Penerapan RESTful API untuk Membangun Program Pembayaran Piutang Menggunakan Otentikasi OAuth 2.0

Nina Wulandari[1], Argo Wibowo[2], Budi Susanto[3]
[12]*Sistem Informasi, Universitas Kristen Duta Wacana*
Jl. Dr. Wahidin Sudirohusodo No 5-25 Klitren, Yogyakarta
nina.wulandari@si.ukdw.ac.id
argo@staff.ukdw.ac.id
budsus@ti.ukdw.ac.id

Abstract*— Retail business competition which is increasingly fast requires businesses to have a product sales strategy that is able to compete. Amigo Group convert this strategy by providing account receivable services for customers. The study "* **Penerapan RESTful API untuk Membangun Program Pembayaran Piutang Menggunakan Otentikasi OAuth 2.0", focused on how to use RESTful to provide the transaction API for accounts receivable payments, and the use of OAuth 2.0 as an authentication method, as well as load testing of API.***

The system design has been done with UML modeling language in the form of activity diagrams and sequence diagrams. Programming was done using the Laravel framework and MySQL database. Load testing was performed using the Locust application.

These designs produced Back End functions that can be used on web and mobile platforms. As a result of load testing, the use of the OAuth 2.0 method is more advantageous than using Basic Auth. This is shown by the response time and request per second which is more stable in the use of OAuth 2.0.

Key Wordi*— *Payment, Acoounts Receivable, RESTful, OAuth 2.0, Load Testing

Intisari— Persaingan bisnis ritel yang semakin pesat mengharuskan pelaku usaha untuk memiliki strategi penjualan produk yang mampu bersaing. Amigo Group menerjemahkan strategi tersebut dengan menyediakan layanan bon untuk pelanggan. Penelitian "Penerapan RESTful API untuk Membangun Program Pembayaran Piutang Menggunakan Otentikasi OAuth 2.0", memiliki rumusan masalah baga imana pemanfaatan RESTful untuk menyediakan kebutuhan Back End transaksi pembayaran piutang, dan penggunaan OAuth 2.0 sebagai metode otentikasi, serta pengujian terhadap beban yang mampu dijalankan.

Perancangan sistem dilakukan dengan Bahasa pemodelan UML (Unified Modeling Language) berupa activity diagram dan sequence diagram. Pemrograman dilakukan menggunakan framework Laravel dan basis data MySQL. Pengujian beban dilakukan menggunakan aplikasi Locust.

Hasil perancangan tersebut menghasilkan fungsi-fungsi Back End yang dapat digunakan pada platform web dan mobile. Adapun hasil dari pengujian beban, menunjukkan bahwa penggunaan metode otentikasi OAuth 2.0 lebih menguntungkan dibanding penggunaan Basic Auth. Hal tersebut ditunjukkan oleh response time dan request per detik yang lebih stabil pada penggunan OAuth 2.0.

***Kata Kunci*— Pembayaran, Piutang, RESTful, OAuth 2.0, Pengujian Beban**

I. PENDAHULUAN

Menurut Hery [1], perusahaan adalah sebuah organisasi yang beroperasi dengan tujuan menghasilkan keuntungan, dengan menjual barang atau jasa kepada para pelanggannya. Kemampuan perusahaan untuk meraih keuntungan dibatasi oleh berbagai faktor, seperti sumber daya manusia, sumber pendanaan, dan strategi. Strategi yang dimaksud adalah strategi penjualan, pemasaran, dan sebagainya. Amigo Group merupakan perusahaan dagang di bidang ritel fashion yang menjual produk pakaian untuk anak, remaja, dan dewasa, sehingga strategi penjualan produk sangat mempengaruhi operasional dan keuntungan yang didapat perusahaan.

Menurut Utomo [2], perkembangan bisnis ritel yang semakin pesat mengakibatkan adanya persaingan yang ketat di antara pelaku industri, sehingga pelaku industri dituntut untuk memberikan layanan terbaik agar tetap bertahan. Amigo Group sebagai toko ritel *fashion* menerjemahkan tantangan tersebut dengan menyediakan layanan piutang atau bon untuk pelanggan. Layanan ini membolehkan pelanggan untuk membayar barang yang diambil dari toko dengan cara mencicil dalam tenggat yang ditentukan.

Layanan bon merupakan layanan Amigo Group yang sudah dijalankan sejak 2009, dan tercatat ada puluhan transaksi dengan nilai puluhan juta yang dilakukan setiap harinya.

Pengelolaan terhadap pembayaran piutang yang telah dilakukan belum optimal sehingga perlu diubah. Pelanggan harus datang ke toko untuk dapat mengetahui jumlah cicilan dan atau barang yang belum dilunasi. Dengan kata lain, informasi diperoleh hanya melalui petugas kasir. Penanggung jawab bon (karyawan) sering kali tidak hafal siapa saja downline yang dibawa oleh upline yang menjadi tanggung jawabnya. Hal tersebut disebabkan karena downline hanya berinteraksi dengan upline saja dalam pemberian limit belanja.

Penyediaan REST API mampu menjadi solusi untuk mengoptimalkan layanan kepada pelanggan. REST API menjadi antarmuka yang dapat digunakan oleh perusahaan

untuk mengembangkan aplikasi yang dapat diakses dari berbagai platform. Hal tersebut berkontribusi positif terhadap skalabilitas dari pengembangan sistem di perusahaan.

II. LANDASAN TEORI

A. Studi Pustaka

Penelitian yang dilakukan oleh Vijay Surwase [3] menentukan modeling language mana yang lebih baik dipakai dalam implementasi REST API. Adapun modeling language sendiri adalah Bahasa yang digunakan untuk mendeskripsikan RESTful API. Perbandingan dilakukan dengan memaparkan kelebihan dan kekurangan masing-masing modeling language serta melihat banyaknya komunitas pengguna modeling language tersebut di kalangan developer. Kesimpulan yang dapat ditarik adalah bahwa setiap modeling language memiliki kelebihan dan kekurangan dan setiap modeling memiliki format umum yang ditentukan. Pada penelitian ini digunakan modeling language RAML, yaitu pada penjabaran kebutuhan fungsi.

Penelitian yang dilakukan oleh Kusuma, Susanto, dan Mulyono [4] dilakukan untuk mengatasi permasalahan pengelolaan retribusi parkir yang kurang transparan, dan berpengaruh pada retribusi daerah. Peneliti membangun sistem pembayaran retribusi parkir berupa aplikasi android Markir, dengan mengimplementasikan RESTful API dan menerapkan OAuth 2.0 untuk mengamankan setiap transaksi. Hasil aplikasi diuji dan menunjukkan bahwa setiap tahapan sesuai dengan standar OAuth 2.0 dan data transaksi lebih aman jika dibandingkan dengan penggunaan protocol keamanan standar lainnya .

B. Piutang

Piutang adalah adalah tagihan yang ditujukan kepada individu maupun perusahaan lain yang akan diterima dalam bentuk kas [5]. Piutang pada umumnya dikelompokkan menjadi piutang dagang/usaha, piutang wesel, dan lain-lain. Piutang dagang berasal dari penjualan barang dagangan dan jasa secara kredit dalam operasi usaha normal [5]. Piutang wesel adalah klaim yang dibuktikan dengan instrument kredit secara formal. Instrument kredit ini menyaratkan kepada debitor untuk membayar di masa mendatang pada tanggal yang telah ditentukan [5]. Piutang lain-lain merupakan piutang yang tidak berkaitan dengan usaha, seperti pinjaman kepada karyawan maupun pinjaman kepada pihak lain [5].

Berdasar pada jenis-jenis piutang di atas, transaksi piutang di Amigo Group termasuk ke dalam piutang dagang/usaha, yaitu merupakan transaksi oleh pelanggan yang pembayarannya dilakukan dengan mencicil atau tidak sekaligus lunas saat barang diterima, dalam tenggat yang sudah ditentukan. Ketentuan terkait transaksi piutang diatur oleh manajemen dan tertuang dalam buku pedoman ketentuan bon yang dimiliki Amigo Group.

C. REST API

REST memiliki kepanjangan Respresentational State Transfer. Istilah ini dicetuskan oleh Roy Thomas Fielding dalam disertasinya yang berjudul “Architectural Styles and the Design of Network-based Software Architectures” pada tahun 2000. REST API merupakan gaya arsitektural yang mendefinisikan aturan-aturan untuk membuat web service. REST API dapat dispesifikasikan lagi ke dalam beberapa modeling language yang akan merepresentasikan format dari JSON/XML.

D. HTTP Request Method

HTTP merupakan protokol yang mengatur terjadinya komunikasi antara client dan server, sebagai contoh adalah peramban mampu menampilkan dan mengirim data melalui internet. *Request method* berfungsi untuk mengidentifikasi operasi yang akan dipanggil [3].

Method	Semantics
GET	Retrieve the complete state of a resource, in some representational form
HEAD	Retrieve the metadata state of a resource
PUT	Insert a new resource into a store or update an existing, mutable resource
DELETE	Remove the resource from its parent
OPTIONS	Retrieve metadata that describes a resource’s available interactions

Gambar.1. HTTP request method

E. JSON

JSON adalah singkatan dari JavaScript Object Notation, merupakan representasi data yang dipertukarkan antara server dan klien [6]. Data tersebut terdiri dari attributes atau key dan value.

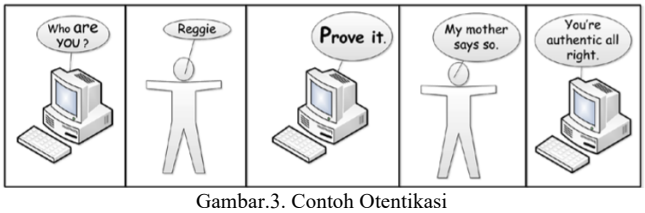
```
{
"kd_customer": "33000333",
"alias": "dss",
"migrasi": null,
"ktp": "22233344411122",
"nama": "MAWAR MERAH",
"alamat": "SMP N I KR.ANOM KALORAN RT 01 RW 00",
"tempat_lahir": "KLATEN",
"tgl_lahir": "1959-08-27",
"jns_kelamin": "L"
}
```

Gambar.2. Contoh JSON

F. Otentikasi dan Otorisasi

Otentikasi dan otorisasi merupakan 2 hal yang berbeda, sebagaimana dijelaskan sebagai berikut:

Otentikasi berfungsi untuk memverifikasi identitas dari pengguna [7]. Otentikasi membuktikan bahwa seseorang atau sesuatu memang benar adalah dia. Pada umumnya otentikasi dilakukan dengan memasukkan *username* dan *password* [8].

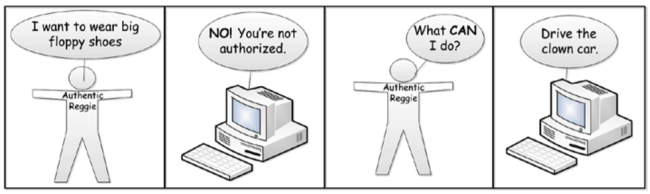


Gambar.3. Contoh Otentikasi

Otorisasi menentukan apa yang dapat (berhak) dilakukan oleh pengguna yang sudah terotentikasi [7]. Otorisasi melakukan proses verifikasi bahwa pengguna

JUTEI Edisi Volume.5 No.1 April 2021
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2021.51.230

memiliki hak untuk melakukan suatu aksi, seperti membaca dokumen.



Gambar.4. Contoh Otorisasi

G. *OAuth 2.0*

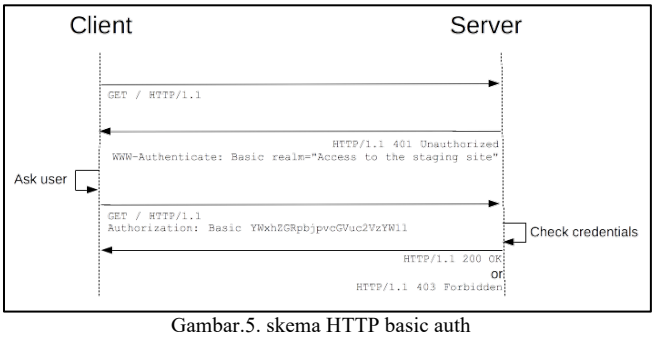
OAuth 2.0 adalah protokol yang memungkinkan aplikasi untuk mengakses data pengguna secara aman tanpa pengguna perlu memasukkan kata sandinya [8]. Menurut Bihis [9], OAuth 2.0 merupakan protokol yang memungkinkan pihak-pihak (aplikasi) yang berbeda untuk berbagi informasi dan sumber daya dengan cara yang aman dan andal. Secara umum penggunaan protokol OAuth 2.0 melibatkan dua hal [9]:

1) Federated Identity: Bagian dari Identity Management yang mengizinkan seorang pengguna untuk masuk ke sebuah aplikasi menggunakan akun aplikasi lain. Contoh dari federated identity adalah aplikasi Spotify yang memperbolehkan pengguna untuk login menggunakan akun Facebook.

2) Delegated Authority: Otoritas yang diperoleh dari aplikasi lain, yaitu mengizinkan sebuah layanan untuk mengakses sumber daya pada layanan lain atas nama pengguna. Contoh dari delegated authority adalah membagikan lagu yang sedang didengarkan di Spotify ke status media sosial.

H. *Basic Authentication*

Basic Authentication merupakan skema di mana klien perlu mengotentikasi dirinya menggunakan sebuah id pengguna dan sebuah kata sandi untuk setiap tindakan [9]. Id dan kata sandi tersebut akan diencode dan dikirim melalui HTTP Header.



Gambar.5. skema HTTP basic auth

I. *Load Testing*

Menurut Enrile [10], Load Testing adalah proses menempatkan permintaan pada suatu sistem dan mengukur responsnya; yaitu menentukan berapa banyak volume yang dapat ditangani sistem.

Load Testing diperlukan untuk membuat simulasi akses ke aplikasi secara simultan. Cara ini lebih menguntungkan daripada mengundang sejumlah orang untuk mengakses aplikasi secara bersamaan [11].

J. *Upline dan Downline*

1) Upline / Penanggungjawab Bon: pelanggan yang melakukan transaksi bon untuk keperluan pribadi atau untuk dijual kembali kepada anggota bon.
2) Downline / Anggota Bon: orang yang melakukan transaksi bon untuk keperluan pemakaian sendiri melalui upline.

III. METODOLOGI PENELITIAN

Pada bab ini akan dibahas mengenai tahapan-tahapan dari penelitian yang dilakukan, yaitu adalah pengumpulan data; perancangan sistem, berupa *activity diagram*, *sequence diagram*, dan kebutuhan fungsi yang akan dibangun; rancangan masukan, dan rancangan hasil.

A. *Pengumpulan Data*

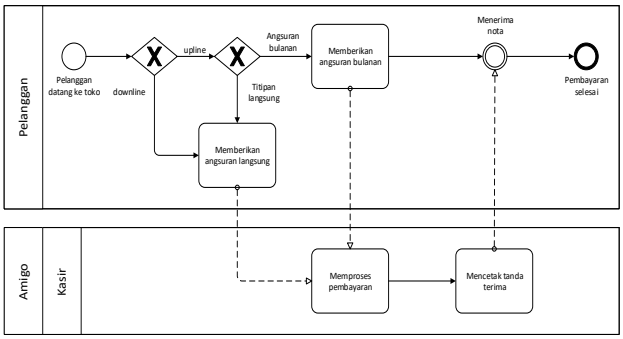
Data yang digunakan dalam penelitian ini adalah data pelanggan, data karyawan, dan data pencatatan piutang. Adapun metode pengumpulan data yang dilakukan adalah sebagai berikut:

1) *Dokumentasi:* Data didapatkan dalam bentuk berkas, yaitu berkas basis data dan berkas SOP piutang. Berkas basis data berisi data pelanggan, karyawan, dan transaksi piutang yang dilakukan. Berkas SOP berisi terminologi dan ketentuan-ketentuan terkait piutang.
2) *Wawancara tidak terstruktur:* Wawancara dilakukan untuk mendapatkan pemahaman yang lebih baik terkait penerapan ketentuan piutang. Wawancara dilakukan terhadap karyawan dengan menganalisis berkas basis data.

B. *Rancangan Sistem*

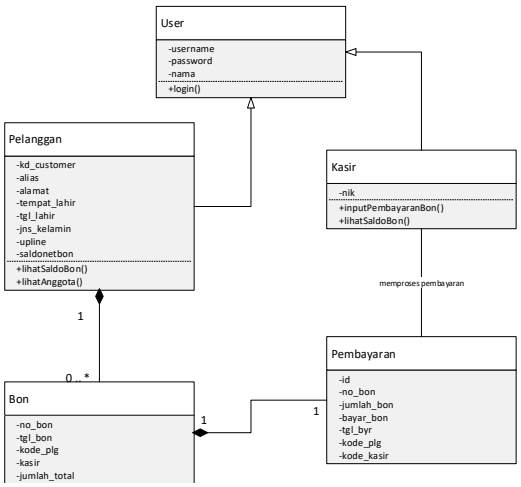
UML (Unified Modeling Language) digunakan untuk menggambarkan rancangan sistem yang dibangun. Beberapa jenis UML yang dipakai telah disesuaikan dengan kebutuhan dan tujuan visualisasi.

1) Proses Pembayaran Piutang : Proses pembayaran piutang di Amigo digambarkan seperti pada Gambar 6.



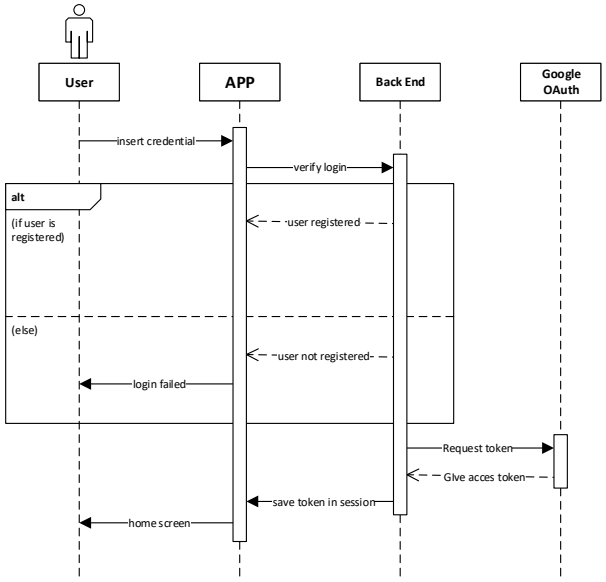
Gambar.6. Activity diagram pembayaran piutang

2) Class Diagram: Menggambarkan kelas yang terlibat dalam pembayaran piutang beserta dengan metode yang dilakukan tiap kelas. Lihat Gambar 7.



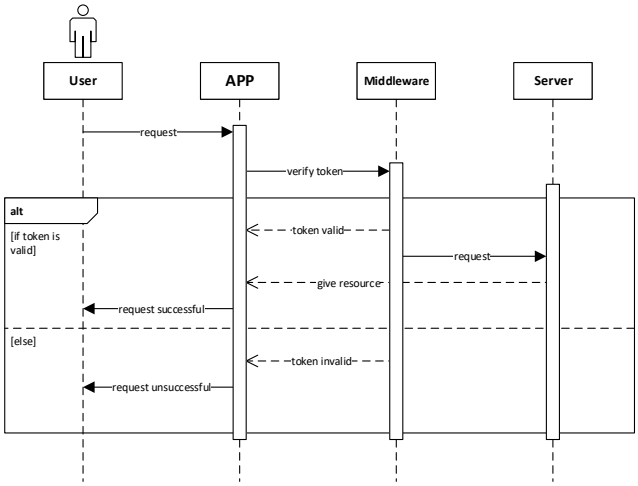
Gambar.7. Class diagram pembayaran piutang

3) Sequence Diagram Log in: Menggambarkan alur *log in* ke dalam sistem.



Gambar.8. Sequence diagram proses login

4) Sequence Diagram Kirim Request: Gambar 9 menampilkan urutan proses saat mengirim *request* ke server.



Gambar.9. Sequence diagram mengirim request

5) Kebutuhan Fungsi: Daftar fungsi yang dibangun adalah sebagai berikut:

TABEL I
DAFTAR KEBUTUHAN FUNGSI

Fungsi	Method	End Point	Parameter Input
Get upline	GET	root/api/o/uplines	-
Get anggota bon	GET	root/api/o/uplines/{upline}	Kode upline
Get data bon	GET	root/api/o/bons	-
Get data bon berdasar kode bon	GET	root/api/o/bons/byNo/{no_bon}	Kode bon
Get data bon berdasar pelanggan	GET	root/api/o/bons/{kd_customer}	Kode pelanggan
Get data bon berdasar upline	GET	root/api/o/bons/byup/{upline}	Kode upline
Get data bon belum lunas berdasar upline	GET	root/api/o/bons/unlunas/{upline}	Kode upline
Get data pembayaran bon berdasar pelanggan	GET	root/api/o/bons/dtBayar/{kd_customer}	Kode pelanggan
Post angsuran langsung	POST	root/api/o/bons/{kd_customer}	Kode pelanggan, jumlah bayar, kode kasir
Post pelunasan nota bon	POST	root/api/o/bons/lunas/{no_bon}	Kode bon

1. Fungsi Pelanggan
   Fungsi pelanggan mencakup pengambilan data pelanggan yang terkait dengan bon berdasarkan kriteria tertentu. Berikut merupakan rancangan JSON data pelanggan:

JUTEI Edisi Volume.5 No.1 April 2021
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2021.51.230

```
{
    kd_customer: "03000001",
    alias: "",
    migrasi: null,
    ktp: "",
    nama: "SUPARTI/KEC KLT UT",
    alamat: "KANTOR KEC.KLATEN UTARA/JL KOPRAL SAYOM GG GIRISO NO 3",
    tempat_lahir: "KLATEN",
    tgl_lahir: "1955-08-09",
    jns_kelamin: "P",
    agama: "1",
    kd_lokasi: "",
    wilayah: "Mojayan,KLATEN TENGAH,KLATEN,JAWA TENGAH",
    pekerjaan: "8",
    status_nikah: "2",
    telp: "0272323532",
    ponsel: "",
    email: "",
    status: "",
    upline: "#",
    bts_bon: 1,
    nilai_bon: 0,
    bts_bon_istimewa: 0,
    nilai_komisi: 0,
    komisi_diambil: 0,
    jenis_customer: "2",
    jenis_customer_bon: "3",
    jenis_komisi: "0",
    status_ptt: "N",
    status_plgaktif: "Y",
    tgl_entry: null,
    user_entry: null,
    tgl_update: "2020-05-09 12:42:30",
    user_update: "GA",
    tgl_pemutihan: "0000-00-00 00:00:00",
    user_pemutihan: "",
    saldonetbon: 0,
    poin: 6
}
```


Gambar.10. JSON pelanggan

2. Fungsi Bon

Fungsi bon mencakup pengambilan data pencatatan bon yang telah dilakukan oleh kasir. Berikut merupakan bentuk JSON dari data bon:


```
{
    "no_bon": "B-03-0520-00746",
    "tgl_bon": "2020-05-17 20:41:28",
    "kode_plg": "03045806",
    "diskon": 0,
    "status": "B",
    "kasir": "04512",
    "jumlah_total": 1214000,
    "status_bayar": null,
    "bank": null,
    "no_kartu": null,
    "no_validasi_bank": null,
    "no_pengantar": ""
}
```


Gambar.11. JSON bon

3. Fungsi Pembayaran

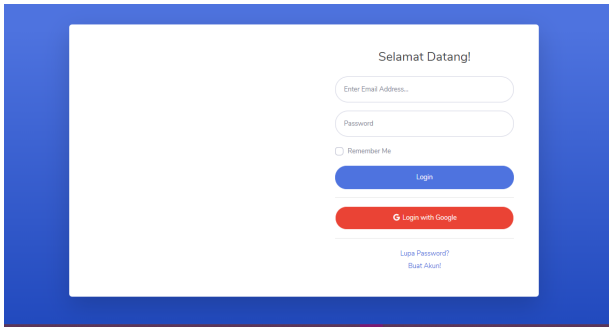
Fungsi pembayaran mencakup pencatatan angsuran langsung dan angsuran bulanan. Berikut merupakan JSON data pembayaran:


```
{
    "id": 201410,
    "no_bon": null,
    "jumlah_bon": 0,
    "bayar_bon": 150000,
    "tgl_byr": "2020-07-15",
    "kd_plg": "03669966",
    "kd_kasir": "11223344",
    "status_lgs": "1",
    "id_komisi": null
}
```


Gambar.12. JSON pembayaran piutang

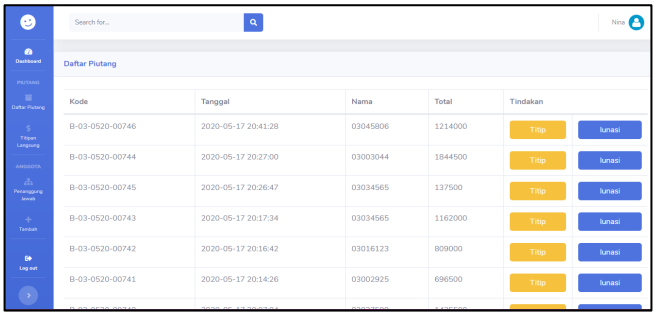
C. Rancangan Masukan

Masukan sistem didapatkan melalui aplikasi berbasis web yang dioperasikan oleh petugas kasir. Gambar 13 merupakan tampilan halaman *lo gin* dari aplikasi web.



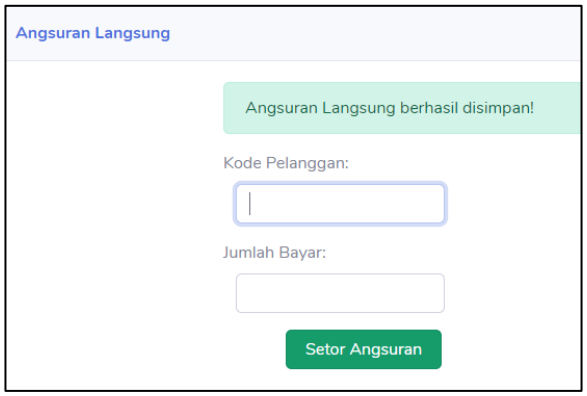
Gambar.13. Halaman login

Antarmuka daftar piutang menampilkan informasi piutang yang dimiliki pelanggan berupa table dan aksi yang dapat diambil (menitipkan angsuran atau melunasi). Antarmuka daftar piutang terlihat pada Gambar 14.



Gambar.14. Halaman daftar piutang

Antarmuka angsuran langsung digunakan untuk menginputkan angsuran yang dilakukan oleh pelanggan. Data yang dibutuhkan berupa kode pelanggan dan jumlah angsuran. Antarmuka tersebut terlihat pada Gambar 15.



Gambar.15. Halaman setor angsuran langsung

D. *Rancangan Hasil*

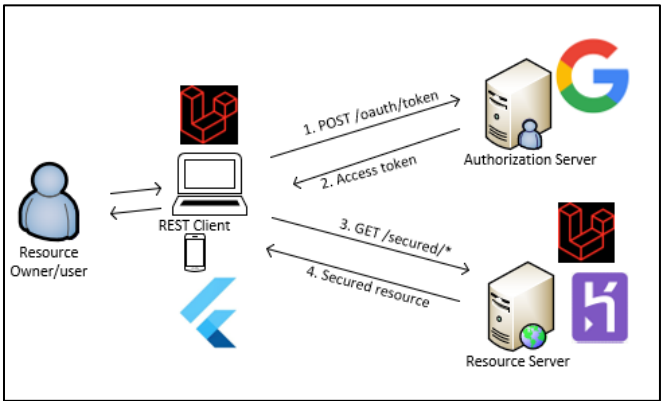
Hasil yang diharapkan dari perancangan REST API berupa format JSON yang digunakan untuk pertukaran data. Gambar 16 merupakan JSON piutang.


```
{
    "no_bon": "B-03-0909-00001",
    "tgl_bon": "2009-09-12 10:01:12",
    "kode_plg": "03001603",
    "diskon": 0,
    "status": "L",
    "kasir": "01388",
    "jumlah_total": 117500,
    "status_bayar": null,
    "bank": null,
    "no_kartu": null,
    "no_validasi_bank": null,
    "no_pengantar": null
}
```


Gambar.16. JSON piutang

IV. HASIL DAN ANALISIS

Pada bagian ini akan dibahas mengenai implementasi dan pengujian REST API. Implementasi yang dilakukan terlihat seperti Gambar 17.



Gambar.17. Implementasi system

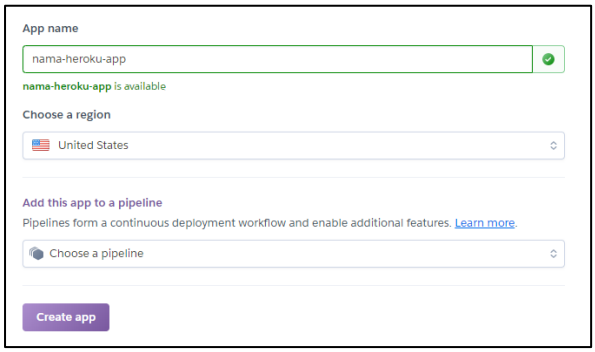
Penjelasan dari Gambar 17 adalah sebagai berikut, yang akan dijelaskan poin per poin tentang implementasi sistem yang digunakan:

1) *Resource Owner:* Merupakan pengguna aplikasi yang memiliki hak penuh terhadap akun yg digunakan untuk otorisasi.
2) *REST Client:* Aplikasi yang membutuhkan data pengguna untuk otorisasi agar dapat mengakses data pengguna.
3) *Authorization Server:* Server yang melakukan proses otentikasi dan otorisasi pengguna, serta memberikan kode akses berupa token.
4) *Resource Server:* REST API yang dibutuhkan dan diakses oleh aplikasi klien dan pengguna.

A. *Mempersiapkan Resource Server*

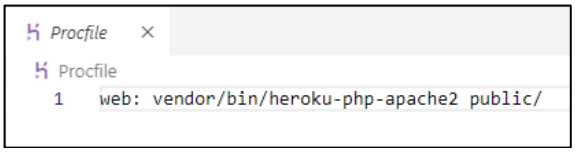
Resource server / RESTful API dibangun menggunakan Laravel dan didaringkan menggunakan platform Heroku. API memuat antarmuka untuk mengambil data yang dibutuhkan oleh pengguna. Berikut merupakan langkah-langkah yang dilakukan untuk menyiapkan resource server pada platform Heroku:

1. Langkah pertama yang harus dilakukan adalah mendaftar dan log in di Heroku dengan alamat situs heroku.com. Apabila sudah memiliki akun maka langsung saja log in.
2. Kemudian, perlu dibuat proyek baru dengan nama yang diinginkan. Dalam tahap ini dicontohkan nama-heroku-app sebagai nama proyek. Tekan Create app untuk membuat proyek.



Gambar.18. Tampilan membuat proyek heroku

3. Setelah proyek berhasil dibuat, perlu dilakukan instalasi Heroku CLI untuk melakukan operasi CLI dan konfigurasi di Heroku. Operasi yang dimaksud seperti push, pull, commit, clone, maupun penambahan variabel konfigurasi aplikasi yang akan dideploy. Untuk OS windows, Heroku CLI 64 bit dapat diunduh pada tautan berikut https://cli-assets.heroku.com/branches/stable/heroku-windows-amd64.exe
4. Langkah berikutnya adalah membuat fail Procfile pada proyek Laravel. Fail Procfile dibutuhkan agar proyek dapat diakses menggunakan peramban. File Procfile berisi kode seperti pada Gambar 19.



Gambar.19. Konfigurasi Procfile

5. Kemudian, dilakukan inisiasi git di dalam proyek Laravel.

JUTEI Edisi Volume.5 No.1 April 2021
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2021.51.230

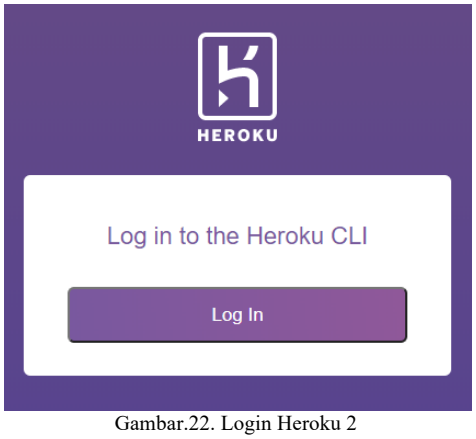
```
PS D:\SKRIPSI\amigo_api> git init
Reinitialized existing Git repository in D:/SKRIPSI/amigo_api/.git/
```

Gambar.20. Inisiasi git

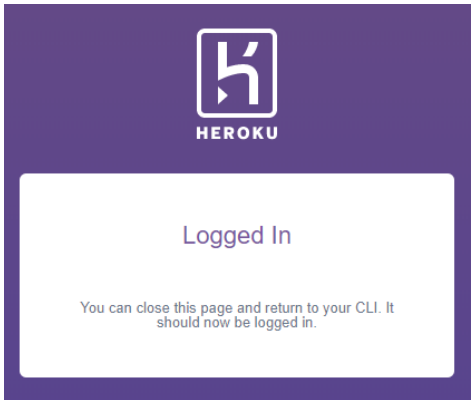
6. Masuk ke akun Heroku untuk dapat melakukan deployment proyek. Pada terminal ketikkan **heroku login [enter]**, kemudian tekan sembarang tombol maka akan muncul jendela peramban untuk melakukan log in seperti pada Gambar 21.

```
PS D:\SKRIPSI\amigo_api> heroku login
heroku: Press any key to open up the browser to login or q to exit:
```

Gambar.21. Login Heroku 1



Gambar.22. Login Heroku 2



Gambar.23. Login Heroku 3

Gambar 23 dan Gambar 24 merupakan tampilan saat login akun heroku telah berhasil.

```
PS D:\SKRIPSI\amigo_api> heroku login
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/08f69bd9-872d-4dc1-84ed-445aa0b73b0a
Logging in... done
Logged in as nina.wulandari@si.ukdw.ac.id
PS D:\SKRIPSI\amigo_api>
```

Gambar.24. Login berhasil

Proyek telah terhubung ke akun Heroku, namun belum terhubung ke proyek yang dibuat. Untuk menghubungkannya digunakan perintah seperti pada Gambar 25.

```
PS D:\SKRIPSI\amigo_api>  heroku git:remote -a nama-heroku-app
set git remote heroku to https://git.heroku.com/nama-heroku-app.git
PS D:\SKRIPSI\amigo_api>
```

Gambar.25. Menghubungkan proyek ke heroku

Setelah proyek Heroku sudah terhubung dengan proyek Laravel, maka proyek dapat dimasukkan ke dalam repository Heroku dengan perintah seperti pada Gambar 26.

```
PS D:\SKRIPSI\amigo_api> git add .
PS D:\SKRIPSI\amigo_api> git commit -m "initial commit"
On branch master
nothing to commit, working directory clean
PS D:\SKRIPSI\amigo_api> git push heroku master
```

Gambar.26. Push proyek ke heroku

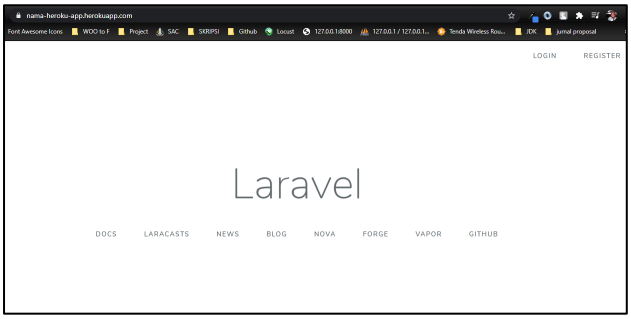
Langkah terakhir sebelum proyek Laravel dapat diakses adalah memasukkan APP KEY yang ada di fail .env Laravel ke dalam proyek Heroku. Lihat Gambar 27.

```
PS D:\SKRIPSI\amigo_api> heroku config:set APP_KEY=base64:Ch+JvEz2ITG5TwDkPuhzsKv0hEh1zg9mdRv0bT+cTkI=

Setting APP_KEY and restarting ● nama-heroku-app... done, v4
APP_KEY: base64:Ch+JvEz2ITG5TwDkPuhzsKv0hEh1zg9mdRv0bT+cTkI=
PS D:\SKRIPSI\amigo_api>
```

Gambar.27. Menambahkan APP KEY Laravel

Seperti pada Gambar 28, proyek Laravel sudah dapat diakses di domain heroku. Selanjutnya, proyek Heroku dapat digunakan sebagai root untuk Back End yang dibuat.



Gambar.28. JSON pembayaran piutang

B. Proses Otentikasi OAuth 2.0

Dalam otentikasi OAuth, proses request data diamankan dengan cara mengirimkan token di *authorization header.* Token diberikan oleh server otorisasi kepada pengguna menggunakan mekanisme tertentu. Tahapan mekanisme tersebut kurang lebih sebagai berikut:

1. Aplikasi didaftarkan pada server otorisasi untuk mendapatkan *client id* dan *client secret.*
2. *Client credentials* tersebut digunakan untuk meminta access token.
3. Server otorisasi memberikan access token dan refresh token. Access token memiliki masa berlaku tertentu, sehingga saat access token sudah kadaluwarsa, refresh token digunakan untuk meminta access token yang baru.
4. Access token digunakan untuk mengirim request kepada resource server / RESTful API.
5. Jika access token valid, data diberikan.
6. Jika access token kadaluwarsa / expired, dilakukan proses meminta access token menggunakan refresh token dan client credentials.
7. Token yang didapat digunakan untuk mengirim request.

Implementasi mekanisme tersebut dilakukan dengan menerapkan konfigurasi di aplikasi klien dan resource server, yang akan dibahas kemudian.

A.1. Mendapatkan Token

Token dibutuhkan *user* untuk dapat mengakses REST API. Google digunakan sebagai server otorisasi, sehingga user perlu melakukan *log in* ke sistem menggunakan Google.

Pada penelitian ini, digunakan paket resmi Laravel, yaitu Socialite, yaitu paket yang memungkinkan pengguna untuk log in menggunakan beberapa media sosial seperti Google. Penggunaan Socialite memerlukan pengaturan *provider* yang digunakan. Pengaturan tersebut berupa kredensial, yang didapat setelah registrasi aplikasi di Google Cloud Platform. Gambar 11 merupakan pengaturan provider Google.

```
'google' => [
    'client_id' => '568989792447-vfd706c6490j0tmvtg90tqko1hmvec9b.apps.googleusercontent.com',
    'client_secret' => 'cZuyuJM0jrci-Meb6DxdVNtx',
    'redirect' => 'http://webpiutang.herokuapp.com/auth/google/callback',
],
```

Gambar.11. Konfigurasi provider di config/services.php

Socialite diterapkan saat user *log in* ke sistem, yaitu digunakan untuk mengotentikasi menggunakan akun Google dan memberikan otoritas ke aplikasi melalui *consent prompt* saat user *log in* untuk pertama kali. Penerapan Socialite pada fungsi *log in* dapat dilihat pada Gambar 29 dan Gambar 30.

```
public function redirectToGoogle()
{
    return Socialite::driver('google')
    ->with(["access_type" => "offline",
            "prompt" => "consent select_account"])->redirect();
}
```

Gambar.29. Penggunaan Socialite untuk *log in 1*

```
$user = Socialite::driver('google')->user();
//dd($user);
$finduser = User::where('email', $user->email)->first();

if($finduser){
    $_SESSION['refresh_token'] = $user->refreshToken;
    Auth::login($finduser);
    return redirect('/home');

}
```

Gambar.30. Penggunaan Socialite untuk menyimpan token

A.2. Mengirim Request

Proses mengirim request dari aplikasi klien ke server membutuhkan token yang harus disertakan. Adapun library yang digunakan untuk proses request terlihat pada Gambar 31.

```
use GuzzleHttp\Exception\RequestException;
use Guzzle\RequestOptions;
use GuzzleHttp\Client;
use kamermans\OAuth2\GrantType\RefreshToken;
use kamermans\OAuth2\OAuth2Middleware;
use GuzzleHttp\HandlerStack;
```

Gambar.31. Library yang digunakan untuk kirim request

Guzzle digunakan untuk melakukan http request, dan untuk menangani token kadaluwarsa digunakan library guzzle-oauth2-subscriber yang dibuat oleh kamermans. Gambar 32 menunjukkan penerarapan penggunaan library tersebut.

```
session_start();
$reauth_client = new Client([
    // URL for access_token request
    'base_uri' => 'https://accounts.google.com/o/oauth2/token',
]);
$reauth_config = [
    "refresh_token" => $_SESSION['refresh_token'],
    "client_id" => $client_id,
    "client_secret" => $client_secret,
];
$grant_type = new RefreshToken($reauth_client, $reauth_config);
$oauth = new OAuth2Middleware($grant_type);

$stack = HandlerStack::create();
$stack->push($oauth);
$client = new Client([
    'handler' => $stack,
    'auth'    => 'oauth',
]);
```

Gambar.32. Penggunaan library guzzle-oauth2-subscriber

Pada kode pengiriman request tidak diperlukan lagi penulisan header, dikarenakan hal tersebut sudah ditangani oleh *library* yang digunakan.

```
$response = $client->post('https://hidden-ridge-82392.herokuapp.com/api/o/bons/'.$req->kode_plg.'',
    [
    'form_params' => [
    'bayar_bon'   => $req->bayar_bon,
    'kd_kasir' => '11223344',
    ]
    ]);

if(($response->getStatusCode())==200){
    return redirect('/titipan')->with('status','Angsuran Langsung berhasil disimpan!');
}
```

Gambar.33. Mengirim request

A.3. Verifikasi Token

Sebelum *request* diteruskan ke server, token yang disertakan dicek terlebih dahulu oleh *middleware*. Gambar 34 menunjukkan kode pengecekan token di file *middleware*.

```
$accessToken = $request->bearerToken();

$user = Socialite::driver('google')->userFromToken($accessToken);
if(!$user){
    return response()->json([
        "error" => $e,
        "message" => "Unauthorized user"],
        401);
}
return $next($request);
```

Gambar.34. Pengecekan token

C. Proses Otentikasi HTTP Basic Auth

C.1. Penerapan HTTP Basic Auth

Pada basic authentication digunakan kredensial berupa email dan password untuk bisa masuk ke sistem. Laravel menyediakan otentikasi basic bawaan berupa paket laravel/ui. Adapun penggunaannya adalah dengan menginstall paket tersebut menggunakan perintah composer require laravel/ui [enter] yang dijalankan di dalam folder proyek.

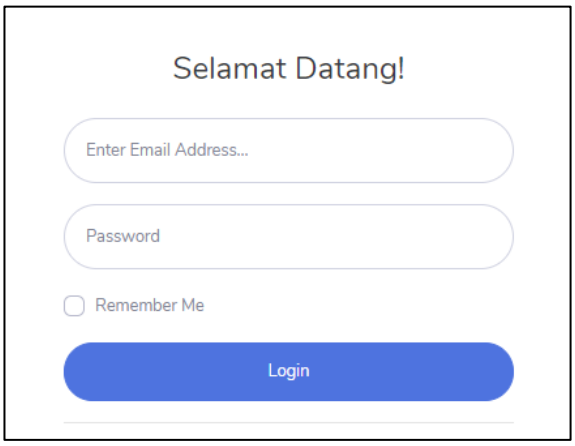
JUTEI Edisi Volume.5 No.1 April 2021
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2021.51.230

```
composer require laravel/ui

php artisan ui vue --auth
```

Gambar.34. Perintah untuk membuat halaman login

Kemudian perintah php artisan vue --auth digunakan untuk membuat tampilan halaman log in. Gambar 35 merupakan tampilan log in yang sudah dimodifikasi. Pengguna dapat masuk ke sistem dengan menginput email dan kata sandi yang sudah terdaftar.



Gambar.35. Halaman log in

Pada fungsi log in password dan email pengguna disimpan guna melakukan request. Lihat Gambar 36.

```
session_start();
$user = User::where('email', $req->email)->first();

if(Hash::check($req->password, $user->password)){
    Auth::login($user);
    $_SESSION['pass'] = $req->password;
    $_SESSION['email'] = $req->email;
    return redirect('/home');
}else{
    return redirect('/login');
}
```

Gambar.36. penerapan Basic Auth

C.2. Mengirim Request

Untuk mengirim permintaan diperlukan library seperti pada Gambar 37. Pada penelitian ini digunakan library Guzzle.

```
use GuzzleHttp\Client;
use GuzzleHttp\Exception\RequestException;
use Guzzle\RequestOptions;
use Illuminate\Http\Request;
use Auth;
```

Gambar.37. Library yang digunakan

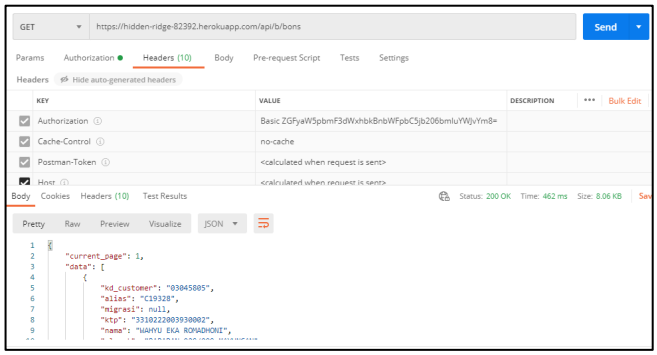
Pengiriman permintaan dilakukan seperti pada Gambar 38. Email dan kata sandi yang disimpan dimasukkan ke dalam baris kode request sebagai atribut otentikasi.

```
session_start();
$email = $_SESSION['email'];
$pass = $_SESSION['pass'];

$client = new Client();
$request= $client->get('https://hidden-ridge-82392.herokuapp.com/api/b/bons',
['auth' => [$email, $pass]]);

$response = $request->getBody()->getContents();
$bon=json_decode($response, true);
return view('piutang.tampil')->with('bon', $bon);
```

Gambar.38. Mengirim request



Gambar.39. Contoh mengirim request menggunakan Postman

Gambar 39 di atas merupakan contoh pengiriman request menggunakan Postman. Terlihat bahwa pada tab header terdapat key Authorization yang bernilai kredensial Basic, yaitu tipe otentikasi yang digunakan.

C.3. Verifikasi Request

Sama seperti OAuth 2.0, pengecekan pengguna diperlukan konfigurasi middleware. Adapun middleware untuk pengecekan Basic Auth ditunjukkan pada Gambar 40.

```
public function handle($request, Closure $next)
{
    return Auth::onceBasic() ?: $next($request);
}
```

Gambar.40. Middleware Basic Auth

Middleware yang sudah dibuat perlu didaftarkan pada file app\Http\Kernel.php dan kemudian bisa diterapkan pada end point yang dibuat, seperti pada Gambar 41 dan Gambar 42.

```
'authbasic' => \App\Http\Middleware\AuthBasic::class,
```

Gambar.41. Mendaftarkan file middleware

```
Route::group(['prefix' => '/b', 'middleware' => 'authbasic'], function(){
```

Gambar.42. Penerapan *middleware* di *end point*

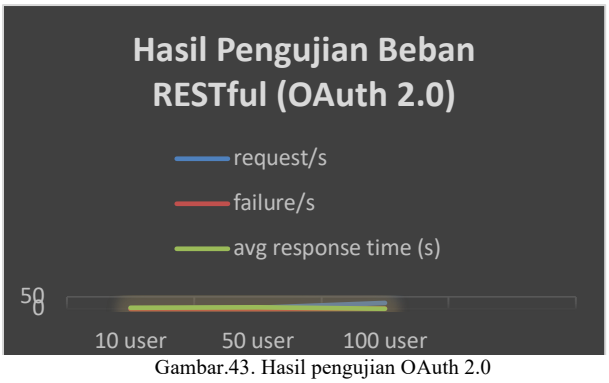
D. Pengujian Beban

Pengujian dilakukan menggunakan aplikasi Locust. Dengan aplikasi ini dapat dilihat bagaimana kinerja server dalam menangani *request* yang masuk secara bersamaan dalam waktu tertentu.Pengujian dilakukan dengan simulasi 10, 50, dan 100 pengguna yang mengakses REST API yang masing-masing telah diamankan dengan OAuth 2.0 dan Basic Auth. Hasil pengujian disampaikan dalam bentuk tabel pada Tabel 2 dan grafik yang disajikan pada Gambar 43 dan Gambar 44.

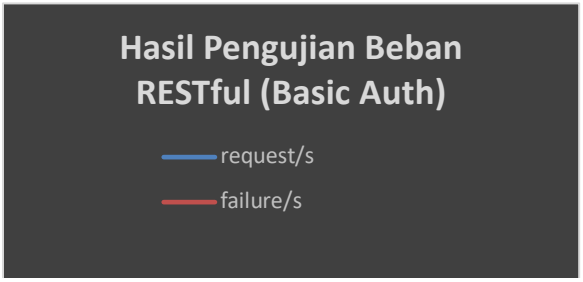
TABEL II
HASIL PENGUJIAN BEBAN

Kriteria	OAuth 2.0			Basic Auth		
User	10	50	100	10	50	100
Durasi	15 m	1 j	12 m	15 m	49 m	18 m
Requests	1131	14210	18585	2769	18211	18576
Failures	0	511	1431	849	97	1350
Median RT (ms)	4900	730	640	310	700	1400
Average RT (ms)	5242	6866	1036	393	5062	3257
Min RT (ms)	940	2	379	124	0	415
Max RT (ms)	18479	574914	32769	1631	714269	123751
Avg content size	88941	86359	111794	54162	79129	111524
Request/s	1,21	3,67	24,53	2,93	3,05	15,79
Failure/s	0	0,13	1,89	0,9	0,02	1,15

Dari table terlihat bahwa, Basic Auth menangani request lebih banyak dibanding OAuth 2.0, dan berbanding lurus dengan rata-rata kegagalan *request* per detik yang lebih tinggi.



Gambar.43. Hasil pengujian OAuth 2.0



Gambar.44. Hasil pengujian Basic Auth

Seperti terlihat di Gambar 43 dan Gmbar 44, hasil pengujian kedua metode tersebut memiliki pola yang mirip. Kemiripan terdapat pada penanganan request dengan simulasi 50 user, yaitu rata-rata response time memiliki nilai tertinggi. Pada simulasi 100 user, response time pada OAuth 2.0 mengalami penurunan pada nilai tercepat, sedangkan pada Basic Auth mengalami penurunan namun tidak lebih cepat dibanding simulasi 10 user.

V. KESIMPULAN

Adapun kesimpulan dan saran yang didapat dari penelitian ini adalah sebagai berikut:

A. Kesimpulan

Berdasarkan penerapan metode dan pengujian yang dilakukan terhadap RESTful API, dapat diambil kesimpulan sebagai berikut:

1. Penggunaan RESTful memudahkan penyampaian informasi terkait piutang. Hal tersebut dikarenakan informasi bon anggota yang diperoleh penanggungjawab dipastikan sesuai dengan yang ada di toko.
2. Pengembangan sistem selanjutnya dapat menggunakan web service yang telah dibuat dengan link / root yang sudah disediakan.
3. Penggunaan media sosial untuk masuk ke sistem memudahkan pelanggan, sehingga tidak membebani pelanggan untuk mengingat username dan password aplikasi.
4. Penggunaan OAuth 2.0 sebagai protokol keamanan berpengaruh positif terhadap kinerja server dalam penanganan request. Terlihat dari pengujian, bahwa meski diberi request yang banyak (100 user), response time server terhitung cepat dengan rata-rata 1,03 detik.

B. Saran

1. Pengujian beban terkendala pada jaringan yang tidak stabil, sehingga untuk mendapatkan hasil pengujian yang lebih baik diperlukan kualitas jaringan yang baik dan stabil.
2. Penelitian ini belum sempurna. Untuk mendapatkan hasil pengujian yang lebih objektif, diperlukan parameter-parameter tertentu yang digunakan sebagai acuan penilaian. Sebagai contoh, parameter jumlah request per detik yang harus dipenuhi, toleransi failed request per detik, dan sebagainya. Dengan demikian metode yang diuji dapat dikategorikan baik atau buruk sesuai parameter tersebut.
3. Penelitian ini dapat dikembangkan lagi dengan lebih berfokus kepada pendekatan microservice dari RESTful API.

DAFTAR PUSTAKA

[1] S. M. Hery, Pengantar Akuntansi, Jakarta, Indonesia, 2015.
[2] T.J. Utomo. (Juni, 2010). Lingkungan Bisnis dan Persaingan Bisnis Ritel. *Fokus Ekonomi*. Hal. 70-80.
[3] V. Surwase. (Januari, 2016). *REST API Modeling Languages - A Developer's Perspective*. *International Journal of Science Technology & Engineering*. Hal 634-637.
[4] I. Kusuma, A. Susanto, I. U. Mulyono. (April, 2019). IMPLEMENTASI RESTFUL WEB SERVICES DENGAN OTORISASI OAUTH 2.0 PADA SISTEM PEMBAYARAN PARKIR. *Simetris*. Hal. 391-404.
[5] S. S. Sodikin, Akuntansi Pengantar, Yogyakarta, Indonesia, 2009.
[6] R. Rischpater, *JavaScript JSON Cookbook,* Birmingham, 2015.
[7] M. Waschke, *Personal Cybersecurity : How to Avoid and Recover from Cybercrime,* Washington, 2017.
[8] R. Boyd, *Geting Started with OAuth 2.0*, Sebastopol, 2012.
[9] C. Bihis, *Mastering OAuth 2.0,* Birmingham, 2015.
[10] B. Enrile, *Performance Testing with JMeter 2.9*, Birmingham-Mumbai, 2013.
[11] D. I. Permatasari, M. Ardani, A. Y. Ma'ulfa, N. Ilhami, S. G. Pratama, S.R. Astuti, & N.W. Naufalita. (Januari, 2020). Pengujian Aplikasi Menggunakan Metode Load Testing dengan Apache Jmeter pada Sistem Informasi Pertanian. *Jurnal Sistem dan Teknologi Informasi*. Hal. 135-139.

JUTEI Edisi Volume.5 No.1 April 2021
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2021.51.230