Idealis: Indonesia Journal Information System
Volume 7, Nomor 1, Januari 2024
ISSN 2684-7280 (*online*)
Halaman 118-127
available online at http://jom.fti.budiluhur.ac.id/index.php/IDEALIS/index

REST API PADA TOKO KELONTONG UNTUK TRANSAKSI PENJUALAN MENGGUNAKAN FRAMEWORK LARAVEL

Hendrik Fery Herdiyatmoko[1*], Yohanes Dicka Pratama[2]
[1*]Informatika, Fakultas Sains Dan Teknologi, Universitas Katolik Musi Charitas, Palembang, Indonesia
[2]Teknik Industri, Fakultas Sains Dan Teknologi, Universitas Katolik Musi Charitas, Palembang, Indonesia
Email: [1*]hendrik@ukmc.ac.id, [2]dicka@ukmc.ac.id

Abstrak- Bisnis waralaba merupakan peluang bisnis yang masih menjanjikan saat ini, sehingga membutuhkan penangan keuangan yang baik untuk mengurangi permsalahan kesalahan transaksi dan keterbatasan akses. EBillToko adalah sistem keuangan toko kelontong yang dikembangkan dengan mengolah transaksi penjualan yang terhubung ke server database. Dengan sebuah sistem terintegrasi, semua pendapatan dari penjualan toko dapat dipantau secara real-time. Perancangan sistem dibuat dengan mengimplementasikan arsitektur RESTFul API dan keamanan token akses. REST API dikembangkan dengan teknologi VUE.JS dan Laravel. Prototipe dikembangkan di atas server database menggunakan tujuh tabel. untuk REST API yang dibuat terdapat satu titik akhir yang dibuat. Eksperimen ditetapkan dengan menguji akses aplikasi dengan white box dan black box testing. Hasil percobaan menunjukkan Implementasi VUE.JS dan Laravel memiliki kinerja baik dengan token akses JASON WEB TOKEN (JWT), dan implementasi Laravel sebagai REST Server dan VUE.JS sebagai REST Client memiliki interpherabilitas yang baik. Penelitian ini memberi kontribusi pada model akses data dengan REST API yang berbeda dengan pemrograman database umumnya. Pemrograman REST API memberikan keleluasaan akses ke platform lain karena mendukung interpherabilitas yang baik.

Kata Kunci: API, Laravel, REST, Vue.JS, Web Service

Abstract- *The franchise business is a business opportunity that is still promising at this time, so it requires good financial management to reduce the problems of transaction errors and limited access. EBillToko is a grocery store financial system developed by processing sales transactions connected to a database server. With an integrated system, all income from store sales can be monitored in real-time. The system design was created by implementing RESTFul API architecture and access token security. The REST API was developed with VUE.JS and Laravel technology. The prototype was developed on top of a database server using seven tables. for the REST API created there is one endpoint created. Experiments are set by testing application access with white box and black box testing. The experimental results show that the VUE.JS and Laravel implementations have good performance with the JASON WEB TOKEN (JWT) access token, and the Laravel implementation as a REST Server and VUE.JS as a REST Client have good interpherability. This research contributes to a data access model with a REST API that is different from general database programming. REST API programming provides flexible access to other platforms because it supports good interpherability.*

***Keywords**: API, Laravel, REST, Vue.JS, Web Service*

1. PENDAHULUAN

Penggunakan teknologi pada setiap pekerjaan manusia merupakan hal yang lazim pada zaman ini. Pada umumnya perusahaan, instansi, dan organisasi tidak terlepas dari penggunaan teknologi, terutama dalam pengolahan data[1][2]. Penggunaan teknologi tidak hanya ditunjukan untuk memudahkan pekerjaan, tetapi juga untuk mencapai hasil kerja secara efektif dan efisien [3]. Pembuatan teknologi ini juga dapat digunakan oleh siapa saja baik usaha besar, menengah walaupun kecil seperti Toko Kelontong XYZ.

Toko Kelontong XYZ merupakan toko sembako yang menjual produk kebutuhan harian. Pengolahan transaksi jual beli barang dan pelaporan keuangan pada Toko Kelontong XYZ masih menggunakan cara manual seperti mencatat seluruh pengeluaran dan pendapatan kedalam satu buku. Permasalahan yang dihadapi Toko Kelontong XYZ adalah sering terjadi pada saat mencatat pemasukan dan pengeluaran terjadi kesalahan input data dan bukti transaksi hilang. Untuk itu dibutuhkan suatu sistem yang dapat mengatasi permasalah di Toko Kelontong XYZ.

Teknologi REST merupakan teknologi *web service* yang bersifat *stateless* artinya tidak ada status selama proses pertukaran data yang menggunakan protocol HTTP serta menggunakan prinsip REST (*Repsentational State Transfer*) dengan memanfaatkan media jaringan[4]. REST juga memiliki keunggulan dalam hal penggunaannya. Menurut terminologi MIS, teknologi *Client/Server* adalah paradigma baru untuk mengatur data yang dihadapi oleh organisasi modern. Istilah *Client/Server* digunakan untuk menggambarkan model komputasi untuk pengembangan sistem komputerisasi[5]. Model ini didasarkan pada distribusi fungsi antara dua jenis proses independen dan otonom: *Server* dan *Client*. [6].

Pemrograman REST didasarkan dari pemrograman berbasis objek, Pemrograman Berorientasi Objek (OOP) menggunakan susunan bahasa pemrograman alternatif dari sintak pemrograman prosedural lama (C, Pascal, dan sebagainya)[7]. Ini adalah metodologi yang pada dasarnya berpusat pada cara objek berkolaborasi untuk menyampaikan dan berbagi data [8]. REST API dapat menggunakan jenis – jenis *database* seperti *database relational*

Idealis: Indonesia Journal Information System
Volume 7, Nomor 1, Januari 2024
ISSN 2684-7280 (*online*)
Halaman 118-127
available online at http://jom.fti.budiluhur.ac.id/index.php/IDEALIS/index

atau *database document* (NoSQL)[9]. *Database relational* yang umum digunakan adalah MySQL, karena memiliki beberapa keuntungan seperti kecepatan, stabil dan jangkauan dan komunitas yang luas [3].

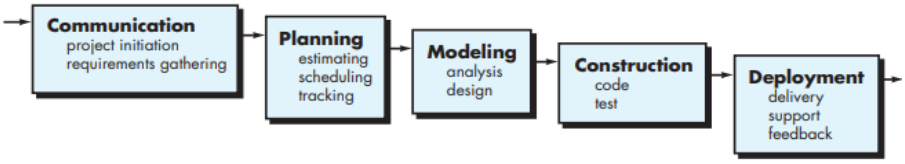
Beberapa jenis penelitian yang telah dilakukan untuk mengkaji penerapan REST API pada berbagai bidang. Penelitian tentang mebuat aplikasi pengolahan data barang dengan mencetak laporan barang masuk dan keluar dengan menerapkan metode REST API berbasis web [10]. Hasil penelitian tersebut terciptanya sistem pengolahan data barang dan cetak laporan barang masuk dan keluar dengan pengimplementasian REST API sesuai kebutuhan aplikasi yang dibuat. Penidas, Danny, dan Ade xxx membangun aplikasi dengan memanfaatkan RESTful *web service* untuk integrasi sistem dan distribusi data ke satu *database* dikarenakan beberapa unit di perusahaan memiliki sistem yang berbeda *platform* dan bahasa pemograman[11]. Aplikasi web modern memanfaatkan API secara ekstensif untuk memperbarui status UI sebagai respons terhadap peristiwa pengguna atau perubahan sisi *server*. Untuk aplikasi seperti itu, pengujian tingkat API dapat memainkan peran penting, di antara pengujian tingkat unit dan pengujian tingkat UI. Tujuan dari penelitian ini adalah menguji API dengan OPEN API untuk menguji kinerja API dan respon UI terhadap *request* dari *client*[7]. Penelitian yang membuat aplikasi berbasis *mobile* yang mudah digunakan di perangkat android atau *iphone* untuk mempermudah akses dokter, akses ke catatan kesehatan pasien, melalui sistem petukaran informasi kesehatan berbasis REST API dengan pengujian unit untuk memberi hasil yang optimal pada aplikasi xxx. Pemrograman database berbasis web dibangun dengan fondasi Bahasa pemrograman PHP yang didukung oleh HTML, CSS. Javascript sehingga dengan *database* MySQL adalah pondasi dasar membangun aplikasi berbasis web yang baik [12]. Berbeda dari penelitian – penelitian sebelumnya, penelitian ini membangun aplikasi berbasis web dengan kerangka kerja/*framework* VUE.JS yang berbasis *javascript.* Untuk akses ke *database* menggunakan teknologi REST API sehingga mendukung interopherabilitas yang baik.

Tujuan utama penelitian ini adalah menerapkan web service dengan metode REST API pada pembuatan aplikasi pengolahan transaksi dan laporan transaksi yang dapat diakses kapanpun, dimanapun dan digunakan oleh banyak perangkat dengan menggunakan transaksi Toko Kelontong XYZ sebagai format pembuatan aplikasi. Penelitian ini membuka peluang baru untuk memanfaatkan web service dengan metode REST API pada bidang lainnya. Manfaat penelitian ini adalah membuat aplikasi menerapkan menggunakan teknologi web service yang berbasis REST API. Aplikasi yang dibangun dapat memudahkan pengguna dalam melakukan pengolahan (penyimpanan, pengambilan, perubahan dan penghapusan data) transaksi dan laporan transaksi. Manfaat berikutnya adalah aplikasi yang dibangun diharapkan dapat menghemat waktu dalam pembuatan laporan transaksi dan dapat diakses dengan mudah.

2. METODE PENELITIAN

2.1 Metode Pengembangan Sistem

Model *waterfall* merupakan metode pengembangan sistem dengan sifat sistematis, yaitu berurutan dalam membangun aplikasi[13]. Disebut dengan *waterfall* terdiri dari tahapan – tahapan yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan. Gambar 1 menunjukan tahapan yang perlu dilalui pada *waterfall*.



Gambar 1. *Waterfall model*

2.1.1 Pengumpulan Data

Tahapan pengumpulan data adalah tahapan untuk mencari permasalahan dari pengguna, yang akan digunakan sebagai dasar perancangan sistem.

2.1.2 Analisis (*Requirement Definition*)

Semua kemungkinan kebutuhan sistem yang akan dikembangkan ditangkap dalam fase ini dan didokumentasikan dalam dokumen spesifikasi kebutuhan. Spesifikasi kebutuhan dari tahap pertama dipelajari pada tahap ini dan desain sistem disiapkan. Desain sistem ini membantu dalam menentukan persyaratan perangkat keras dan sistem serta membantu dalam menentukan arsitektur sistem secara keseluruhan perancangan.

2.1.3 Implementasi (*Implementation and Unit Testing*)

Dengan masukan dari desain sistem, sistem pertama kali dikembangkan dalam program kecil yang disebut unit, yang kemudian diintegrasikan pada tahap berikutnya. Setiap unit dikembangkan dan diuji fungsinya, yang disebut sebagai Pengujian Unit.

Idealis: Indonesia Journal Information System
Volume 7, Nomor 1, Januari 2024
ISSN 2684-7280 (*online*)
Halaman 118-127
available online at http://jom.fti.budiluhur.ac.id/index.php/IDEALIS/index

2.1.4 Pengujian (*Integration and System Testing)*

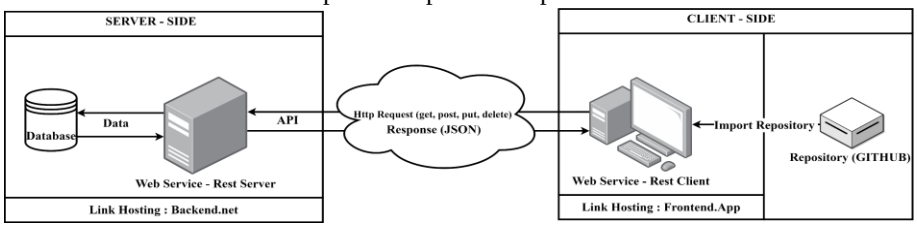
Semua unit yang dikembangkan pada tahap implementasi diintegrasikan ke dalam sistem setelah dilakukan pengujian terhadap setiap unit. Pasca integrasi, seluruh sistem diuji untuk setiap kesalahan dan kegagalan.

3. HASIL DAN PEMBAHASAN

3.1 Perancangan Sistem

3.1.1 Desain Arsitektur

Perancangan desain arsitektur terdapat REST Client dan REST Server yang terhubung melalui internet dan berinteraksi menggunakan API. Desain arsitektur aplikasi dapat dilihat pada Gambar 3.



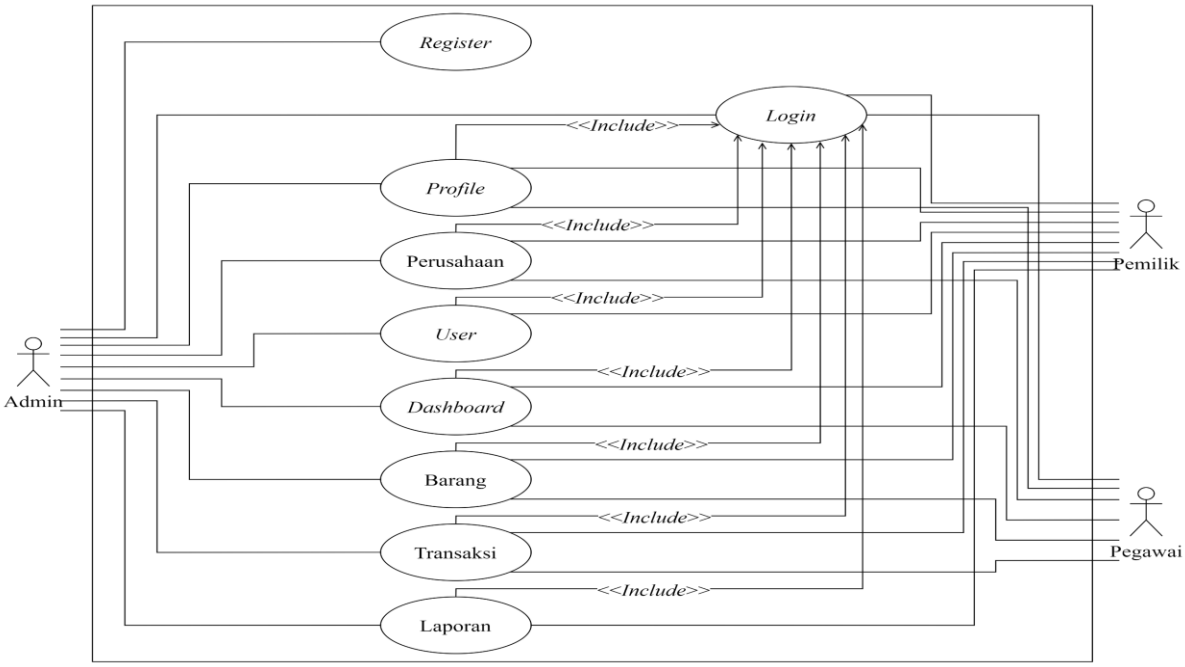
Gambar 3. Desain Arsitektur REST API

Rest Client akan melakukan request dengan mengirimkan HTTP request berupa GET, POST, PUT, DELETE ke REST Server melalui RESTful API yang terdapat Controller dan Model[14]. Sebelum request diterima, dilakukan autentikasi, kemudian Controller menerima request dari client, dan mengarahkan ke service yang sesuai untuk memproses request tersebut.

Pemrograman MVS dilakukan ketika Controller memanggil Model untuk melakukan method CRUD (Create, Read, Update, Delete) data pada database server. *Database handler* pada Model kemudian menerima data/kondisi (berhasil/tidak) dari database server dan mengembalikan nilai data/kondisi tersebut kepada Controller. Controller kemudian menyusun data JSON dan kode status dan mengirim kembali response kepada client.

3.1.2 *Use Case Diagram*

Use case diagram merupakan pemodelan untuk kelakuan sistem informasi yang akan dibuat Dalam aplikasi penjualan ini, terdapat tiga aktor, yaitu admin, pemilik dan pegawai[13]. Berdasarkan Gambar 4. *user* admin, pemilik, dan pegawai harus login ke sistem untuk mengakses fitur-fitur atau layanan yang telah disediakan. Deskripsi dari tiap *use case* dapat dilihat pada Tabel 1 dan Tabel 2.



Gambar 4. *Use Case Diagram*

Idealis: Indonesia Journal Information System
Volume 7, Nomor 1, Januari 2024
ISSN 2684-7280 (*online*)
Halaman 118-127
available online at http://jom.fti.budiluhur.ac.id/index.php/IDEALIS/index

Tabel 1. Alur *Use Case* Transaksi *Role Admin*

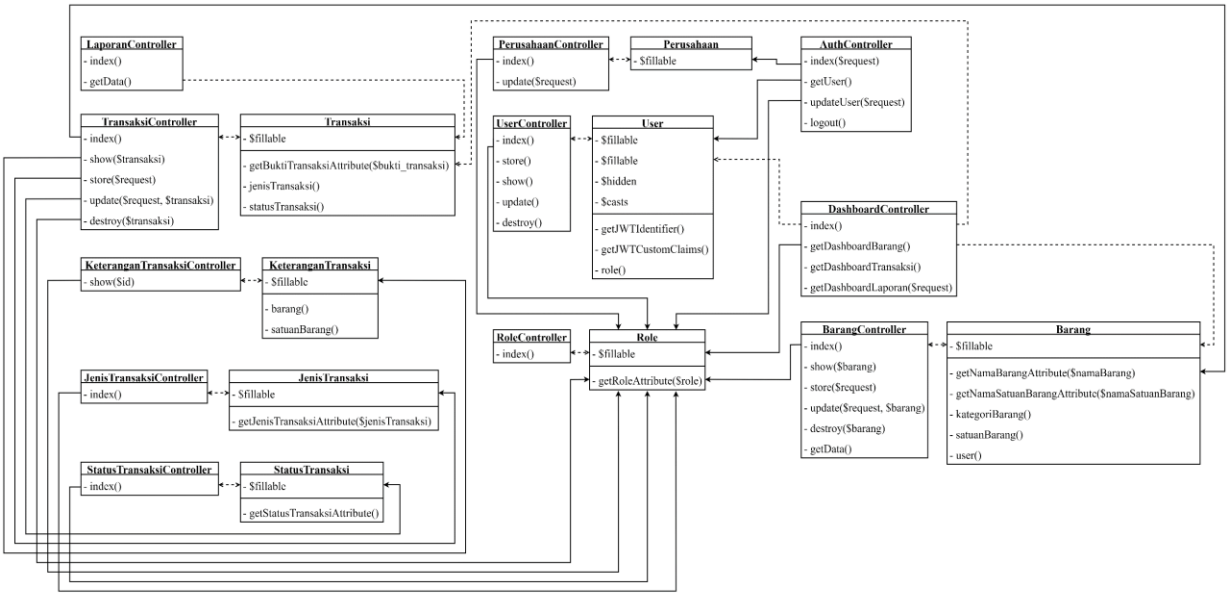
Nama Use Case	Transaksi
Aktor	*Admin*
Kondisi Awal	*Admin* sudah melakukan *login*
Aliran Kejadian	1. Menampilkan halaman transaksi. 2. *Admin* mengklik tombol tambah transaksi untuk menambah transaksi. 3. *Admin m*encari transaksi dengan mengisi pada *search bar* dan menekan tombol *search*. 4. *Admin* mengklik tombol *show* untuk menampilkan transaksi. 5. *Admin* mengklik tombol *edit* untuk mengubah transaksi. 6. *Admin* mengklik tombol *delete* untuk menghapus transaksi.
Kondisi Akhir	*Admin* masuk kedalam halaman transaksi dan dapat mengakses layanan untuk menampilkan data, mencari data, menambah data, menampilkan keseluruhan data, mengedit data, dan menghapus data transaksi.

Tabel 2. Alur *Use Case* dan Laporan *Role Admin*

Nama Use Case	Laporan
Aktor	*Admin*
Kondisi Awal	*Admin* sudah melakukan *login*
Aliran Kejadian	1. Menampilkan halaman laporan. 2. *Admin m*encari laporan dengan mengisi *start date* dan *end date* pada *search bar* dan menekan tombol *search*. 3. *Admin* mengklik tombol pdf untuk menuju halaman pdf dan mendownload.
Kondisi Akhir	*Admin* masuk kedalam halaman laporan dan dapat mengakses layanan untuk menampilkan data, mencari data, dan membuat laporan berformat pdf.

3.1.3 *Class Diagram*

Class diagram menggambarkan struktur sistem dalam istilah kelas dan objek [15]. Gambar merupakan semua struktur *class diagram* beserta relasi antar kelas. Proses ini dapat dilihat pada Gambar 5.



Gambar 5. *Class Diagram*

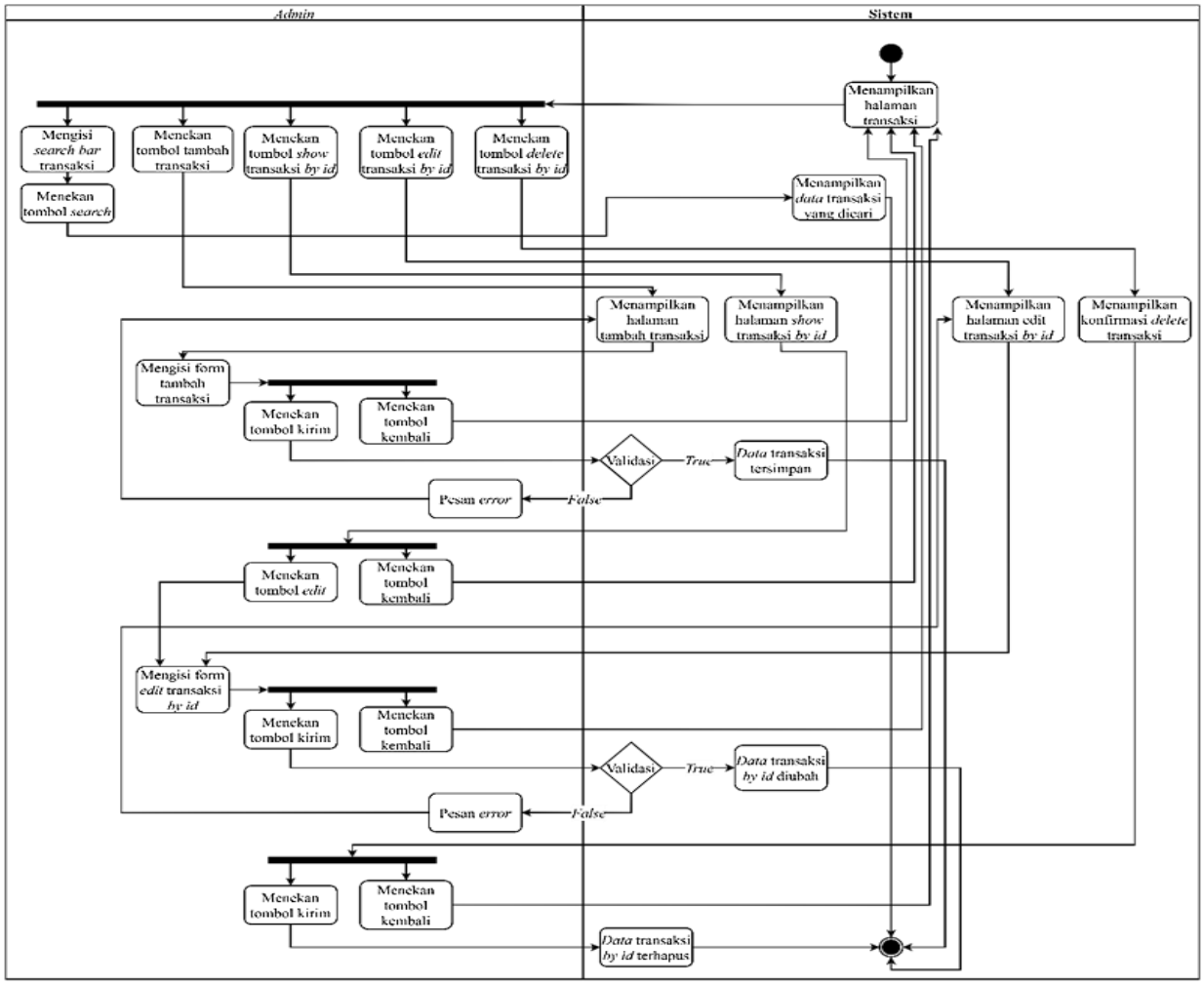
3.1.4 *Activity Diagram*

Diagram aktifitas menjelaskan tentang aliran kerja dari sistem atau proses bisnis atau menu yang ada pada perangkat lunak[13]. *Activity diagram* merupakan gambaran yang dilakukan pada setiap aktivitas-aktivitas pengguna pada sistem.

Idealis: Indonesia Journal Information System
Volume 7, Nomor 1, Januari 2024
ISSN 2684-7280 (*online*)
Halaman 118-127
available online at http://jom.fti.budiluhur.ac.id/index.php/IDEALIS/index

- *Activity Diagram* Transaksi Admin

User Admin juga dapat mencari transaksi dengan mengisi *search bar* yang telah diberikan setelah itu menekan tombol *search* maka transaksi yang dicari akan ditampilkan di halaman transaksi. Admin juga dapat menekan tombol tambah transaksi, untuk tombol tambah transaksi, admin diarahkan ke halaman transaksi lalu admin dapat mengisi *form* transaksi yang ada pada halaman transaksi lalu menekan tombol kirim atau menekan tombol kembali untuk kembali pada halaman transaksi. Gambar 6 adalah *activity diagram* transaksi.



Gambar 6. *Activity Diagram* Transaksi Admin

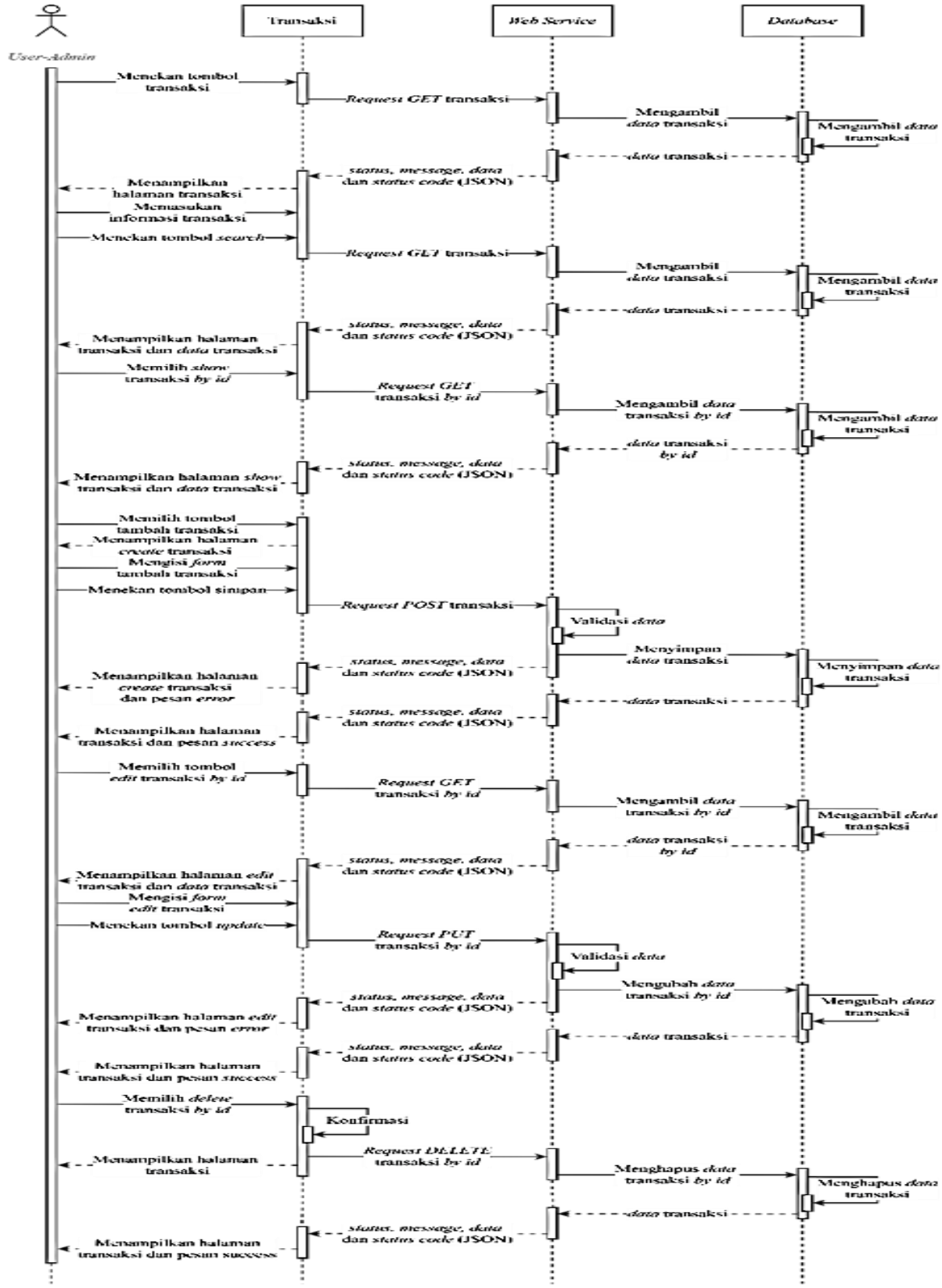
3.1.5 *Sequence Diagram*

Diagram urutan mewakili objek yang berpartisipasi dalam interaksi secara horizontal dan waktu secara vertical. *Sequence diagram* menggambarkan urutan aktivitas-aktivitas yang dilakukan oleh pengguna pada sistem dalam urutan waktu[13]. *Sequence* diagram sistem yang akan dibuat yaitu, *sequence* diagram login, *sequence* diagram *dashboard*, *sequence* diagram transaksi, *sequence* diagram kategori barang, *sequence* diagram satuan barang, *sequence* diagram barang, *sequence* diagram laporan, *sequence* diagram *user*, *sequence* diagram *profile*.

- *Sequence* Diagram Transaksi

Sequence diagram transaksi *actor* admin dapat dilihat pada Gambar 7. Pada diagram ini dapat dilihat bahwa, admin mendapatkan informasi mengenai transaksi yang telah tersimpan, mencari transaksi yang diinginkan, menampilkan seluruh data transaksi, membuat transaksi baru, mengedit transaksi yang sudah ada, dan menghapus transaksi.

Idealis: Indonesia Journal Information System
Volume 7, Nomor 1, Januari 2024
ISSN 2684-7280 (*online*)
Halaman 118-127
available online at http://jom.fti.budiluhur.ac.id/index.php/IDEALIS/index



Gambar 7. *Sequence* Diagram Transaksi Admin

Idealis: Indonesia Journal Information System
Volume 7, Nomor 1, Januari 2024
ISSN 2684-7280 (*online*)
Halaman 118-127
available online at http://jom.fti.budiluhur.ac.id/index.php/IDEALIS/index

3.2 Hasil Sistem

a. *Get* Transaksi Admin

Menampilkan halaman transaksi pada *user admin*. Halaman ini menampilkan *data* transaksi yang dipanggil menggunakan metode HTTP *get* pada *server*, halaman transaksi dapat dilihat pada Gambar 9.

Gambar 8. Antarmuka *Get* Transaksi *Admin*

- *Pseudocode Get* Transaksi *Admin*

Pseudocode get transaksi *user admin* digunakan untuk menampilkan *data* transaksi yang telah disimpan pada *server* kepada *client. Client* mengakses *route* API ini melalui alamat http://127.0.0.1:8000/api/admin/transaksi, *data* yang dihasilkan berupa *data* JSON.

b. *Get Transaksi Controller Admin*

Pada Gambar 9 adalah *pseudocode controller* transaksi *user admin.* Pada *controller* ini dibuat sebuah *property* yaitu transaksi dimana nilai dari *property* transaksi adalah pencarian transaksi menggunakan fungsi *when* dan *request* dengan kelas tanggal_transaksi, jenis_transaksi, dan status_transaksi sebagai parameter pencarian data berdasarkan perusahaan_id sama dengan perusahaan_id *user* yang login, dengan menggunakan fungsi jenisTransaksi dan statusTransaksi dari *model*, data diurutkan berdasarkan tanggal_transaksi dan data menggunakan pagination untuk membatasi jumlah data yang dikeluarkan pada 1 halaman.

```
public function index()
{
    $transaksi = Transaksi::when(request()->search, fn($transaksi) =>
        $transaksi->where('perusahaan_id', $auth->perusahaan_id)
        ->where('tanggal_transaksi', 'like', '%'. request()->search . '%')
        ->orWhereHas('jenisTransaksi', fn($transaksi) => $transaksi->where('perusahaan_id', $auth->perusahaan_id)
        ->where('nama_jenis_transaksi', 'like', '%'. request()->search .'%'))
        ->orWhereHas('statusTransaksi', fn($transaksi) => $transaksi->where('perusahaan_id', $auth->perusahaan_id)
        ->where('nama_status_transaksi', 'like', '%'. request()->search .'%'))
    )->with('jenisTransaksi', 'statusTransaksi')->where('perusahaan_id', $auth->perusahaan_id)
    ->orderBy('tanggal_transaksi', 'desc')->paginate(5);
    return responseSuccess(true, 'List Data Transaksi', $transaksi, Response::HTTP_OK);
}
```

Gambar 9. *Pseudocode Get* Transaksi *Controller Admin*

c. *Get Transaksi Model Admin*

Gambar 10 adalah *pseudocode model* transaksi. Pada *model* transaksi ini dibuat sebuah *property fillable* yang digunakan pada saat ada tambah *data*, dan perubahan *data* hanya kelas-kelas yang ada pada *property fillable* yang boleh ditambah atau diubah. Pada *model* transaksi ini juga terdapat fungsi jenisTransaksi sebagai penghubung atau *foreign key* antar 2 tabel yaitu: Tabel transaksis dan Tabel jenis_transaksis. Fungsi statusTransaksi sebagai penghubung atau *foreign key* antar 2 tabel yaitu: Tabel transaksis dan Tabel status_transaksis. Fungsi getBuktiTransaksiAttribut digunakan pada saat memanggil *data* berupa file atau gambar.

```
class Transaksi extends Model
{
    use HasFactory, uuid;

    protected $fillable = [
        'tanggal_transaksi', 'jenis_transaksi_id', 'nominal_transaksi', 'status_transaksi_id', 'bukti_transaksi', 'perusahaan_id'
    ];
    public function jenisTransaksi()
    {
```

Idealis: Indonesia Journal Information System
Volume 7, Nomor 1, Januari 2024
ISSN 2684-7280 (*online*)
Halaman 118-127
available online at http://jom.fti.budiluhur.ac.id/index.php/IDEALIS/index

```
    return$this->belongsTo(JenisTransaksi::class, 'jenis_transaksi_id')->select('id', 'nama_jenis_transaksi');
}
public function statusTransaksi()
{
    return$this->belongsTo(StatusTransaksi::class,'status_transaksi_id')->select('id', 'nama_status_transaksi');
}
    public function getBuktiTransaksiAttribute($bukti_transaksi)
{
    return asset('storage/bukti_transaksi/' . $bukti_transaksi);
}}
```

Gambar 10. *Pseduocode* Transaksi *Model*

d. *Get Transaksi View Admin*

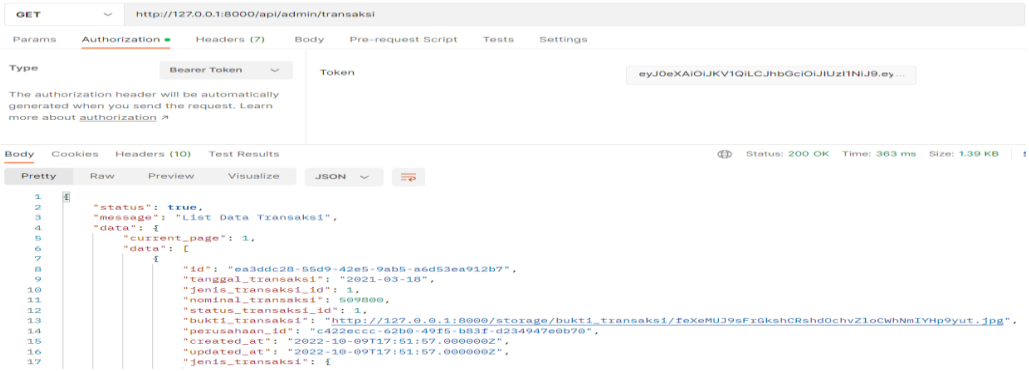
Gambar 11 adalah *pseudocode* dari *view* transaksi *user admin.* Pada *view* halaman transaksi *user admin* menggunakn proses *async/await* untuk mengambil *data* transaksi pada *server* menggunakan *method get* dengan alamat /api/admin/transaksi lalu data akan di-*return* ke halaman transaksi *user admin*.

```
async asyncData({ $axios, query })
{
    const transaksi = await $axios.$get(`/api/admin/transaksi`)
    return {
        `transaksis`: transaksi.data.data
    }}
```

Gambar 11. *Pseduocode* Transaksi *View Admin*

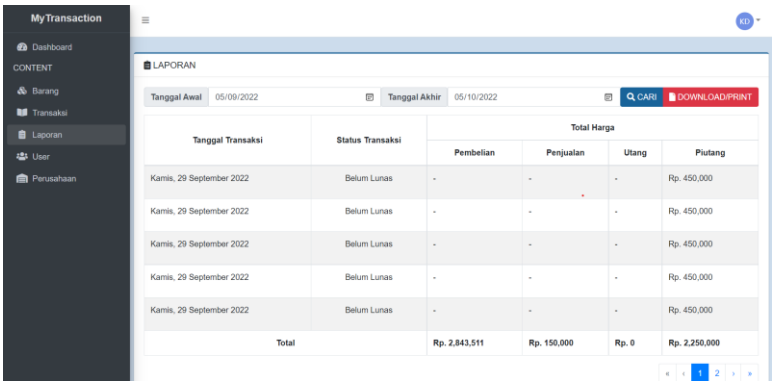
e. Pengujian Get Transaksi Admin

Gambar 12 adalah hasil pengujian melalui *postman*. *Pseudocode* GET transaksi admin adalah hasil *data* JSON *get* transaksi *user admin* yang menampilkan semua data transaksi yang ingin dicari atau tidak dicari menggunakan aplikasi *Postman* melalui alamat http://127.0.0.1:8000/api/admin/transaksi. Dengan *response time* 363 ms dan status *code 200* (*success*).



Gambar 12. Pengujian Postman Get Transaksi Admin

Gambar 13 adalah tampilan dari halaman laporan pada *user admin*. Halaman ini menampilkan *data* transaksi yang dipanggil menggunakan metode HTTP *get* pada *server*.



Gambar 13. Antarmuka Get Laporan Admin

Idealis: Indonesia Journal Information System
Volume 7, Nomor 1, Januari 2024
ISSN 2684-7280 (*online*)
Halaman 118-127
available online at http://jom.fti.budiluhur.ac.id/index.php/IDEALIS/index

f. *Get Laporan Model*

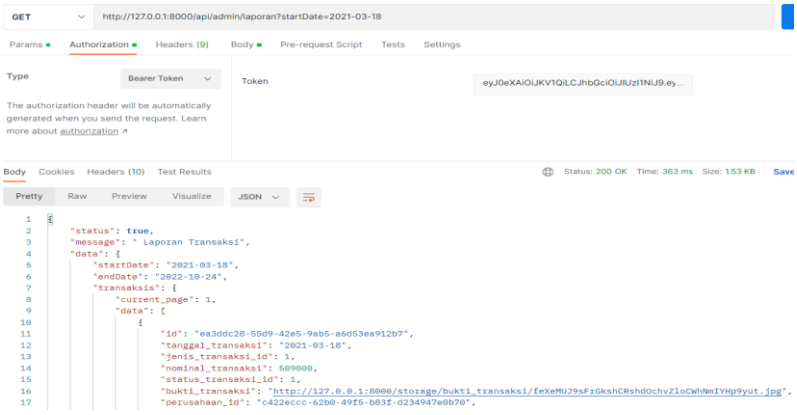
Gambar 14 adalah *pseudocode model* transaksi. Pada *model* transaksi ini dibuat sebuah *property fillable* yang digunakan pada saat ada tambah *data*, dan perubahan *data* hanya kelas-kelas yang ada pada *property fillable* yang boleh ditambah atau diubah. Pada *model* transaksi ini juga terdapat fungsi jenisTransaki sebagai penghubung atau *foreign key* antar 2 tabel yaitu: tabel transaksis dan tabel jenis_transaksis. Fungsi statusTransaksi sebagai penghubung atau *foreign key* antar 2 tabel yaitu: tabel transaksis dan tabel status_transaksis. Fungsi getBuktiTransaksiAttribut digunakan pada saat memanggil *data* berupa file atau gambar.

```
class Transaksi extends Model
{
   use HasFactory, uuid;
   protected $fillable = [
      'tanggal_transaksi', 'jenis_transaksi_id', 'nominal_transaksi', 'status_transaksi_id', 'bukti_transaksi', 'perusahaan_id' ];
   public function jenisTransaksi()
   {       return $this->belongsTo(JenisTransaksi::class, 'jenis_transaksi_id')->select('id', 'nama_jenis_transaksi');  }
   public function statusTransaksi()
   {       return $this->belongsTo(StatusTransaksi::class, 'status_transaksi_id')->select('id', 'nama_status_transaksi');   }
      public function getBuktiTransaksiAttribute($bukti_transaksi)
   {
      return asset('storage/bukti_transaksi/' . $bukti_transaksi);
   }
}
```

Gambar 14. *Pseduocode* Transaksi *Model Admin*

g. Pengujian Get Laporan Admin

Gambar 15 adalah hasil pengujian melalui *postman*. Hasil *data* JSON *get* laporan transaksi *user admin* yang menampilkan data transaksi yang dicari menggunakan aplikasi *Postman* melalui alamat http://127.0.0.1:8000/api/admin/laporan. Dengan response time 363 ms dan status code 200 (*success*).



Gambar 15. Pengujian *Postman Get* Laporan *Admin*

3.3 Pengujian Sistem

Pengujian dilakukan dengan metode *blackbox testing*. *Blackbox testing* merupakan metode *equivalence portioning* yang dilakukan untuk menguji suatu fitur-fitur yang ada pada sistem. Keseluruhan pengujian beserta hasil dari *black box testing* dapat dilihat pada Tabel 3 dibawah ini:

Tabel 3. Pengujian *Blackbox*

No	URI API Pengujian	Skenario Pengujian	Hasil yang Diharapkan	Hasil Uji
1.	http://localhost:3000/api/admin/transaksi	Kirim *request* dengan HTTP *Method Get*	*Success* dengan *status code* 200	Berhasil
2.	http://localhost:3000/api/admin/keteranganTransaksi/{id}	Kirim *request* dengan HTTP *Method Get* beserta *id* transaksi	*Success* dengan *Status Code* 200	Berhasil
3.	http://localhost:3000/api/admin/laporan?startDate=’...’&endDate=’...’	Kirim *request* dengan HTTP *Method Get* beserta *startDate* dan *endDate* untuk mencari data	*Success* dengan *status code* 200	Berhasil

Idealis: Indonesia Journal Information System
Volume 7, Nomor 1, Januari 2024
ISSN 2684-7280 (*online*)
Halaman 118-127
available online at http://jom.fti.budiluhur.ac.id/index.php/IDEALIS/index

berdasarkan tanggal_transaksi
bersifat *optional*

4. KESIMPULAN

Aplikasi yang dibangun dengan teknologi REST API telah berhasil mengurangi kesalahan *entry* data keuangan toko dan meningkatkan interoperabilitas karena respon dari REST Server berbentuk JSON yang dapat di*consume* oleh *platform* lain yang dalam penelitian ini menggunakan *framework* VUE.JS. Aplikasi yang dibangun menggunakan otentikasi dengan teknologi JSON Web Token (JWT) yang menghasilkan proses otentikasi lebih aman karena otentikasi menggunakan token yang dihasilkan oleh JWT. Hasil uji dengan *blackbox testing* yang berhasil dilakukan pada URL dengan status *code* 200 yang artinya sukses.

DAFTAR PUSTAKA

[1] H. Gunawan, H. Soetanto, F. T. Informasi, U. B. Luhur, and P. Aset, "ANALISA DAN IMPLEMENTASI WEB SERVICE MENGGUNAKAN WEB SERVICE ANALYSIS AND IMPLEMENTATION USING THE," vol. 2, no. September, pp. 2066–2073, 2023.
[2] V. Gunawan, *Gunawan, V. (2020). 1. APLIKASI INVENTORY BERBASIS WEB MENGGUNAKAN FRAMEWORK CODEIGNITER DENGAN WEB SERVICE REST API (Doctoral dissertation, Universitas Buddhi Dharma)*. 2020. [Online]. Available: http://repositori.buddhidharma.ac.id/id/eprint/542
[3] Andy Oram, Ed., *MySQL in a Nutshell a Desktop Quick Reference*, Second Edi. y O'Reilly Media, Inc, 2008.
[4] C. A. Győrödi, D. V. Dumşe-Burescu, D. R. Zmaranda, R. Győrödi, G. A. Gabor, and G. D. Pecherle, "Performance analysis of nosql and relational databases with couchdb and mysql for application's data storage," *Appl. Sci.*, vol. 10, no. 23, pp. 1–21, 2020, doi: 10.3390/app10238524.
[5] A. A. Prayogi, M. Niswar, Indrabayu, and M. Rijal, "Design and Implementation of REST API for Academic Information System," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 875, no. 1, 2020, doi: 10.1088/1757-899X/875/1/012047.
[6] P. A. M. Serrano and J. J. S. Onate, "Integration of RESTFul API to Student Information System for Secured Data Sharing and Single Sign-on," *2021 IEEE 13th Int. Conf. Humanoid, Nanotechnology, Inf. Technol. Commun. Control. Environ. Manag. HNICEM 2021*, 2021, doi: 10.1109/HNICEM54116.2021.9731898.
[7] R. Yandrapally, S. Sinha, R. Tzoref-Brill, and A. Mesbah, "Carving UI Tests to Generate API Tests and API Specification," pp. 1971–1982, 2023, doi: 10.1109/icse48619.2023.00167.
[8] de Rosal Ignatius Moses Setiadi, A. F. Najib, E. H. Rachmawanto, C. A. Sari, M. K. Sarker, and N. Rijati, "A comparative study MD5 and SHA1 algorithms to encrypt REST API authentication on mobile-based application," *2019 Int. Conf. Inf. Commun. Technol. ICOIACT 2019*, pp. 206–211, 2019, doi: 10.1109/ICOIACT46704.2019.8938570.
[9] S. B. Cleveland *et al.*, "Tapis API Development with Python: Best Practices in Scientific REST API Implementation: Experience implementing a distributed Stream API," *ACM Int. Conf. Proceeding Ser.*, no. August, pp. 181–187, 2020, doi: 10.1145/3311790.3396647.
[10] Jamal *et al.*, "Design and Implementation of Web Application for Attendance List of Lecturers Using Codeigniter and Bootstrap Framework," *J. Phys. Conf. Ser.*, vol. 1807, no. 1, 2021, doi: 10.1088/1742-6596/1807/1/012030.
[11] D. Darmawan, F. Satrya Fajar Kusumah, S. Hidayat Al Ikhsan, U. K. Ibn Khaldun Bogor Jl Sholeh Iskandar Raya Km, and K. Badak, "Web Service Untuk Transaksi Data Pada Aplikasi Fasilitas Keuangan Dengan Metode REST," *J. Sains Komput. Inform. (J-SAKTI*, vol. 5, no. 2, pp. 852–865, 2021.
[12] S. Kleijnen and S. Raju, "Web Services architecture," *Queue*, vol. 1, no. 1, pp. 38–46, 2003, doi: 10.1145/637958.637961.
[13] F. N. Hasanah, *Buku Ajar Rekayasa Perangkat Lunak*. 2020. doi: 10.21070/2020/978-623-6833-89-6.
[14] M. M. Hidayat, R. Dimas Adityo, and A. Siswanto, "Design of Restaurant Billing System (E Bill Resto) by Applying Synchronization of Data Billing in Branch Companies to Main Companies Based on Rest API," *Proceeding - ICoSTA 2020 2020 Int. Conf. Smart Technol. Appl. Empower. Ind. IoT by Implement. Green Technol. Sustain. Dev.*, 2020, doi: 10.1109/ICoSTA48221.2020.1570615039.
[15] N. E. Helwig, S. Hong, and E. T. Hsiao-wecksler, *Service Design Patterns Fundamental Design Solution For SOAP WSDL And RESTful Web Service*, 1st ed. New Jersey: Pearson Education, 2012.