Pemanfaatan Laravel Eloquent ORM Pada Fitur Kelola Transaksi Dalam Aplikasi Balelabs Billing

Naufal Hanan Oktavanusa
Jurusan Informatika, Fakultas Teknologi Industri
Universitas Islam Indonesia
Yogyakarta
18523208@students.uii.ac.id

Rahadian Kurniawan
Jurusan Informatika, Fakultas Teknologi Industri
Universitas Islam Indonesia
Yogyakarta
rahadiankurniawan@uii.ac.id

***Abstract*— PT Bale Lab Indonesia (Balelabs) adalah salah satu perusahaan di Yogyakarta yang bergerak dalam bidang teknologi informasi. Balelabs memberikan layanan pada pengembangan perangkat lunak dan *IT Consultant* dengan berfokus pada jasa pembuatan, pengembangan dan optimasi *website* serta aplikasi berbasis *mobile*. Selama ini, Balelabs masih menggunakan model bisnis konvensional untuk memasarkan produk mereka. Balelabs tengah mengembangkan aplikasi *e-commerce* bernama Balelabs Billing guna meningkatkan proses bisnis di perusahaan. Dalam pengembangan aplikasi ini, Balelabs ingin agar aplikasi tersebut dapat dikembangkan dengan cepat dan juga dapat di-*maintain* dengan mudah. Oleh karena itu, Laravel Eloquent digunakan sebagai *model query* pada pengembangan aplikasi tersebut. Laravel Eloquent adalah salah satu *model query* yang disediakan oleh Laravel. Eloquent memberikan *readibility* dan *maintainability* yang lebih baik dari pada *model query* lain seperti *Raw SQL* [1]. Tujuan akhir dari pemanfaatan Laravel Eloquent pada fitur kelola transaksi dalam aplikasi Balelabs Billing ialah dapat mengurangi masa pengembangan aplikasi dan mempermudah perawatan aplikasi nantinya oleh pengembang yang lain.**

Keywords—Teknologi Informasi, *E-commerce*, Laravel, Eloquent ORM, PT Bale Lab Indonesia

I. PENDAHULUAN

Perkembangan ilmu pengetahuan dan teknologi yang cepat khususnya pada bidang teknologi informasi sangat berperan dalam berbagai aspek kehidupan. Segala kegiatan yang dilakukan sehari-hari pasti tidak lepas dari sentuhan teknologi. Pengembangan teknologi informasi sendiri memiliki berbagai tujuan, antara lain: untuk memecahkan suatu masalah, mengefisiensikan pekerjaan, dan menciptakan hal baru. PT Bale Lab Indonesia (Balelabs) adalah salah satu perusahaan di Yogyakarta yang bergerak dalam bidang teknologi informasi. Balelabs memberikan layanan pada pengembangan perangkat lunak dan *IT Consultant* dengan berfokus pada jasa pembuatan, pengembangan dan optimasi *website* serta aplikasi berbasis *mobile*.

Selama ini, Balelelabs masih menggunakan metode transaksi konvensional untuk setiap transaksi yang didapat. Dalam metode konvensional, transaksi terjadi secara langsung dimana konsumen dan perusahaan saling bertatap muka saat melakukan transaksi. Metode konvensional memiliki beberapa kelemahan, antara lain: kurangnya efisiensi saat pemesanan aplikasi, jangkauan pasar yang lebih kecil, biaya yang lebih mahal, dan keterbatasan waktu oleh konsumen ketika ingin membeli aplikasi [2]. Jika hal ini tetap dibiarkan, proses bisnis akan menjadi lebih lambat dan perusahaan menjadi tidak dapat berkembang.

Balelabs tengah mengembangkan aplikasi *e-commerce* bernama Balelabs Billing yang bertujuan untuk meningkatkan proses bisnis yang terjadi di perusahaan. *E-commerce* dipilih karena telah diuji di beberapa perusahaan dan terbukti mampu untuk meningkatkan proses bisnis perusahaan tersebut [3]. Aplikasi ini dikembangkan sebagai media pembelian dan manajemen paket aplikasi untuk konsumen balelabs. Dengan Balelabs Billing, konsumen tidak perlu mendatangi kantor Balelabs saat hendak memesan sebuah aplikasi. Hal ini memberikan efisiensi transaksi antara konsumen dan perusahaan sehingga proses bisnis menjadi lebih cepat dan optimal [2].

Selain itu, Balelabs juga ingin agar Balelabs Billing memiliki masa pengembangan yang cepat, dan pemeliharaan yang mudah. Dibutuhkan sebuah teknologi yang dapat memperpendek waktu pengembangan sekaligus meningkatkan *maintainability* sistem agar sistem dapat dengan mudah dikelola oleh pengembang yang lain. Untuk memenuhi tujuan tersebut, pengembangan dilakukan dengan memanfaatkan Laravel Eloquent sebagai *model query* yang digunakan. Laravel Eloquent adalah salah satu *model query* yang disediakan oleh Laravel. Eloquent memberikan *readibility* dan *maintainability* yang lebih baik dari pada *model query* lain seperti *Raw SQL* [1]

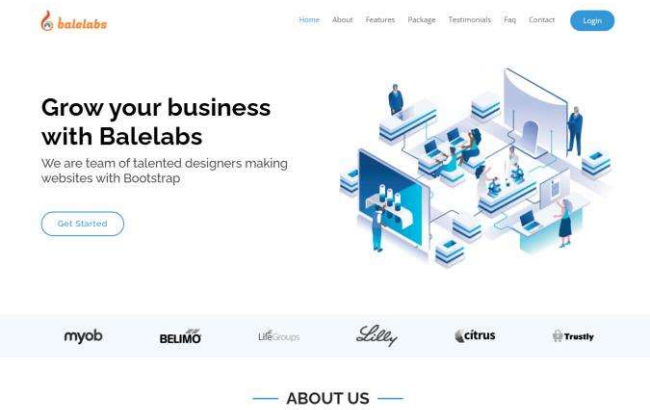
Pada makalah ini, akan dijelaskan tentang bagaimana Laravel Eloquent menjadi sebuah teknologi yang dapat mengurangi masa pengembangan aplikasi dan mempermudah perawatan aplikasi nantinya oleh pengembang yang lain. Makalah ini juga akan membahas tentang kapan sebaiknya Laravel Eloquent dimanfaatkan dalam pengembangan sebuah aplikasi. Dari makalah ini, diharapkan agar pengembang memiliki gambaran tentang pemilihan *model query* yang cocok digunakan pada aplikasi yang akan dikembangkan.

II. LANDASAN TEORI

A. *Aplikasi Balelabs Billing*

Aplikasi Balelabs Billing merupakan aplikasi *e-commerce* berbasis web yang dikembangkan sebagai media pembelian dan manajemen paket aplikasi untuk konsumen

Balelabs. Dengan Balelabs Billing, konsumen tidak perlu mendatangi kantor Balelabs saat hendak memesan sebuah aplikasi. Balelabs Billing menyediakan berbagai pilihan paket aplikasi dan *add-on* yang dapat ditambahkan pada aplikasi sehingga konsumen dapat dengan bebas memilih fitur apa saja yang ingin ditambahkan pada aplikasi mereka. Balelabs Billing juga memberikan kemudahan dalam pengelolaan data transaksi, data konsumen, dan data aplikasi yang akan ditawarkan kepada konsumen Balelabs. Gambar 1 adalah tampilan dari *landing page* dari aplikasi Balelabs Billing.

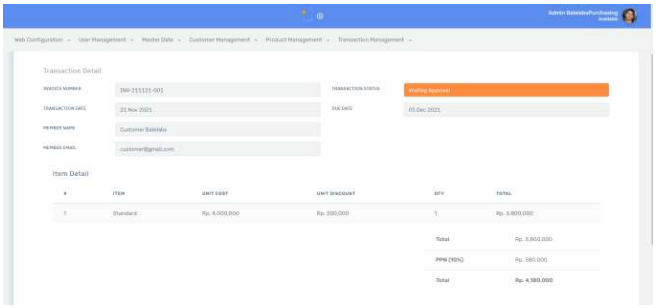


Gambar 1 *Landing page* dari aplikasi Balelabs Billing

B. *Fitur Kelola Transaksi*

Fitur kelola transaksi adalah salah satu fitur yang terdapat pada aplikasi Balelabs Billing. Fitur ini berfungsi sebagai media administrator untuk mengelola setiap transaksi yang dilakukan oleh konsumen. Fitur ini dibutuhkan agar administrator dapat menyeleksi keabsahan transaksi yang telah dilakukan.

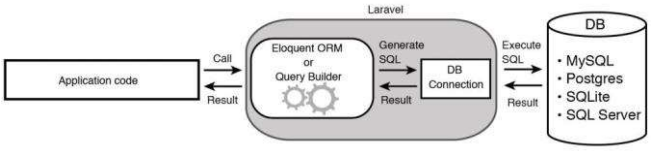
Pada fitur kelola transaksi, administrator dapat menolak atau menyetujui sebuah transaksi. Gambar 2 merupakan tampilan dari halaman kelola transaksi pada salah satu transaksi yang telah dilakukan oleh konsumen.



Gambar 2 Halaman kelola transaksi pada aplikasi Balelabs Billing

C. *Eloquent ORM*

Laravel memberikan sebuah fitur yang sangat memudahkan pengembangnya dalam menuliskan *query*. Fitur tersebut adalah Eloquent ORM (*Object Relational Mapping)* [4]. Laravel Eloquent menyediakan berbagai fungsi untuk mengeksekusi suatu *query* dengan penggunaan yang mudah. Selain memberikan kemudahan dalam menuliskan *query,* Eloquent juga memberikan pencegahan terhadap serangan seperti SQL Injection. Hal ini disebabkan karena Eloquent menggunakan *PDO parameter binding* agar pengguna tidak dapat mengirim suatu *input* yang dapat mengubah maksud dari suatu *query* [5]. Gambar 3 merupakan *workflow* dari Laravel Eloquent.



Gambar 3 *Workflow* Laravel Eloquent (Sumber: https://maxoffsky.com/code-blog/laravel-first-framework-chapter-6-database-operations/)

Gambar 3 memperlihatkan tentang *workflow* dari Laravel Eloquent. Ketika aplikasi mengeksekusi kode yang berisikan perintah operasi basis data, kode akan diubah menjadi SQL *statement* oleh Laravel. Lalu SQL *statement* tersebut kemudian dieksekusi oleh basis data. SQL *statement* yang telah dieksekusi kemudian dikembalikan ke aplikasi dalam bentuk *collection* [6].

D. *SQL*

Structured Query Language (SQL) adalah bahasa pemrograman khusus yang ditujukan untuk menyimpan, memanipulasi, dan menerima respon data yang disimpan dalam sebuah basis data. SQL merupakan bahasa standar yang digunakan oleh berbagai *database management system* (DBMS), seperti: MySQL, Oracle, Postgres, dan SQL Server [7].

E. *E-Commerce*

E-Commerce adalah model bisnis yang berfokus pada skala ekspansi yang cepat. *e-commerce* meningkatkan efektivitas operasi dan prospek perusahaan terhadap pertumbuhan produktivitas. Beberapa manfaat yang didapatkan dari model bisnis *e-commerce*, antara lain: memperluas jangkauan pasar, menggunakan teknik modern dengan media elektronik alih-alih teknik tradisional dengan kertas, dan pengurangan jumlah gudang atau biaya persediaan [8].

F. *Laravel*

Laravel merupakan salah satu framework PHP yang digunakan dalam pengembangan aplikasi berbasis web. Laravel membantu memaksimalkan penggunaan PHP dalam pengembangan *website* [9]. Hal ini dikarenakan Laravel menyediakan sintaks yang ekspresif, jelas, dan menghemat waktu. Laravel telah dikembangkan sejak tahun 2011 oleh Taylor Otwell dan telah mengalami perkembangan yang pesat. Framework ini berfokus pada kejelasan dan kerapia sehingga menghasilkan fungsionalitas aplikasi yang optimal. Hal inilah yang membuat laravel dapat digunakan dalam mengembangkan proyek dari skala kecil hingga besar. Laravel juga menggunakan konsep *Model View Controller* (MVC) sehingga proses pengembangan aplikasi menjadi lebih terstruktur [10].

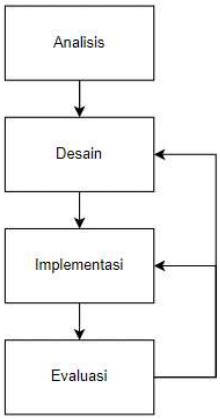
G. *Laravel Artisan*

Laravel Artisan adalah *Command-line Utility* (CLI) yang dimiliki oleh *framework* Laravel. Artisan menyediakan

banyak perintah yang sangat memudahkan pengembang dalam mengembangkan sebuah aplikasi. Penggunaan artisan sangat membantu pengerjaan banyak hal yang bila dikerjakan secara manual akan memakan banyak waktu [11].

III. METODOLOGI

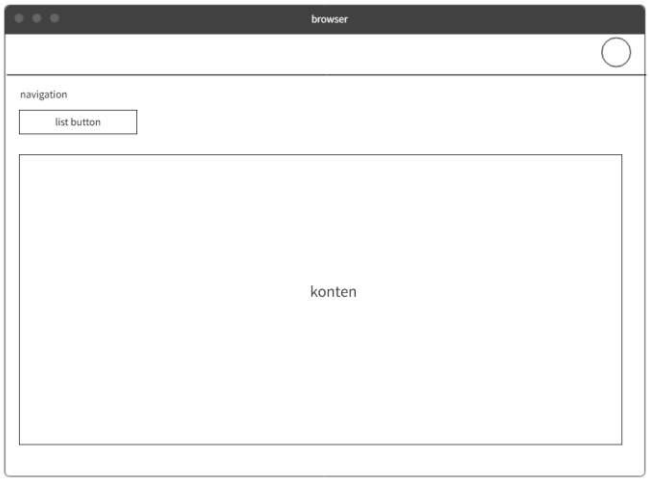
Gambar 4 memperlihatkan metodologi yang digunakan dalam makalah ini. Tahapan metodologi dimulai dari tahap analisis sampai ke tahap evaluasi. Apabila terdapat keluaran yang belum sesuai pada tahap evaluasi, pengembangan akan kembali pada tahap desain/implementasi tergantung dari hasil evaluasi yang sebelumnya telah didapat.



Gambar 4 Metodologi pengembangan aplikasi Balelabs Billing

A. *Analisis*

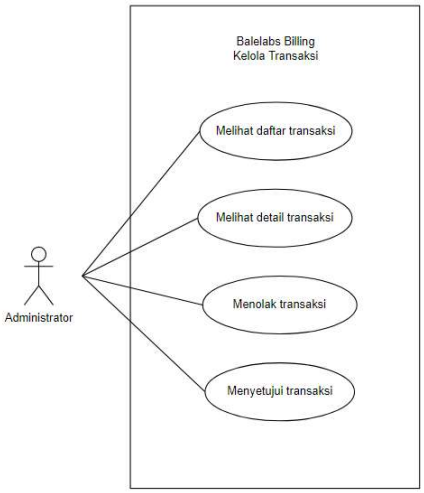
Pada tahap awal pengembangan, dilakukan pengumpulan data. Pengumpulan data dilakukan dengan melakukan wawancara pada pengguna dan observasi pada sistem sejenis. Data yang telah dikumpulkan kemudian dianalisis untuk mendapatkan kebutuhan sistem. Kebutuhan tersebut lalu diubah menjadi *wireframe* agar gambaran sistem menjadi lebih jelas. Gambar 5 adalah *wireframe* yang telah dibuat pada fitur kelola transaksi dalam aplikasi Balelabs Billing.



Gambar 5 *Wireframe* fitur kelola transaksi pada aplikasi Balelabs Billing

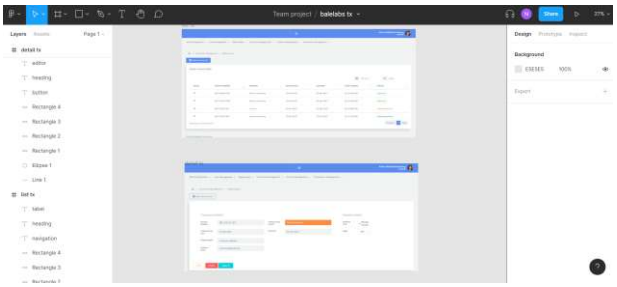
B. *Desain*

Berdasarkan analisis yang telah dilakukan, pengembangan dilanjutkan dengan membuat desain *use case diagram,* dan *mockup* dari *wireframe* yang sebelumnya telah dibuat. Gambar 6 adalah *use case diagram* yang dihasilkan untuk fitur kelola transaksi pada aplikasi Balelabs Billing.



Gambar 6 *Use case diagram* untuk fitur kelola transaksi pada aplikasi Balelabs Billing

Mockup dibuat menggunakan Figma oleh UI/UX *Designer*. Pembuatan *mockup* bertujuan untuk memberikan pemahaman terkait alur kerja sebuah fitur kepada pengembang. Gambar 7 adalah beberapa *mockup* yang telah dibuat untuk fitur kelola transaksi.



Gambar 7 *Mockup* fitur kelola transaksi pada aplikasi Balelabs Billing

C. *Implementasi*

Pengembangan kemudian dilanjutkan dengan mengimplementasikan tahapan desain ke dalam sistem/pengkodean. Pada tahap ini, fitur kelola transaksi akan diimplementasikan dengan memanfaatkan Laravel Eloquent.

1. Membuat *model*

Model adalah salah satu bagian dari konsep *Model View Controll (MVC)* yang mempresentasikan struktur dan logika dari basis data. *Model* ini nantinya akan digunakan oleh *controller* sebagai media untuk mendapatkan respon dari basis data. Salah satu cara untuk mendefinisikan *model* adalah dengan menggunakan perintah Artisan **make:model** pada Command Line Prompt.

```
php artisan make:model SalesInvoice
```

Gambar 8 Perintah artisan untuk membuat *model*

Gambar 8 merupakan perintah yang dapat digunakan untuk membuat *model* menggunakan Laravel Artisan. SalesInvoice pada perintah tersebut merupakan nama *model* yang akan dibuat. *Model* ini nantinya akan digunakan sebagai media untuk berkomunikasi dengan tabel sales_invoices pada basis data.

Setelah perintah dieksekusi, sistem akan membuat *file* baru bernama SalesInvoice.php. Untuk menghubungkan *model* pada basis data terkait, terdapat beberapa variabel yang harus ditambahkan, yakni nama dan *primary key* tabel. Beberapa fungsi juga harus ditambahkan pada *model* SalesInvoice agar dapat menerima data relasi dari *model* yang lain. Gambar 9 merupakan struktur kode pada *file* SalesInvoice.php.

```php
<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class SalesInvoice extends Model
{
    protected $primaryKey = 'id';
    protected $table = 'sales_invoices';

    public function company(){
        return $this->belongsTo(Company::class);
    }
    public function user(){
        return $this->belongsTo(User::class, 'payment_verify_by');
    }
    public function bank(){
        return $this->belongsTo(MgtBank::class, 'payment_bank');
    }
    public function customer(){
        return $this->belongsTo(Customer::class)->withTrashed();
    }
    public function salesInvoiceLines(){
        return $this->hasMany(SalesInvoiceLine::class);
    }
}
```

Gambar 9 Struktur kode *file* SalesInvoicesLine.php

2. Mendeklarasikan *model* pada *controller*

Setelah *model* didefinisikan, *controller* kemudian dapat menggunakan *class model* tersebut untuk mengambil data dari basis data menggunakan Laravel Eloquent ORM. Untuk menggunakan sebuah *model*, *controller* harus terlebih dahulu mendeklarasikan *class model*. Gambar 10 memperlihatkan pendeklarasian *model* SalesInvoice pada *controller.*

```php
use App\Models\SalesInvoice;
```

Gambar 10 Pendeklarasian *model* pada *controller*

3. Pengambilan data menggunakan Laravel Eloquent

Pengambilan data dilakukan dengan menggunakan Laravel Eloquent. Laravel Eloquent menyederhanakan penggunaan *query* sehingga kode menjadi lebih sederhana dan mudah dipelihara oleh pengembang [1]. Gambar 11 adalah kode pada *controller* untuk mengambil data transaksi dengan menggunakan *model query* Laravel Eloquent.

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

use Illuminate\Http\Request;
use App\Models\SalesInvoice;

class SalesInvoiceController extends Controller
{
    public function detail(Request $req)
    {
        $transaction = SalesInvoice::find($req->route('id'))
            ->salesInvoiceLines()
            ->with('package', 'addon')
            ->get();

        return view('sales_invoice.detail', compact('transaction'));
    }

}
```

Gambar 11 Struktur kode pada *controller* fitur kelola transaksi

4. Menampilkan respon data pada *view*

Data yang telah diterima kemudian diteruskan pada komponen *view*. Dengan menggunakan Laravel Eloquent, penulisan kode pada view juga menjadi lebih mudah. Hal ini disebabkan karena respon data dari Laravel Eloquent berbentuk objek sehingga pengambilan nilai-nilai pada respon data menjadi lebih sederhana. Gambar 12 adalah kode untuk menampilkan respon data pada *view.*

```blade
<div class="table-responsive-sm">
    <table class="table table-striped">
        <thead>
            <tr>
                <th class="center">#</th>
                <th>Item</th>
                <th class="right">Unit Cost</th>
                <th class="right">Unit Discount</th>
                <th class="center">Qty</th>
                <th class="right">Total</th>
            </tr>
        </thead>
        <tbody>
            @foreach ($transaction as $item)
            <tr>
                <td class="center">{{$loop->iteration}}</td>
                <td class="left strong">{{$item->package->name}}</td>
                <td class="right">Rp. {{number_format($item->package->amount)}}</td>
                <td class="center">Rp. {{number_format($item->discount_amount)}}</td>
                <td class="center">1</td>
                <td class="right">Rp. {{number_format($item->total_amount)}}</td>
            </tr>
            @endforeach
        </tbody>
    </table>
</div>
```

Gambar 12 Struktur kode pada *view* fitur kelola transaksi

D. Evaluasi

Saat sebuah kebutuhan telah selesai diimplementasikan, kebutuhan tersebut kemudian dievaluasi agar sesuai dengan keluaran yang diharapkan. Evaluasi dilakukan dengan menggunakan metode *black box testing*. *Black box testing* dipilih karena penggunaannya yang mudah sehingga lebih banyak anggota tim yang dapat ikut serta melakukan pengujian.

Apabila keluaran masih belum sesuai, fase pengembangan akan kembali pada tahap yang diperlukan. Sebagai contoh, apabila terdapat *bug/error* pada sistem, fase akan kembali pada tahap implementasi. Namun apabila terdapat kebutuhan yang berubah sehingga memerlukan pembaharuan desain, fase akan kembali pada tahap desain. Tabel 1 adalah hasil dari pengujian aplikasi Balelabs Billing pada fitur kelola transaksi menggunakan metode *black box testing.*

Tabel 1 Hasil pengujian menggunakan metode *black box testing* pada fitur kelola transaksi

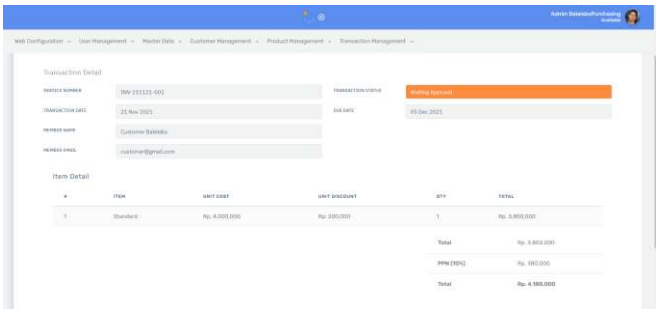
No	Uji kasus	Hasil yang diharapkan	Hasil pengujian	status
1	Admin mengubah status transaksi menjadi disetujui	Status transaksi berubah menjadi *approved* dan Sistem mengirim email pemberitahuan pada konsumen bahwa transaksi telah disetujui	Status transaksi berubah menjadi *approved* dan Sistem mengirim email pemberitahuan pada konsumen bahwa transaksi telah disetujui	Valid
2	Admin mengubah status transaksi menjadi ditolak	Status transaksi berubah menjadi *rejected* dan Sistem mengirim email pemberitahuan pada konsumen bahwa transaksi telah ditolak	Status transaksi berubah menjadi *rejected* dan Sistem mengirim email pemberitahuan pada konsumen bahwa transaksi telah ditolak	Valid
3	Konsumen tidak melakukan pembayaran sampai dengan satu 3 hari sebelum batas pembayaran	Sistem akan mengirim email pemberitahuan pada konsumen bahwa transaksi akan kadaluwarsa	Sistem akan mengirim email pemberitahuan pada konsumen bahwa transaksi akan kadaluwarsa	Valid
4	Konsumen tidak melakukan pembayaran selama lebih dari 14 hari	Status transaksi berubah menjadi *expired* dan Sistem mengirim email pemberitahuan pada konsumen bahwa transaksi kadaluwarsa	Status transaksi berubah menjadi *expired* dan Sistem mengirim email pemberitahuan pada konsumen bahwa transaksi kadaluwarsa	Valid

IV. PEMBAHASAN

Terdapat beberapa *model query* yang dapat digunakan dalam mengembangkan sebuah aplikasi. Pada aplikasi web berbasis PHP dengan framework Laravel sendiri, terdapat dua *model query* yang dapat digunakan, yaitu: Eloquent ORM, dan raw SQL. Pada pengembangan aplikasi Balelabs Billing, Eloquent ORM dipilih sebagai *model query* yang digunakan.

Eloquent dipilih karena memberikan beberapa kelebihan, seperti *readibility* dan *maintainability* [1]. Secara *default,* penggunaan Eloquent juga memberikan keamanan yang lebih baik daripada *model* raw SQL sehingga aplikasi menjadi lebih aman terhadap serangan seperti SQL Injection [5].

Pada pembahasan ini, akan dilakukan perbandingan antara kesederhanaan kode antara *model query* menggunakan Laravel Eloquent dan Raw SQL. Perbandingan akan dilakukan pada fitur kelola transaksi dalam aplikasi Balelabs Billing. Gambar 13 adalah tampilan halaman kelola transaksi dalam aplikasi Balelabs Billing.



Gambar 13 Tampilan halaman kelola transaksi pada aplikasi Balelabs Billing

Agar dapat menghasilkan informasi yang nantinya akan digunakan pada halaman kelola transaksi seperti pada gambar 13, sistem perlu mengeksekusi *query* untuk menerima respon data dari basis data. Gambar 14 merupakan *query* yang digunakan untuk menerima respon data pada fitur kelola transaksi dengan menggunakan *model query* Laravel Eloquent. Sedangkan Gambar 15 merupakan *query* dengan menggunakan *model* raw SQL.

```
$invoice = SalesInvoice::with('bank', 'user')
        ->find($req->route('id'));

$transaction = SalesInvoice::find($req->route('id'))
        ->salesInvoiceLines()
        ->with('package', 'addon')
        ->get();
```

Gambar 14 *Query* data dengan menggunakan Laravel Eloquent

```
$invoice = DB::select("select * from sales_invoices
    left join mgt_banks on
        sales_invoices.payment_bank = mgt_banks.id
    left join users on
        sales_invoices.payment_verify_by = users.id
    where sales_invoices.id = :id", ['id'=>$req->route('id')]);

$transaction = DB::select("select * from sales_invoice_lines
    left join mgt_packages on
        sales_invoice_lines.package_id = mgt_packages.id
    left join mgt_package_addons on
        sales_invoice_lines.addon_id = mgt_package_addons.id
    where sales_invoice_lines.id = :id", ['id'=>$req->route('id')]);
```

Gambar 15 *Query* data dengan menggunakan raw SQL

Pada gambar 14, seluruh relasi yang dibutuhkan telah didefinisikan sebagai fungsi di dalam *model* sehingga Eloquent hanya perlu melakukan pemanggilan fungsi-fungsi tersebut. Sedangkan pada gambar 15, data relasi yang dibutuhkan perlu ditulis dan didefinisikan terlebih dahulu sehingga membuat baris kode yang diperlukan menjadi lebih banyak dibandingkan dengan baris kode yang menggunakan *model query* Eloquent.

Laravel Eloquent hanya membutuhkan 7 baris kode untuk melakukan *query*. Sedangkan pada Raw SQL, baris kode yang dibutuhkan berjumlah 13 baris. Jumlah baris kode yang digunakan pada Laravel Eloquent lebih sedikit dibandingkan dengan baris kode pada *query* yang menggunakan *model* Raw SQL. Masa pengembangan aplikasi tentu akan menjadi lebih cepat apabila seluruh fungsionalitas sistem dituliskan menggunakan *model* Eloquent.

Selain baris kode yang lebih minim, dapat dilihat pada gambar 14 bahwa kode yang ditulis lebih sederhana dan lebih mudah dibaca dibandingkan dengan kode pada gambar 15. Dengan kemudahan pembacaan kode, pengembang yang lain dapat memahami kode yang ditulis dengan lebih baik. Hal ini juga membuat pemeliharaan sistem menjadi lebih efisien karena seluruh relasi basis data telah didefinisikan dalam satu tempat yakni pada *model*.

Perlu diketahui bahwa penggunaan Eloquent tidak selalu lebih baik dari raw SQL. Pada aspek kecepatan pengambilan data, *model* raw SQL masih lebih baik dari pada Laravel Eloquent [7]. Hal ini disebabkan karena pada eloquent, setiap parameter yang diberikan akan dikeluarkan dalam bentuk objek sehingga proses pengambilan data menjadi lebih lambat. Sedangkan pada raw SQL, pengambilan data langsung dilakukan pada basis data dan dikeluarkan dalam bentuk array [12]. Tabel 2 menyajikan data terkait waktu respon rata-rata yang diperlukan untuk mengeksekusi *query* dengan menggunakan Eloquent. Sedangkan tabel 3 adalah data terkait waktu respon rata-rata yang diperlukan untuk mengeksekusi *query* dengan menggunakan Raw SQL

Tabel 2 Waktu respon rata-rata antara *model query* Laravel Eloquent dan raw SQL dalam detik

Joins	*Average (ms)*
1	162,2
3	1002,7
4	1540,0

Sumber : Jound, I., & Halimi, H. "Comparison of performance between Raw SQL and Eloquent ORM in Laravel" Blekinge Institute of Technology, Karlskrona Sweden, 2016

Tabel 3 Waktu respon rata-rata antara *model query* Laravel Eloquent dan raw SQL dalam detik

Joins	*Average (ms)*
1	116,4
3	130,6
4	155,2

Sumber : Jound, I., & Halimi, H. "Comparison of performance between Raw SQL and Eloquent ORM in Laravel" Blekinge Institute of Technology, Karlskrona Sweden, 2016

Tabel 2 dan 3 menampilkan kecepatan rata-rata respon data untuk tiap *join* pada Eloquent dan Raw SQL. Waktu respon data semakin bertambah seiring dengan jumlah join yang digunakan. Dengan membandingkan kedua tabel yang ada, dapat dilihat bahwa Raw SQL memiliki kecepatan yang lebih baik daripada Eloquent. Pada pengujian menggunakan 1 *joins,* Raw SQL 45,8 milidetik lebih cepat dibandingkan dengan Eloquent. Lalu Pada pengujian menggunakan 3 *joins,* perbedaan kecepatan bertambah menjadi 872,1 milidetik. Dan pada pengujian menggunakan 4 *joins,* perbedaan kecepatan respon data menggunakan Raw SQL mencapai 1284,8 milidetik lebih cepat dibandingkan dengan Eloquent.

V. KESIMPULAN

Salah satu fokus dari pengembangan aplikasi Balelabs Billing adalah membangun sistem dengan masa pengembangan yang cepat, dan pemeliharaan yang mudah. Untuk mencapai hal tersebut, pengembangan aplikasi memanfaatkan Laravel Eloquent ORM sebagai *model query* yang digunakan. Dari pengembangan sistem ini, dapat disimpulkan bahwa:

- Eloquent ORM dapat digunakan ketika proyek membutuhkan struktur kode yang sederhana serta *readibility* dan *maintainability* yang baik.
- Jumlah baris kode yang digunakan pada Laravel Eloquent lebih sedikit dibandingkan dengan baris kode pada *query* yang menggunakan *model* Raw SQL. Masa pengembangan aplikasi tentu akan menjadi lebih cepat apabila seluruh fungsionalitas sistem dituliskan menggunakan *model* Eloquent.
- Eloquent ORM tidak direkomendasikan untuk digunakan pada aplikasi yang membutuhkan waktu pemuatan/performa yang cepat.

REFERENSI

[1] F. Malatesta, Learning Laravel's Eloquent, MUMBAI: Packt Publishing Ltd, 2015.

[2] D. C. Gupta, P. M. Bindal, N. Agarwal and K. Khandelwal, "Traditional Commerce v/s E-commerce and Impact of Demonetization on E-commerce," *International Journal of Engineering and Management Research,* vol. 8, pp. 136-142, 2018.

[3] J. F. Andry, "ANALISA PERBANDINGAN PENERAPAN E-COMMERCE TERHADAP TRANSAKSI PENJUALAN KONVENSIONAL MENGGUNAKAN METODE SIMPLE MOVING AVERAGE," *JURNAL TEKNOLOGI INFORMASI,* vol. 11, pp. 19-26, 2015.

[4] M. B. Alhaq and A. Sujarwo, "UTILISASI PENGOLAHAN PEMROSESAN DATA UNTUK MENINGKATKAN PERFORMA APLIKASI," *Automata Vol. 2 No. 1 2021,* vol. 2, no. 1, p. 330, 2021.

[5] A. Pratama, P. Sukarno and A. A. Wardana, "Analisis dan Perbandingan Pencegahan SQL Injection pada Framework CodeIgniter dengan Escaping Query dan Framework Laravel dengan Eloquent ORM," *e-Proceeding of Engineering : Vol.7, No.1 April 2020,* vol. 7, p. 2245, 2020.

[6] M. Surguy, "maxoffsky," 20 September 2014. [Online]. Available: https://maxoffsky.com/code-blog/laravel-first-framework-chapter-6-database-operations/.

[7] I. Jound and H. Halimi, "Comparison of performance between Raw SQL and Eloquent ORM in Laravel," Blekinge Institute of Technology, Karlskrona Sweden, 2016.

[8] T. Gajewska, D. Zimon, G. Kaczor and P. Madzík, "The Impact of The Level of Customer Satisfaction on

The Quality of E-commerce Services," *International Journal of Productivity and Performance Management, Vol. 69 No. 4, pp. 666-684. https://doi.org/10.1108/IJPPM-01-2019-0018,* 2020.

[9] Iwana, "Laravel Adalah," 6 February 2020. [Online]. Available: https://jeriatno.medium.com/laravel-adalah-4f876fc71a72.

[10] K. NFA, "Laravel — Pengertian, Kelebihan, Kekurangan dan Cara Install Laravel," 4 September 2019. [Online]. Available: https://medium.com/@kevinnfa0107/laravel-pengertian-kelebihan-kekurangan-dan-cara-install-laravel-224a79550a91.

[11] S. E. Wahyudi, "Artisan Console," 9 October 2019. [Online]. Available: http://informatika.uc.ac.id/id/2019/10/laravel-artisan/.

[12] H. Budaraju, "Laravel Eloquent ORM Vs Raw SQL ?," 20 December 2017. [Online]. Available: https://medium.com/@harshavardhanbudaraju/laravel-eloquent-orm-vs-raw-sql-d576276ee848.