

```
<?php
namespace App\Services;
// use directives lainnya
use PhpOffice\PhpSpreadsheet\IOFactory;
use PhpOffice\PhpSpreadsheet\Worksheet\Worksheet;
use PhpOffice\PhpSpreadsheet\Writer\Xlsx;
use PhpOffice\PhpSpreadsheet\Style;
use PhpOffice\PhpSpreadsheet\Worksheet\Drawing;
class TurnamenService
{
    public function export(Turnamen $turnmnamen)
    {
        // Buka file dan sheet template
        $templateFilePath = "template/template_pingpong.xlsx";
        $reader = IOFactory::createReader('Xlsx');
        $spreadsheet = $reader->load($templateFilePath);
        // Ubah placeholder: panitia, desa, turnamen
        $pertandingans = $turnamen->pertandingans()->get();
        $templateWorksheet = $spreadsheet->getSheet(0);
        $templateWorksheet
            ->setCellValue('A2', $turnamen->panitia)
            ->setCellValue('A3', $turnamen->asalPanitia)
            ->setCellValue('A4', $turnamen->nama);
        // Tambah logo
        $this->tambahLogoKanandanLogoKiri(
            $templateWorksheet,
            Storage::disk('local')->path($turnamen->logoKiri),
            Storage::disk('local')->path($turnamen->logoKanan),
        );
        /* Menyalin data ke Spreadsheet */
        /* Hapus sheet templat, simpan ke file .xlsx dan unduh
           ke perangkat */
    }
    protected function tambahLogoKanandanLogoKiri(
```

```
        Worksheet $worksheet,
        string $pathLogoKiri,
        string $pathLogoKanan,
    ){ /* Implementasi fungsi */}
}
```

Potongan Kode 5. 74. TurnamenService

Setelah logo dimasukkan pada *worksheet* templat, data tiap pertandingan akan dimasukkan ke *file* tersebut dengan mengikuti format pada *worksheet* templat. Pertama, *worksheet* templat diduplikat. *Worksheet* duplikat ini diganti judulnya sesuai jenis pertandingan dan ditambahkan ke *file sheet*. Lalu, jenis pertandingan dimasukkan ke dalam *worksheet*. Variabel $tabels berfungsi untuk menampung tabel peserta yang sudah dikelompokkan berdasarkan nama tabel dan diurutkan berdasarkan nomor urut. Lalu, perulangan dilakukan pada variabel ini untuk menambahkan data tiap peserta ke dalam tabel dalam *worksheet* dengan *style* yang sesuai.

```
...
foreach ($pertandingans as $pertandingan) {
    // Duplikat template
    $clonedWorksheet = clone $spreadsheet->getSheet(0);
    $clonedWorksheet->setTitle($pertandingan
    ->jenisPertandingan);
    $spreadsheet->addSheet($clonedWorksheet);
    // Ganti nama pertandingan
    $clonedWorksheet->setCellValue('A7', $pertandingan
    ->jenisPertandingan);
    // Buat tabel pada Excel untuk tiap tabel peserta
    $tabels = $pertandingan->pesertas()->get()
        ->sortBy('kodeTabel')->sortBy('nomorUrut')
        ->groupBy('kodeTabel');

    $rowMulaiTabelPertama = 9;
    $offset = 0;
    $tinggiTabel = 6;
    $jarakAntaraTabel = 2;
```

```
    // Isi data pada tabel
    foreach ($tabels as $kodeTabel => $pesertas) {
        $rowMulai = $rowMulaiTabelPertama + $offset;

        $clonedWorksheet
        ->setCellValue('A'.$rowMulai,$kodeTabel);
        // Set frame table
        $frameTabel = $spreadsheet->getSheet(0)
        ->rangeToArray('A10:J14');
        $clonedWorksheet->fromArray(
            $frameTabel, null, 'A'.$rowMulai+1
        );
        // Set style
        $borderStyleArray = [
            'borders' => [
                'allBorders' => [
                    'borderStyle' => Style\Border::BORDER_THICK,
                ],
            ],
        ];

        $headerStyleArray = [
            'font' => ['bold' => true,'color' => [
                    'argb' => 'FFFFFFFF',],
            ],
            'alignment' => [
                'horizontal' =>
                Style\Alignment::HORIZONTAL_CENTER,
            ],
            'borders' => [
                'allBorders' => [
                    'borderStyle' => Style\Border::BORDER_THICK,
                ],
            ],
            'fill' => [
                'fillType' => Style\Fill::FILL_SOLID,
                'rotation' => 90,
                'startColor' => [
                    'argb' => 'FF980005',
                ],
            ],
```

```
        ];

        // Mengeset border
        $clonedWorksheet->getStyle(
            'A' . $rowMulai+1 . ':' . 'J' . $rowMulai+5
        )->applyFromArray($borderStyleArray);

        // Mengeset header
        $clonedWorksheet->getStyle(
            'A' . $rowMulai+1 . ':' . 'J' . $rowMulai+1
        )->applyFromArray($headerStyleArray);
        $clonedWorksheet->getStyle(
            'A' . $rowMulai+1 . ':' . 'A' . $rowMulai+5
        )->applyFromArray($headerStyleArray);

        $blockedColumn = 'D';
        foreach ($pesertas as $index => $peserta) {
            $rowPeserta = $rowMulai+2+$index;
            $clonedWorksheet->getRowDimension($rowPeserta)
            ->setRowHeight(15.5);

            $clonedWorksheet->setCellValue('B'.$rowPeserta,
            $peserta->nama);
            $clonedWorksheet->setCellValue('C'.$rowPeserta,
            $peserta->dari);

            $clonedWorksheet->getStyle($blockedColumn .
            $rowPeserta)->applyFromArray($headerStyleArray);
            $blockedColumn++;
        }
        $offset = $offset + $tinggiTabel + $jarakAntaraTabel;
    }
}
...
```

Potongan Kode 5.75. Proses memasukkan data ke file daftar nilai turnamen pada *method* export() TurnamenService

0 menunjukkan implementasi proses finalisasi penyimpanan data ke file .xlsx. Pertama, worksheet templat dihapus. Lalu, *file sheet* ditulis ke respon. Oleh karena $writer-

>save dikirim argumen 'php://output', file sheet akan tersimpan ke folder download perangkat.

```
...
// Hapus worksheet templat
$spreadsheet->removeSheetByIndex(0);
// Simpan ke file .xlsx dan unduh ke perangkat
$dateNow = date_create(
    'now', timezone_open('Asia/Makassar')
)->format('Ymd');
$fileName = "{$turnamen->nama}_{$turnamen->panitia}_{$dateNow}";
$response = response()
->streamDownload(function() use ($spreadsheet) {
    $writer = new Xlsx($spreadsheet);
    $writer->save('php://output');
});
$response->setStatusCode(200);
$response->headers->set('Content-Type',
'application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet; charset=sjis'
);
$response->headers->set(
    'Content-Disposition',
    'attachment; filename= "'.$fileName.'.xlsx" '
);
$response->send();
exit();
...
```

Potongan Kode 5.76. Bagian kode untuk finalisasi penyimpanan data ke dalam file .xlsx

Selain *method* export(), TurnamenService juga memiliki *method* tambahLogoKanandanLogoKiri(). Fungsi ini berfungsi untuk mengatur logo kanan dan kiri pada *worksheet*.

```
Protected function tambahLogoKanandanLogoKiri(
    Worksheet $worksheet,
    string $pathLogoKiri,
    string $pathLogoKanan,
)
{
```

```
    $height = 76;
    $logoKiri = new Drawing();
    $logoKiri->setName('Logo Daerah');
    $logoKiri->setDescription('Logo Daerah');
    $logoKiri->setPath($pathLogoKiri);
    $logoKiri->setHeight($height);
    $logoKiri->setCoordinates('A2');
    $logoKiri->setOffsetX(15);
    $logoKiri->setWorksheet($worksheet);

    $logoKanan = new Drawing();
    $logoKanan->setName('Logo Panitia');
    $logoKanan->setDescription('Logo Panitia');
    $logoKanan->setPath($pathLogoKanan);
    $logoKanan->setHeight($height);
    $logoKanan->setCoordinates('I2');
    $logoKanan->setOffsetX(-15);
    $logoKanan->setWorksheet($worksheet);
}
```

Potongan Kode 5.77. Method tambahLogoKanandanLogoKiri TurnamenService

Controller selanjutnya pada Sistem Turnamen Pingpong adalah PertandinganController yang dapat dilihat pada 0. *Controller* ini berisi tiga fungsi, yaitu index(), create(), dan store(). Fungsi index() me-*redirect user* ke *route* 'turnamen.show' karena daftar pertandingan tercantum pada halaman detail turnamen. Fungsi create() mengembalikan *render view* formulir penambahan pertandingan. Fungsi store() memvalidasi data yang dikirim dari halaman penambahan pertandingan dan menyimpannya ke basis data melalui model.

```
<?php
namespace App\Http\Controllers;
// use directives...

class PertandinganController extends Controller
{
    public function index(Turnamen $turnamen)
    {
```

```
        return redirect(route('turnamens.show', $turnamen));
    }
    public function create(Turnamen $turnamen)
    {
        // Periksa apakah turnamen dimiliki oleh user yang
        // sekarang authenticated
        if ($turnamen->user()->get()
        ->first()->id !== Auth::user()->id) {
            return redirect(route('turnamens.index'));
        }
        return Inertia::render(
            'Pertandingans/Create', ['turnamen' => $turnamen]
        );
    }
    public function store(Request $request, Turnamen $turnamen)
    {
        // Periksa apakah turnamen dimiliki oleh user yang
        // sekarang authenticated
        if ($turnamen->user()->get()
        ->first()->id !== Auth::user()->id) {
            return redirect(
                route('turnamens.index')
            );
        }
        $validated = $request->validate([
            'jenisPertandingan' => 'required|string|max:255',
            'jumlahPeserta' => [
                'required','int','max:255', new MultipleOf4
            ],
        ]);
        $pertandingan = $turnamen->pertandingans()->create([
            'jenisPertandingan' => $validated[
                'jenisPertandingan'],
            'jumlahPeserta' => $validated['jumlahPeserta'],
        ]);
        return redirect(route('turnamens.show', $turnamen));
    }
}
```

Potongan Kode 5.78. PertandinganController

Controller terakhir adalah PesertaController yang tercantum dalam 0. Sesuai namanya, *controller* ini mengatur manipulasi model Peserta. PesertaController terdiri dari tiga fungsi, yaitu index(), create(), dan store(). Fungsi index() berfungsi untuk menyalurkan data daftar peserta suatu pertandingan dari model dan basis data ke *view*. Fungsi create() akan menunjukkan *view* formulir pendaftaran peserta jika belum ada peserta yang terdaftar. Jika peserta sudah lengkap, controller ini akan mengarahkan *user* ke *view* daftar peserta pertandingan. Fungsi store() berfungsi untuk memvalidasi data peserta-peserta dan menyimpannya ke basis data. Oleh karena data yang dikirimkan dari *view* berbentuk *array* peserta, validasi data peserta dilakukan dengan menambahkan *wildcard character* (*) untuk memeriksa atribut setiap peserta.

```
<?php
namespace App\Http\Controllers;
// use directive
class PesertaController extends Controller
{
    public function index(
        Turnamen $turnamen, Pertandingan $pertandingan
    ){
        return Inertia::render('Pesertas/Index', [
            'pesertas' => $pertandingan->pesertas()
            ->orderBy('kodeTabel', 'asc')
            ->orderBy('nomorUrut', 'asc')
            ->get(),
            'pertandingan' => $pertandingan,
            'turnamen' => $turnamen,
        ]);
    }
    public function create(
        Turnamen $turnamen, Pertandingan $pertandingan
    ): Response | RedirectResponse
    {
        $pesertas = $pertandingan->pesertas()->get();
        if($pertandingan->jumlahPeserta === count($pesertas)) {
            return redirect(route(
                'turnamens.pertandingans.pesertas.index',
```

```
                [$turnamen,$pertandingan,]
            ));
        }
        return Inertia::render(
            'Pesertas/Create', [
                'pertandingan' => $pertandingan,
                'turnamen' => $turnamen,
            ]
        );
    }
    public function store(
        Turnamen $turnamen, Pertandingan $pertandingan,
        Request $request
    ){
        $jumlahPeserta = $pertandingan->jumlahPeserta;
        $validatedData = $request->validate([
            "pesertas"=>"required|array|size:$jumlahPeserta",
            "pesertas.*"=>"array|size:6",
            "pesertas.*.id"=>"required|int",
            "pesertas.*.nama"=>"required|string",
            "pesertas.*.dari"=>"required|string",
            "pesertas.*.kodeTabel"=>"required|string",
            "pesertas.*.nomorUrut"=>"required|int",
            "pesertas.*.kodeTabelDanNomorUrut"=>"string",
        ]);
        $pesertas = $validatedData['pesertas'];
        try {
            foreach ($pesertas as $peserta) {
                $newPeserta = $pertandingan->pesertas()
                ->create([
                    'nama' => $peserta['nama'],
                    'dari' => $peserta['dari'],
                    'kodeTabel' => $peserta['kodeTabel'],
                    'nomorUrut' => $peserta['nomorUrut'],
                ]);
            }
            return redirect(route(
                'turnamens.pertandingans.pesertas.index',
                 [$turnamen, $pertandingan]
            ));
        } catch (Exception $e) {
```

```
            logger("Cannot insert peserta. " . $e);
        }
    }
}
```

Potongan Kode 5.79. PesertaController

[Halaman ini sengaja dikosongkan]

BAB VI
PENGUJIAN DAN EVALUASI

Bab ini menjelaskan tahap pengujian terhadap Sistem Automasi Surat RfQ dan Sistem Turnamen Pingpong. Tahap ini memiliki tujuan untuk memeriksa kesesuaian fungsionalitas dan implementasi sistem dengan analisis dan rancangan arsitektur yang telah dibuat.

6.1. Tujuan Pengujian

Pengujian terhadap Sistem Automasi Surat RfQ bertujuan untuk memastikan fungsi-fungsi yang dapat dilakukan masing-masing *role* dapat berjalan sebagaimana mestinya dan integrasi dengan Google Sheets API tidak mengalami kendala. Lalu, pengujian pada Sistem Turnamen Ping Pong bertujuan untuk memastikan fungsi manajemen data turnamen pingpong berjalan dengan seharusnya, terutama bagian *input* peserta – *preview* tabel nilai peserta dan pembuatan *file* Excel dengan PHPSpreadsheet.

6.2. Kriteria Pengujian

Penilaian atas pencapaian tujuan pengujian didapatkan dengan memperhatikan beberapa hasil yang diharapkan berikut.

6.2.1. Sistem Automasi Surat RfQ

a. Kemampuan aplikasi untuk menampilkan daftar RfQ yang diambil dari Google Sheets *tracking* kepada *role* vendor.
b. Kemampuan aplikasi untuk menerima unggahan *file* daftar penawaran dan menyimpannya dalam basis data.
c. Kemampuan aplikasi untuk menampilkan daftar penawaran yang diunggah oleh vendor ke vendor bersangkutan dan admin.
d. Kemampuan aplikasi untuk menjalankan proses *approval* oleh *role* admin, yang termasuk memperbarui status

penawaran pada basis data dan memroses *file* penawaran RfQ menjadi surat RfQ dalam format Google Sheets.

6.2.2. Sistem Turnamenn Pingpong

a. Kemampuan arsitektur untuk menyimpan dan mengelola data turnamen, yaitu detail turnamen, detail pertandingan, dan data peserta.
b. Kemampuan aplikasi untuk menampilkan *preview* dari daftar peserta yang akan didaftarkan ke pertandingan dan membuat lembar nilai dari data tersebut.

6.3. Skenario Pengujian

Skenario pengujian dilakukan dengan melakukan peran sebagai *user* yang akan menjalankan fitur-fitur. Langkah-langkah untuk setiap kebutuhan fungsionalitas yaitu sebagai berikut.

Skenario pengujian dilakukan melalui perspektif pengguna yang akan menggunakan sistem dan fitur-fiturnya. Tahap-tahap yang dilakukan untuk setiap kebutuhan fungsionalitas dipaparkan pada subbab-subbab berikut.

6.3.1. Sistem Automasi Surat RfQ

1. Kemampuan aplikasi untuk menampilkan daftar RfQ yang diambil dari Google Sheets *tracking* kepada *role* vendor.
2. Kemampuan aplikasi untuk menerima unggahan *file* daftar penawaran dan menyimpannya dalam basis data.
3. Kemampuan aplikasi untuk menampilkan daftar penawaran yang diunggah oleh vendor kepada vendor bersangkutan dan admin.
4. Kemampuan aplikasi untuk menjalankan proses *approval* oleh *role* admin, yang termasuk memperbarui status penawaran pada basis data dan memroses *file* penawaran RfQ menjadi surat RfQ dalam format Google Sheets.

6.3.2. Sistem Turnamen Pingpong

1. Kemampuan arsitektur untuk menyimpan dan mengelola data turnamen, yaitu detail turnamen, detail pertandingan, dan data peserta.

2. Kemampuan aplikasi untuk menampilkan *preview* dari daftar peserta yang akan didaftarkan ke pertandingan dan membuat lembar nilai dari data tersebut.

6.4. Evaluasi Pengujian

Hasil pengujian didapatkan dari pengamatan perilaku sistem aplikasi saat dilakukan skenario uji coba. Tabel 6.1 menjelaskan hasil uji coba terhadap Sistem Automasi Surat RfQ.

Tabel 6.1. Hasil Evaluasi Pengujian untuk Sistem Automasi Surat RfQ

Kriteria Pengujian	**Hasil Pengujian**
Pengguna dengan *role* vendor dapat melihat data daftar RfQ yang berasal dari Google Sheets *tracking*	Berhasil memroses data dari Google Sheets menjadi data dalam tabel HTML (Terpenuhi)
Pengguna dapat mengunggah *file* daftar penawaran	Berhasil menerima daftar penawaran melalui bagian *upload* dan *update* penawaran untuk *role* vendor dan bagian *reupload* penawaran gagal untuk *role* admin. *File* penawaran tersimpan di server dan datanya tercatat di basis data. (Terpenuhi)
Pengguna dengan *role* vendor dapat melihat penawaran yang ia ajukan sendiri dan pengguna dengan *role* admin dapat melihat semua	Berhasil menampilkan ajuan penawaran kepada *role* vendor dan admin sesuai batasannya. (Terpenuhi)

penawaran yang diajukan semua vendor	
Pengguna dengan *role* admin dapat memberikan *approval* terhadap penawaran yang diajukan	Berhasil memeriksa data, membuat surat RfQ dalam format Google Sheets dengan menggunakan fitur *background jobs* dan *queue*, dan menyimpan histori perubahan ke basis data. Berhasil pula mengirimkan notifikasi ke pengguna jika pembuatan surat sukses atau tidak. (Terpenuhi)

Tabel 6.2 menjelaskan hasil uji coba terhadap Sistem Turnamen Pingpong.

Tabel 6.2. Hasil Evaluasi Pengujian untuk Sistem Turnamen Pingpong

Kriteria Pengujian	**Hasil Pengujian**
Pengguna dapat membuat turnamen baru dan menambahkan pertandingan dan peserta ke dalamnya.	Berhasil mengambil data dari pengguna melalui halaman tambah data masing-masing entitas dan menyimpan data tersebut ke basis data. (Terpenuhi)
Pengguna dapat melihat *preview* dari lembar nilai pertandingan saat mengisi formulir pendaftaran peserta	Berhasil menampilkan tabel serupa daftar nilai saat pengguna melakukan *input* data peserta pada halaman

	Tambah Peserta (Terpenuhi)
Pengguna dapat mengunduh *file* lembar nilai untuk turnamen yang sudah lengkap pesertanya	Berhasil mengolah data menjadi daftar nilai berformat XLSX dengan menggunakan *library* PHPSpreadsheet dan dapat diunduh pengguna melalui tombol "Simpan ke *file* XLSX" pada halaman Detail Turnamen. (Terpenuhi)

[Halaman ini sengaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

Kesimpulan yang didapatkan penulis dari perancangan dan implementasi Sistem Automasi Surat RfQ dan Sistem Turnamen Pingpong pada kegiatan kerja praktik di PT. Max Solusindo adalah sebagai berikut:

a. Aplikasi Sistem Automasi Surat RfQ dibangun menggunakan *framework* Laravel dan Blade *templating engine*. Integrasi Google Sheets *tracking* ke dalam sistem ini dan pembuatan surat RfQ yang terautomasi dengan format Google Sheets dibantu dengan Google Workspace REST API.
b. Aplikasi Sistem Turnamen Pingpong dibangun menggunakan *framework* Laravel sebagai *framework* utama dan Vue.js sebagai *framework* tambahan untuk bagian *front-end*. Fitur *preview* daftar nilai secara khusus dibangun dengan memanfaatkan teknik pemrograman *declarative* Vue.js dan fungsi *export* daftar nilai ke *file* XLSX dilakukan dengan bantuan modul PHPSpreadsheet.

7.2. Saran

Saran yang penulis berikan untuk Sistem Automasi Surat RfQ dan Sistem Turnamen Pingpong adalah sebagai berikut:

a. Pada Sistem Automasi Surat RfQ, penggunaan *framework front-end* lainnya dapat dipertimbangkan untuk mempermudah *maintenance* dan sistem notifikasi dapat dikembangkan menjadi dapat ditandai sebagai terbaca.
b. Pada Sistem Turnamen Pingpong, fitur *bracketing* dan fitur sejenis yang membantu kemudahan pelaksanaan turnamen dapat ditambahkan.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

Chun, W. (2023, May 2). *Getting started using Google APIs: Workspace & OAuth client IDs (1/3)*. Dev.To. https://dev.to/googleworkspace/getting-started-using-google-apis-workspace-9a8

Dominte, I. (2023). Web API Development for the Absolute Beginner: A Step-by-step Approach to Learning the Fundamentals of Web API Development with .NET 7. In *Web API Development for the Absolute Beginner*. Apress. https://doi.org/10.1007/978-1-4842-9348-5

Engebreth, G., & Sahu, S. K. (2023). PHP 8 Basics. In *PHP 8 Basics*. Apress. https://doi.org/10.1007/978-1-4842-8082-9

Google for Developers. (n.d.). *API Client Libraries*. Google. Retrieved November 25, 2024, from https://developers.google.com/api-client-library

Jain, J. (2022). Learn API Testing: Norms, Practices, and Guidelines for Building Effective Test Automation. In *Learn API Testing: Norms, Practices, and Guidelines for Building Effective Test Automation*. Apress Media LLC. https://doi.org/10.1007/978-1-4842-8142-0

Ravi Kumar, Y. V, Samayam, A. K., & Miryala, N. K. (2024). Mastering MySQL Administration. In *Mastering MySQL Administration*. Apress. https://doi.org/10.1007/979-8-8688-0252-2

Yang, H. (2023). *Vue. JS Framework Design and Implementation*.

Zammetti, F. (2022). Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, Python, Django, and Docker, Second Edition. In *Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack,*

Python, Django, and Docker, Second Edition. Apress Media LLC. https://doi.org/10.1007/978-1-4842-8811-5

BIODATA PENULIS

Nama	: Arya Widia Putra
Tempat, Tanggal Lahir	: Denpasar, 27 Maret 2002
Jenis Kelamin	: Laki-laki
Telepon	: +6281236033096
Email	: 5025201016@student.its.ac.id

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2020
Semester	: 9 (Sembilan)