UTILISASI PENGOLAHAN PEMROSESAN DATA UNTUK MENINGKATKAN PERFORMA APLIKASI

by James Murphy

Submission date: 24-Nov-2020 04:13PM (UTC+0700)
Submission ID: 1455024645
File name: DRAFT_PAPER.pdf (1.18M)
Word count: 3821
Character count: 25292

UTILISASI PENGOLAHAN PEMROSESAN DATA UNTUK MENINGKATKAN PERFORMA APLIKASI

Abstrak*—Ada beberapa masalah yang terjadi selama proses pengembangan modul LIQUID *(Leadership Quality Feedback)* pada aplikasi KOMANDO (Komunikasi Manajemen dan Budaya Organisasi) dan setelah kemudian digunakan oleh klien *(production)*. Masalah yang ditemui meliputi *bug, issue, error*, dan performa aplikasi. Lambatnya performa aplikasi yang terjadi pada beberapa fitur yang ada pada modul LIQUID tentunya sangat menjadi perhatian. Selain dirasakan oleh tim pengembang saat proses pengembangan, hal yang sama juga dirasakan oleh klien sesaat setelah modul tersebut mulai digunakan. Tentunya hal tersebut sangat menghambat jalannya kegiatan LIQUID yang sedang berlangsung pada aplikasi KOMANDO. Komplain klien terkait masalah tersebut kemudian menjadi perhatian oleh tim pengembang untuk dapat dilakukan optimasi. Menurunnya performa aplikasi pada beberapa fitur dalam modul LIQUID terjadi karena kondisi tertentu. Kuantitas data yang banyak disertai relasi data terkait dan pengolahan data yang kompleks hingga menjadi informasi yang siap ditampilkan akan berpengaruh terhadap cepat atau lambatnya performa aplikasi. Secara teknis, teknik yang digunakan untuk memuat dan mengolah data tersebut kurang tepat dan tidak efektif. Optimasi data yang dilakukan untuk meningkatkan performa apliksi agar dapat mengatasi sejumlah kondisi adalah dengan utilisasi pada pengolahan dan pemrosesan data. Utilisasi dilakukan agar teknik yang dilakukan untuk memuat dan mengolah data menjadi efektif di segala bentuk kondisi. Beberapa teknik utilisasi dilakukan pada aplikasi KOMANDO, meliputi teknik *server-side processing* pada penggunaan Datatable, mengimplementasikan *lazy eager loading* untuk memuat relasi data terkait, dan menggunakan *view table*. Utilisasi yang dilakukan saling berkaitan satu sama lain dan memiliki peran fungsinya masing-masing sesuai dengan kebutuhan.

Keywords—server-side processing; datatable; lazy eager loading; view table;

I. PENDAHULUAN

Aplikasi KOMANDO adalah kepanjangan dari Komunikasi Manajemen dan Budaya Organisasi merupakan aplikasi yang dikembangkan sebagai layanan atau sarana untuk mengorganisisr pedoman perilaku. KOMANDO merupakan aplikasi internal PLN, yang telah dikembangkan dan digunakan dari sejak tahun 2017 lalu. Pada tahun 2020, dikembangkan sebuah modul baru bernama LIQUID. Modul LIQUID adalah kepanjangan dari *Leadership Quality Feedback* adalah merupakan sarana yang mendukung kegiatan umpan balik karyawan terhadap atasannya untuk mengetahui kualitas kepemimpinan serta harapan dan saran. Tujuan dikembangkan modul tersebut adalah untuk memfasilitasi kegiatan umpan balik yang dilakukan rutin secara periodik, dimana pelaksanaan sebelumnya dilakukan secara konvensional, menjadi berbasis *digital*. Dengan adanya fasilitas aplikasi, kegiatan penilaian umpan balik kelebihan menjadi lebih cepat, praktis, mudah, dan efisien.

Untuk mencapai tujuan dikembangkannya modul tersebut agar lebih mudah, cepat, praktis, dan efisien maka banyak hal yang harus diperhatikan dalam proses pengembangan. Selain hal teknis seperti apakah seluruh fungsionalitas sudah berjalan dengan baik sesuai dengan proses bisnis yang semestinya tanpa adanya kesalahan *logic* maupun sistem. Performa aplikasi ikut menjadi pertimbangan karena dapat menghambat proses kegiatan umpan balik karyawan yang sedang berlangsung.

Mayoritas pengembang hanya terpaku pada masalah teknis yang terjadi pada aplikasi ketika masa pengembangan maupun setelah aplikasi tersebut *rilis (production)* [1]. Selama tidak ada kesalahan *logic* maupun sistem dan fungsionalitas aplikasi berjalan sesuai dengan proses bisnis yang semestinya, maka tidak ada tindakan lebih lanjut yang perlu dilakukan. Namun kenyataannya, kecepatan performa sebuah aplikasi menjadi hal yang krusial. Walaupun masalah performa tidak secara langsung berdampak terhadap fungsionalitas aplikasi, namun akan menghambat jalannya proses bisnis yang berlangsung dan mempengaruhi kenyamanan pengguna dalam menggunakan aplikasi tersebut.

Tertulis pada referensi [2], laporan yang ditulis oleh *eMarketer* (November 1998) mengatakan bahwa seorang pengguna akan meninggalkan suatu halaman ketika halaman tersebut terlalu lama untuk dimuat. Sebanyak 51% pengguna tidak akan menunggu sebuah halaman untuk dimuat dengan waktu lebih dari 15 detik lamanya. Sedangkan *Zona Research Group* melaporkan bahwa waktu kepergian pengguna untuk menunggu suatu halaman dimuat meningkat menjadi 7 hingga 8 detik lamanya. Ketika telah mencapai waktu tersebut sebuah halaman belum dapat dimuat, maka seorang pengguna akan beralih pada halaman maupun aplikasi *website* lainnya.

Banyak faktor yang dapat mempengaruhi kecepatan sebuah performa aplikasi. Dalam hal ini, faktor yang mempengaruhi beberapa performa fungsionalitas yang ada pada modul LIQUID aplikasi KOMANDO adalah data. PLN merupakan perusahaan besar milik negara, yang mempunyai masing-masing distrik atau unit tersebar di seluruh Indonesia. Semakin besar sebuah perusahaan maupun instansi, maka akan semakin banyak kuantitas data yang dimiliki. Menggunakan teknik yang kurang tepat untuk memuat data dengan kuantitas yang banyak dan proses pengolahan yang kompleks akan berdampak pada performa aplikais tersebut. Beberapa teknik optimasi kecepatan performa aplikasi dilakukan untuk mengatasi dan menyesuaikan jumlah data yang sudah tersedia dan jumlah data dalam kurun waktu yang akan dating [3].

II. DASAR TEORI

A. *Datatable Serverside Processing*

Datatable merupakan *plugin* JQuery yang bersifat *open source*. Datatable mengkonversi tampilan *table* html menjadi format dengan tampilan datatable dengan antarmuka yang intuitif. *Plugin* datatable menyediakan fitur penting yang mendukung penyajian data. Fitur tersebut meliputi pencarian data dengan menggunakan *keyword,* paginasi, filter jumlah baris data yang ditampilkan per halaman, *sorting*, dan masih banyak fitur lain yang dapat digunakan. Proses penyajian data yang dilakukan dengan datatable dapat menggunakan teknik

pemrosesan klien *(client-side processing)* maupun pemrosesan dari sisi *server (server-side processing)*.

Penyajian data menggunakan *server-side processing* merupakan teknik yang dilakukan pada fungsionalitas modul LIQUID pada aplikasi KOMANDO, yang digunakan untuk meningkatkan performa aplikasi. Ditemui beberapa masalah performa aplikasi sebelumnya ketika menggunakan teknik pemrosesan klien. Penyajian data dengan menggunakan teknik *client-side processing* akan menyebabkan beban *client (browser)* menjadi lebih berat, karena semua prosesnya bertumpu pada *client*. Teknik *client-side processing* tidak cocok digunakan untuk menyajikan data dengan kuantitas yang banyak dan pengolahan yang kompleks. Dengan menggunakan teknik *serverside processing*, proses pengolahan data akan dilakukan oleh *server* sehingga beban *client* untuk menampilkan halaman tidak terlalu berat. *Request* terhadap *server* dilakukan secara berkala agar tidak langsung memuat semua data pada saat pertama kali memuat sebuah halaman.

Utilisasi yang dilakukan dengan menggunakan teknik *server-side processing* dibuktikan dengan hasil yang efektif. Dalam referensi penelitian [4] membuktikan bahwa penyajian data dengan kuantitas yang banyak dengan menggunakan teknik *serverside processing* 98,6% lebih cepat dibandingkan dengan teknik *client-side processing*. Pengujian *performance test* yang dilakukan pada penelitiannya untuk membuktikan angka perbandingan tersebut dilakukan dengan menggunakan JMeter sebanyak 50 kali. Dari *performance test* yang dilakukan, didapatkan hasil rata-rata 62651,02 ms atau sekitar 62 deetik waktu yang diperlukan untuk memuat data dan 2207860 *bytes* atau sekitar 2 *Megabyte* besarnya *memory* yang digunakan pada *client-side processing*. Sedangkan rata-rata yang dihasilkan pada *server-side processing* 960,28 ms atau sekitar kurang dari satu detik waktu yang digunakan untuk memuat data dan 8610 *bytes* atau sekitar kurang dari satu *Megabyte* besarnya *memory* yang digunakan.

B. *Software Development Framework*

Software Development Framework adalah sebuah struktur kerangka kerja yang digunakan untuk mengembangkan berbagai aplikasi, meliputi aplikasi desktop, aplikasi berbasis *website,* maupun aplikasi *mobile. Software development framework* diciptakan untuk memudahkan para pengembang dalam mengembangkan sebuah perangkat lunak dengan menuliskan kode sumber yang lebih konsisten. Banyak keunggulan dengan menggunakan *software development framework* dalam mengembangkan sebuah aplikasi. Berikut merupakan keunggulan yang didapatkan menggunakan *software development framework* dalam mengembangkan sebuah perangkat lunak.

1) Struktur Kode Sumber Aplikasi yang Tersusun: Dengan menggunakan *software development framework*, terdapat segenap aturan bagaimana struktur masing-masing file pada sebuah kode sumber harus dibuat. Bahkan *framework* dengan model MVC *(Model View Controller)* mempunyai struktur yang baik, di mana antara file logic dari sebuah aplikasi terpisah dengan *file view* yang memuat bentuk tampilan dari sebuah aplikasi.

2) Konsistensi Penulisan Kode: Adanya aturan dalam *software development framework* bagaimana sebuah kode harus dituliskan, mengharuskan para pengembang menuliskan kode secara konsisten. Jika kode yang dituliskan tidak sesuai dengan aturan tersebut, maka akan terjadi *error* pada aplikasi.

3) Praktis: *Software development framework* telah menyediakan kode berupa *function, method, property*, maupun *class* yang dapat langsung digunakan tanpa harus mendefinisikan dari awal. Seorang pengembang tidak perlu mendefinisikan susunan *folder* pada kode sumber dari awal, karena telah disediakan oleh beberapa *framework* pada saat proses instalasi.

4) All in One Documentation: Dokumentasi teknis sebuah *software development framework* telah disediakan oleh penciptanya, dicantumkan pada halaman resmi *framework* tersebut dengan tampilan yang menarik sehingga memudahkan para pengembang penggunanya dapat membaca dengan mudah.

Pengembangan aplikasi KOMANDO dilakukan dengan menggunakan *framework* Laravel. Laravel adalah salah satu *framework open source* yang menggunakan Bahasa pemrogaman PHP Hypertext Processor. Laravel pertama kali rilis publik versi 1 pada tanggal 9 Juni tahun 2011 oleh Taylor Otwell. Laravel diciptakan karena rasa ketidak puasan dari seorang Taylor Otwell, yang menggunakan salah satu *framework* PHP yaitu CodeIgniter. Banyak keuntungan yang didapat oleh para pengembang ketika menggunakan *framework* Laravel. Berikut merupakan keuntungan apa saja ketika seorang pengembang menggunakan Laravel dalam pengembangan aplikasi berbasis *website* [5].

1) Convention Over Configuration: software design pattern yang meminimalkan jumlah keputusan seorang pengembang dengan menyediakan konfigurasi secara default. Seorang pengembang tidak perlu lagi melakukan beberapa konfigurasi awal. Semua kebutuhan konfigurasi tersebut telah secara otomatis diinisiasi oleh Laravel, tidak membatasi untuk seorang pengembang merubah konfigurasi tersebut sesuai dengan kebutuhan.

2) Ready Out of The Box: Hanya dibutuhkan beberapa detik saja setelah proses instalasi selesai dilakukan, dengan menggunakan baris perintah yang sudah disediakan, aplikasi bawaan Laravel sudah dapat dimuat pada *browser*. Seorang pengembang dapat langsung menuliskan baris kode aplikasi yang akan dikembangkan tanpa harus menuliskan seluruh konfigurasi secara manual dari awal.

3) Clear and Organized All Parts of The Application: Dengan menggunakan Laravel, struktur folder di dalamnya tersusun dan terorganisir dengan baik. Semua komponen penyusunnya mempunyai tempat masing-masing. Hal tersebut karena Laravel menggunakan konsep MVC *(Model, View, Controller) pattern*.

4) Built-in Authentication: Dengan menggunakan Laravel hanya dengan menjalankan baris perintah yang sudah disediakan, sistem autentikasi secara otomatis terimplementasikan pada aplikasi. Seorang pengembang dapat memodifikasi sesuai dengan kebutuhan, jika dirasa sistem autentikasi Laravel kurang sesuai.

5) Eloquent ORM: Interaksi data dengan basisdata yang dilakukan secara manual, akan terlihat kacau di mana kode SQL yang saling bercampung dengan bahasa pemrogaman PHP. Sangat tidak nyaman ketika interaksi dilakukan dengan

data yang saling berkaitan satu sama lainnya. Hal tersebut tentunya akan menjadi sangat mudah dan sedehana ketika hanya dengan menggunakan *class object* beserta propertinya, seorang pengembang dapat berinteraksi dengan data tanpa harus menuliskan kode SQL secara utuh. Teknik modifikasi data pada basisdata menjadi objek dinamakan *Active Record pattern*, dimana Laravel telah menggunakan teknik tersebut pada fitur Eloquent ORM.

C. Eloquent ORM

Laravel sangat memudahkan para pengembangnya dalam menuliskan *query*, adalah dengan adanya fitur Eloquent ORM *(Object Relational Mapping)*. ORM sangat memudahkan sebuah kode sumber untuk berkomunikasi dengan basis data. ORM adalah sebuah teknik pemrogaman yang digunakan untuk mengkonversi data dari sebuah basis data dengan menggunakan bahasa pemrogaman beriorentasi objek atau biasa disebut OOP *(Object Oriented Programming)*. Fungsi utama ORM adalah untuk menjembatani kedua sistem yang berbeda, sisi pemrogaman aplikasi dengan sistem basisdata relasional.

Manfaat dari menggunakan fitur Eloquent ORM adalah mempercepat proses pengembangan perangkat lunak, memudahkan definisi relasi antar *table* dengan orientasi objek pada kode sumber, seorang pengembang tidak perlu menuliskan *query* SQL secara manual, *syntax* Eloquent yang mudah dibaca dan dimengerti atau *human readable*. Selain Eloquent, Laravel juga menyediakan fitur Query Builder yang digunakan untuk melakukan proses *query* yang lebih kompleks. Kedua fitur tersebut merupakan fitur unggulan dari *framework* Laravel yang digemari para pengembang.

Eloquent ORM mempunyai banyak fitur yang telah disediakan di dalamnya *(built in)*. Fitur yang disediakan mempunyai fungsi yang berbeda beda. Fitur yang ada pada Eloquent ORM meliputi Lazy Loading, Eager Loading, dan Lazy Eager Loading.

D. Database View Table

View table bukan merupakan sebuah fitur dari Laravel, melainkan bagian dari pada sebuah database relasional. Untuk membuat sebuah *view table,* diperlukan sebuah *query* SQL *(Structured Query Languange)*. SQL sendiri adalah sebuah Bahasa pemrogaman yang digunakan untuk berkomunikasi dengan basis data relasional, baik untuk menjalankan perintah manipulasi atau mengakses data terhadap database terkait. Pada umumnya sebuah DBMS *(Database Management System)* familiar dengan bahasa SQL tersebut, hampir semua basis data relasional memiliki perintah yang sama. SQL sendiri memiliki beberapa kategori, dikelompokkan sesuai dengan jenis perintahnya. Berikut merupakan kategori perintah pada SQL.

1) DDL *(Data Definition Language):* DDL merupakan perintah paling mendasar dalam Bahasa SQL. DDL dikelompokkan sesuai dengan fungsi dan tujuan yaitu mendifinisikan dan membuat struktur dari sebuah basis data relasional. Perintah dengan kategori DDL meliputi *CREATE, ALTER, RENAME, DROP, SHOW.*

2) DML *(Data Manipulation Language):* Sesuai dengan nama kategorinya, Data Manipulation Language mengelompokkan perintah SQL dengan tujuan untuk memanipulasi atau pengolahan record data dari sebuah basis data relasional. Perintah yang tergolong ke dalam kategori DML meliputi *INSERT, SELECT, UPDATE, DELETE.*

3) DCL *(Data Control Language):* Berbeda dengan kedua kategori sebelumnya yang sudah dijelaskan pada poin-poin sebelumnya. DCL bertujuan untuk manipulasi dan mengatur hak akses apa saja yang dimiliki oleh pengguna untuk melakukan komunikasi terhadap basis data relasional. Baik hak terhadap sebuah entitas database maupun table dan field yang ada. Perintah ini berfungsi untuk menjaga keamanan record data pada sebuah basis data relasional. Perintah yang tergolong dalam kategori DCL meliputi *GRANT, REVOKE*.

View table sendiri adalah bagian dari sebuah DBMS, *view table* dapat didefinisikan sebagai *table* virtual pada sebuah entitas *database. View table* memuat *record* data yang telah diolah dari sebuah *table* tertentu maupun kumpulan dari beberapa *table.* Keunggulan menggunakan *view table* adalah perintah *query* yang tersimpan, tidak perlu menulis ulang sebuah *query* dengan kebutuhan data yang sama, *indexing* pada *view table* dapat mempercepat performa, dan masih banyak keunggulan lainnya. Tujuan dari *view table* sendiri adalah meningkatkan keamanan data karena data yang dimuat terisolasi dengan *table* terkait, mempermudah pengolahan data, dan dalam kasus kasus tertentu *view table* dapat mempercepat dalam proses menampilkan data.

Dalam mengembangkan modul LIQUID pada aplikasi KOMANDO utilisasi pengolahan data juga dilakukan dengan menggunakan *view table.* Menggunakan *view table* untuk merepresentasikan data pada sebuah halaman adalah salah satu cara yang sangat efektif. Dalam referensi [6] terkait optimasi *query* pada sistem informasi penjadwalan mata pelajaran sekolah menggunakan *view table* terbukti efektif. Hasil dalam penelitian tersebut membuktikan menggunakan *view table* untuk memuat semua data jadwal pelajaran berdasarkan kelas memiliki waktu yang lebih singkat dibanding dengan menggunakan teknik *join* antar *table.* Selain itu keunggulan menggunakan *view table* untuk merepresentasikan data adalah mempersingkat penulisan *syntax* pada kode sumber aplikasi.

Dengan menggunakan *view table,* tidak diperlukan lagi proses pengolahan data yang kompleks menjadi sebuah informasi pada kode sumber aplikasi. Data yang dimuat dalam *view table* adalah hasil olahan dengan menggunakan *query* dari berbagai *table* sesuai dengan kebutuhan menjadi sebuah informasi yang siap ditampilkan pada halaman tertentu. Tidak adanya pengolahan data pada kode sumber aplikasi, menyebabkan kerja sebuah *server* untuk memuat sebuah data menjadi lebih ringan sehingga performa aplikasi menjadi lebih optimal.

III. METODOLOGI

Masalah yang muncul pada aplikasi dapat diketahui dari hasil uji coba fungsionalitas pada aplikasi tersebut. Uji coba fungsionalitas dilakukan oleh seorang *tester* maupun seorang sistem analis. Sebuah fungsionalitas akan lolos uji coba ketika memenuhi beberapa kondisi seperti tidak ada kesalahan *logic,* sistem *error,* dan proses bisnis berjalan sesuai dengan skenario saat dilakukan uji coba. Ketika ditemukan sebuah *issue* ketika uji coba fungsionalitas, seorang *tester* maupun sistem analis segera mungkin membuat *task* terhadap *programmer* untuk dilakukan pembenahan.

Berbeda dengan masalah terkait performa aplikasi yang terjadi di beberapa fungsionalitas. Uji coba pada beberapa fungsionalitas tidak menunjukkan adanya kesalahan *logic* maupun sistem *error* dan skenario berjalan sesuai dengan proses bisnis semestinya. Namun, halaman tersebut dimuat dengan waktu yang sangat lama dengan kondisi tertentu saja seperti filter *unit* dengan cakupan data yang sangat banyak. Utilisasi yang dilakukan pada beberapa fungsionalitas melalui tahap diskusi dengan *senior engineer*. Analisa kebutuhan yang akan menentukan teknik utilisasi seperti apa yang cocok diterapkan pada fungsionalitas tersebut. Implementasi teknik optimasi terhadap fungsionalitas tersebut. Baru kemudian dilakukan uji coba kembali apakah optimasi yang dilakukan efektif untuk meningkatkan performa fungsionalitas aplikasi.

IV. PEMBAHASAN DAN HASIL

A. Pembahasan

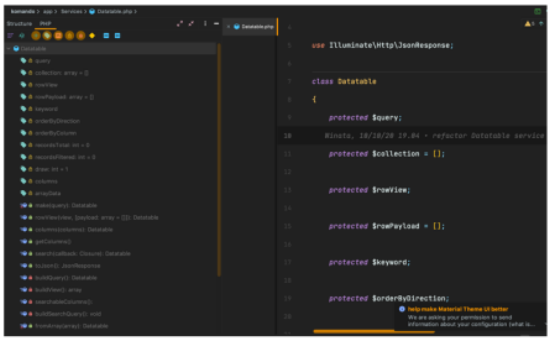
Berbagai macam teknik yang dilakukan dalam utilisasi pengolahan dan pemrosesan data untuk meningkatkan performa aplikasi, pada beberapa fungsioinalitas LIQUID KOMANDO. Berikut merupakan teknik utilisasi data yang dilakukan.

- Datatable *Server-side Processing,* dilakukan pada fungsionalitas yang menggunakan tampilan berupa *table* seperti fungsionalitas LIQUID *report*. Berfungsi agar semua proses tidak bertumpu pada klien saat memuat halaman.
- *Eager Loading; Lazy Eager Loading,* teknik yang digunakan untuk memuat data dengan relasi terhadap model data terkait sebelum dilakukan proses pengolahan data menjadi sebuah informasi.
- *View Table,* digunakan untuk meyimpan data yang telah diolah menggunakan *query*, agar tidak diperlukan pengolahan yang kompleks saat memuat data tersebut.

Ketiga teknik utilisasi mempunyai caranya masing masing untuk dapat diimplementasikan. Berikut merupakan implementasi yang yang dilakukan dari ketiga teknik utilisasi data.

1) Datatable *Serverside Processing*

Datatable *server-side processing* diimplementasikan di beberapa fungsionalitas terutama pada kebanyakan fungsi LIQUID *report*. Untuk memudahkan dalam implementasi teknik *server-side processing* yang digunakan secara berulang kali pada beberapa fungsionalitas, dibuatkan sebuah *service class* Datatable. *Service class* Datatable berisikan *property dan method* untuk memudahkan penggunaan teknik *server-side processing.* Gambar 1 merupakan gambar kerangka *service class* Datatable yang berikan *property* dan *method* didalamnya.



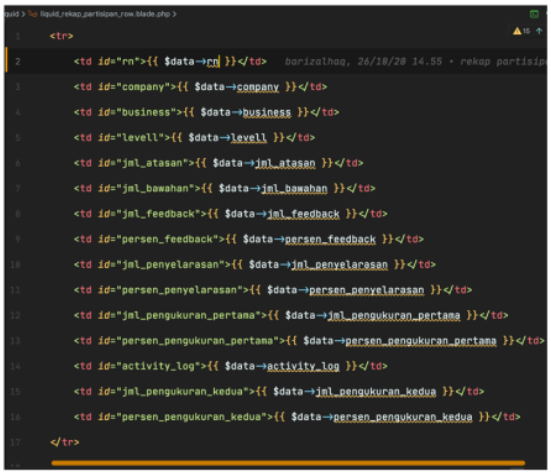
Gambar 1 Kerangka Datatable *service class*

Untuk dapat mengimplementasikan *server-side processing* yang dilakukan hanya melakukan inisiasi terhadap *service class* Datatable tersebut. Inisiasi *service class* Datatable membutuhkan beberapa parameter seperti *query* yang digunakan untuk memuat data, *row views*, dan defenisi kolom. Gambar 2 merupakan cara untuk melakukan inisiasi *service class* Datatable. *Row view* merupakan *view file* Laravel berekstensi blade, yang berisi definisi data yang akan ditampilkan per barisnya. Gambar 3 merupakan tangkapan laya berupa *row* view yang mendifinisikan sebuah data yang akan ditampilkan. Dengan menggunakan *service class* Datatable, untuk mengatasi fitur yang ada pada datatable berupa *searching, sorting,* dan paginasi hanya

Gambar 2 Inisiasi *service class* Datatable

dengan memanggil metode yang sudah disediakan pada *service class* Datatable saja.

Pada modul LIQUID aplikasi KOMANDO teknik utilisasi *server-side processing* digunakan untuk menampilkan sebuah informasi berupa tampilan *table* menggunakan datatable. Tujuan dari teknik tersebut adalah mengurangi beban klien *(browser)* saat memuat sebuah halaman. Halaman akan terlebih dahulu dimuat oleh *browser* tidak bersamaan dengan data, sehingga proses yang dilakukan menjadi lebih ringan. Kemudian setelah semua selesai dimuat barulah *browser* akan melakukan AJAX *request* untuk memuat data dan ditampilkan pada *table* tersebut. *Response* yang dikembalikan oleh *server* adalah



Gambar 3 Tampilan *row view*, definisi data yang akan ditampilkan

berupa format JSON *(Javascript Object Notation)* sehingga kemudian informasi dapat ditampilkan pada datatable.

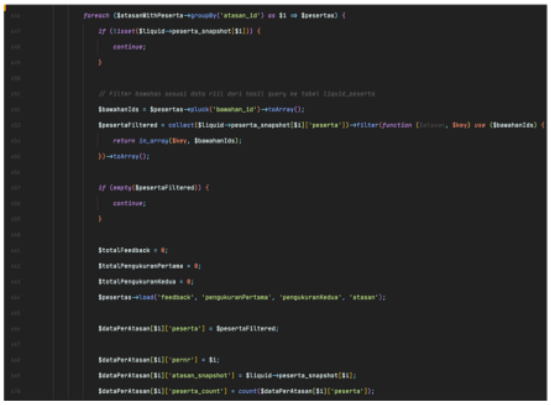
2) Eager Loading; Lazy Eager Loading

Pada aplikasi KOMANDO teknik *query* dengan menggunakan Eager Loading dan Lazy Eager Loading digunakan untuk memuat data relasi dengan model data terkait. Kedua teknik tersebut digunakan sesuai dengan kebutuhan, karena keduanya mempunyai fungsi yang berbeda. Lazy Eager Loading mempunyai sifat penggunaan yang lebih fleksibel, sebuah relasi dapat dimuat setelah sebuah model data utama melalui pengolahan *logic*. Namun, Eager Loading untuk memuat data relasi dari model data utama ketika *query* pertama kali dituliskan. Gambar 4 merupakan contoh penggunaan Eager Loading, sedangkan Gambar 5 merupakan penerapan Lazy Eager Loading.

Kode yang dituliskan pada Gambar 5, akan memuat data yang berkaitan dengan model data LiquidPeserta. Namun, sebelum data relasi tersebut dimuat data LiquidPeserta melalui tahap pengolahan terlebih dahulu, baru setelah itu digunakan Lazy Eager Loading untuk memuat data relasi tersebut. Kedua teknik tersebut sangat mudah untuk diimplementasikan, hanya dengan menggunakan *helper* yang sudah disediakan oleh Laravel. Eager Loading menggunakan *helper* bernama **with()**, sesuai yang ada pada Gambar 4. Sedangkan Lazy Eager Loading menggunakan *helper* **load().**

```
$liquids = Liquid::query()
    ->with('logBook')
    ->activeForUnit($unit)
    ->currentYear()
    ->get();
```

Gambar 4 *Query* data dengan menggunakan teknik Eager Loading



Gambar 5 *Query* data dengan menggunakan teknik Lazy Eager Loading

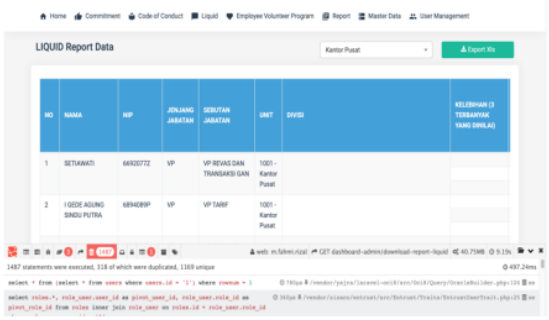
3) View Table

Untuk membuat sebuah *view table* pada sebuah *basisdata* ada beberapa cara yang dilakukan, pada aplikasi KOMANDO *view table* dibuat dengan menggunakan fitur Laravel Command. Pada *framework* Laravel, terdapat fitur di mana kita bisa membuat sebuah *artisan command line* sendiri sesuai dengan kebutuhan. *Custom command* yang digunakan untuk membuat sebuah *view table* sudah disediakan sebelumnya, hanya tinggal membuat *file* dengan ekstensi .sql. File tersebut berisika *query* yang nantinya dieksekusi sehingga menghasilkan *view table*. Gambar 6 merupakan file berektensi .sql, merupakan *query* SQL yang nantinya akan dieksekusi dan menghasilkan *view table* v_rekap_liquid. *View table* hanyalah merupakan *table* biasa pada umumnya, hanya saja kegunaannya mempermudah dalam proses menampikan data yang sudah diolah menjadi sebuah informasi, tanpa harus melalui pengolahan data melalui kode sumber aplikasi.

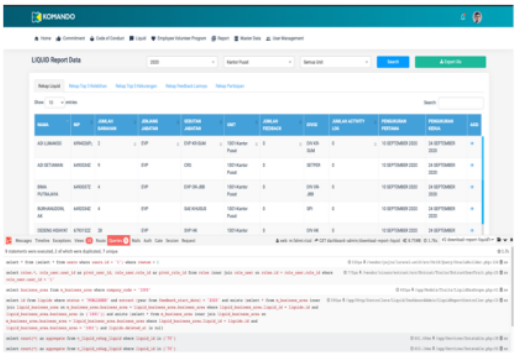
Gambar 6 Query SQL untuk menghasilkan sebuah *view table*

B. Hasil

Setelah dilakukan optimasi dengan menggunakan teknik utilisasi tersebut untuk meningkatkan performa aplikasi, diperoleh hasil yang cukup optimal. Ketiga teknik tersebut dapat dikombinasikan satu sama lain sesuai dengan skenario dan kebutuhan sebuah fungsionalitas. Semaikin banyak teknik utilisasi yang dikombinasikan terhadap sebuah fungsionalitas, maka akan semakin cepat performa yang dihasilkan. Gambar 7 merupakan hasil performa sebuah fungsionalitas rekap *liquid* sebelum dilakukan optimasi. Sedangkan Gambar 8 adalah hasil fungsionalitas rekap *liquid* setelah dilakukan optimasi menggunakan teknik *server-side processing* dan *view table*.



Gambar 7 Rekap *liquid* sebelum dilakukan optimasi

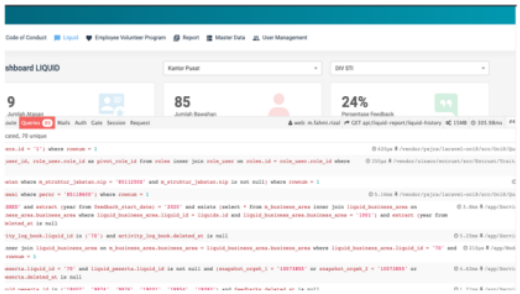


Gambar 8 Rekap *liquid* setelah dilakukan optimasi

Jumlah *query* yang dieksekusi sebelum dan sesudah dilakukan optimasi memiliki selisih jauh. Jumlah *query* sebelum dilakukan optimasi sebanyak 1487 sedangkan setelah dilakukan optimasi jumlah *query* yang dieksekusi hanya sebnyak 9 saja. Selain memangkas sebuah *query* yang dieksekusi, optimasi yang dilakukan juga mempersingkat *load time* sebuah halaman untuk dimuat. Rekap *liquid* memerlukan waktu 9.19 detik untuk dapat dimuat sebelum dilakukan optimasi. Sedangkan setelah dilakukan optimasi, rekap *liquid* hanya memerlukan waktu 1.7 detik.

Memuat data relasi dengan menggunakan teknik Eager Loading dan Lazy Eager Loading memberikan hasil yang sangat signifikan dari jumlah *query* yang dieksekusi dan waktu yang dibutuhkan untuk memuat sebuah halaman. Untuk memuat halaman Dashboard Admin yang di dalamnya memuat fungsionalitas *liquid history* dengan data *unit* Kantor Pusat membutuhkan waktu 501 detik dan jumlah *query* yang harus dieksekusi sebanyak 17.976. Setelah dilakukan optimasi dengan menggunakan Eager Loading dan Lazy Eager Loading untuk memuat data relasi terkait ditambah menggunakan *server-side processing*, hasilnya sangat signifikan. Dengan skenario yang sama, hanya membutuhkan waktu kurang dari satu detik (305 ms) halaman tersebut sudah dapat dimuat dan jumlah *query* yang dieksekusi hanya sebanyak 89. Gambar 9 dan Gambar 10 merupakan hasil tangkapan layar dari skenario pengujian Dashboard Admin sebelum dan sesudah dilakukan optimasi. Selain menggunakan teknik tersebut optimasi dilakukan untuk menambahkan sebuah *filter* divisi, agar data yang dimuat tidak terlalu besar.

Gambar 9 Jumlah *query* dan *load time* Dashboard Admin



Gambar 10 Jumlah *query* dan *load time* Dashboard Admin setelah dilakukan optimasi

V. KESIMPULAN

Berdasarkan pembahasan yang telah diuraikan diatas, tentang bagaimana pentingnya sebuah performa aplikasi serta teknik utilisasi yang dilakukan untuk meningkatkan performa aplikasi pada beberapa fungsionalitas yang terdapat pada modul LIQUID aplikais KOMANDO, dapat ditarik kesimpulan sebagai berikut:

- Teknik yang akan digunakan untuk melakukan optimasi performa aplikasi harus disesuaikan dengan kebutuhan dan skenario sebuah fungsionalitas. Bagaimana skenario sebuah fungsionalitas tersebut berjalan, data apa saja yang akan dimuat, dan bagaimana pengolahan data yang dilakukan sebelum sebuah data tersebut dapat ditampilkan.
- Teknik yang digunakan untuk melakukan pengolahan data harus dapat mengatasi segala kondisi dan skenario, seperti bagaimana ketika jumlah kuantitas data yang ditampilkan berjumlah banyak. Karena seiring dengan berjalannya waktu, ketika aplikasi sudah digunakan khalayak umum, maka tidak dapat dipungkiri kuantitas data yang tersimpan pada basisdata akan bertambah banyak.
- Struktur atau ERD *(Entity Relationship Diagram)* sebuah basisdata yang digunakan akan mempengaruhi proses pengolahan data.

REFERENSI

[1] H. M. Sari, K. A. Laksitowening and H. Hidayati, "Optimasi Aplikasi Web Berbasis Framework Symfonny Dengan Tweak View dan Tweak Cache," *Seminar Nasional Aplikasi Teknologi Informasi,* pp. 1-2, 2010.

[2] B. Subraya, Integrated Aproach to Web Perfomance Testing: A Practitioner's Guide, Hershey: IRM Press, 2006.

[3] Y. B. Samponu and R. Faslah, "Optimasi Query Pada Database Untuk 2-Way SMS DIPENDA Provinsi Sulawesi Utara," vol. III, 2017.

[4] H. Sulastri, A. Rahmatulloh and D. K. Hidayat, "Server-side Processing of Techniques For Optimizing The Speed Of Representing Data," *Jurnal PILAR Nusa Mandiri,* vol. 15, 2019.

[5] M. Surguy, Laravel - My First Framework, Leanpub, 2014.

[6] P. K. Safitri, W. W. Winarno and E. Pramono, "Optimasi Query untuk Sistem Informasi Penjadwalan Mata Pelajaran Sekolah Menggunakan View," *Jurnal Teknologi Informasi,* vol. 13, 2018.

UTILISASI PENGOLAHAN PEMROSESAN DATA UNTUK MENINGKATKAN PERFORMA APLIKASI

ORIGINALITY REPORT

2%	2%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	jti.respati.ac.id Internet Source	1%
2	masihsejarah.blogspot.com Internet Source	<1%
3	nahdian.blogspot.com Internet Source	<1%
4	jurnal.untad.ac.id Internet Source	<1%
5	www.rajaputramedia.com Internet Source	<1%
6	journal.stmikjayakarta.ac.id Internet Source	<1%
7	edihudiata.multiply.com Internet Source	<1%
8	core.ac.uk Internet Source	<1%
9	Julia Kurniasih, Ema Utami, Suwanto Raharjo.	

"Heuristics and Metaheuristics Approach for Query Optimization Using Genetics and Memetics Algorithm", 2019 1st International Conference on Cybernetics and Intelligent System (ICORIS), 2019

<1%

Publication

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		