Revamp Keamanan Web Service Milik PT XYZ Menggunakan REST API

Agus Tedyyana[1], Muhammad Fauzi[2], Fajar Ratnawati[3]
[1,3]Program Studi Keamanan Sistem Informasi Politeknik Negeri Bengkalis
[2]Cloudcode Indonesia
[1,3]Jl. Bathin Alam, Sungai Alam Bengkalis Riau - 28711
[2]Jl. Antara, Kecamatan Bengkalis - Bengkalis Riau - 28711
e-mail: [1]agustedyyana@polbeng.ac.id,[2]fauzi@cloudcode.id, [3]fajar@polbeng.ac.id

Abstrak

Keamanan data merupakan hal paling utama yang harus dijaga oleh perusahaan guna mendapatkan kepercayaan lebih dari pelanggan. Web service merupakan salah satu teknologi yang rentan terhadap kebocoran data jika tidak dirancang dengan benar. PT. XYZ pada kasus ini telah memiliki web service yang sedang berjalan di sisi production. Namun dari segi keamanan masih ada celah yang harus diperbaiki. Dari hasil studi literatur yang dilakukan, peneliti melakukan rancang ulang dari web service dengan menggunakan REST API yang dibangun pada framework Lumen dengan pertimbangan dari kecepatan load data dan kemudahan deploy di shared hosting. Penelitian dilakukan dengan melakukan melakukan analisis terhadap sistem yang sedang berjalan kemudian dilakukan rancang ulang, implementasi ke sistem, dan pengujian API menggunakan Postman. Dari hasil penelitian didapatkan hasil bahwa Lumen mampu digunakan untuk membangun REST API sesuai dengan rancangan guideline dan mampu mengamankan data-data pelanggan dari akses pihak yang tidak berkepentingan dengan menerapkan token saat pengguna melakukan request ke REST API. Dengan rancang ulang REST API ini, membuat keamanan data pelanggan PT. XYZ lebih menjadi aman lagi, karena tidak bisa diakses secara sembarangan.

Kata kunci: *REST API, Web Service, Keamanan Data.*

Abstract

Data security is the most important thing that must be maintained by companies to get more trust from customers. Web service is a technology that is prone to data leakage if it is not designed properly. PT. XYZ in this case already has a web service running on the production side. However, from a security point of view, there are still gaps that need to be fixed. From the results of the literature study conducted, researchers redesigned the web service using the REST API which was built on the Lumen framework with consideration of data load speed and ease of deployment on shared hosting. The research was conducted by analyzing the running system and then redesigning it, implementing it to the system, and testing the API using Postman. The results showed that Lumen can be used to build a REST API following the draft guidelines and can secure customer data from unauthorized access by applying a token when the user requests the REST API. With this REST API redesign, making customers data of PT. XYZ is even more secure because it cannot be accessed carelessly.

Keywords: *REST API, Web Service, Data Security.*

1. Pendahuluan

Pada era digital saat ini, keamanan data merupakan sesesuatu hal yang sangat penting untuk diperhatikan. Apalagi data-data yang berkaitan dengan data milik pelanggan dari suatu perusahaan. Perusahaan yang baik tentunya harus memperhatikan keamanan data pelanggannya agar terjadi saling percaya antara pelanggan dengan perusahaan.

PT. XYZ pada studi kasus kali ini merupakan perusahaan milik daerah yang bergerak di bidang penyediaan air bersih untuk masyarakat. Guna mempermudah proses pencatatan meter

oleh petugasnya yang sebelumnya menggunakan catatan manual. PT. XYZ kemudian melakukan inovasi dengan mengembangkan aplikasi *mobile* yang terhubung dengan sistem utama mereka agar catatan milik petugas bisa terekap secara langsung ke sistem.

Proses integrasi data dari sistem utama milik PT. XYZ dengan aplikasi pencatatan meter sebenarnya telah menggunakan sebuah *web service*. *Web service* adalah sistem yang dimaksudkan untuk mendukung interoperabilitas antara mesin atau platform melalui jaringan [1]. Ketika aplikasi ingin mengakses *web service*, aplikasi membutuhkan protokol arsitektur berorientasi layanan melalui HTTP dan SOAP (Simple Object Access Protocol)[2]. Namun setelah di audit terdapat beberapa masalah baik dari sisi keamanan maupun penerapannya yang belum sesuai dengan metode RESTful API itu sendiri pada *web service* yang telah ada.

REST - (Representational state transfer) sesuai namanya, ini berkaitan dengan hubungan klien dan server dan bagaimana status disimpan. Arsitektur REST didasarkan pada gaya arsitektur klien / server [3]. Sebuah *request* dan *response* dibangun berdasarkan proses transfer sumber daya. Semua sumber daya diidentifikasi oleh *Uniform Resource Identifier* (URI) unik, yang biasanya mewakili dokumen yang merekam status sumber daya. Secara umum, arsitektur gaya REST jauh lebih ringan dibandingkan dengan SOAP. Aplikasi web yang mengikuti arsitektur REST disebut juga sebagai RESTful *web service*. RESTful *web service* menggunakan metode HTTP GET, PUT, POST, dan DELETE untuk mengambil, membuat, memperbarui, dan menghapus sumber daya [4].

Idealnya saat membuat aplikasi server dan klien, klien perlu mengakses *resources*. Dari sisi keamanan *web service* yang telah dibuat, proses pemanggilan data dari API seperti data pelanggan belum menerapkan keamanan dengan tepat. *Web service* bisa memungkinkan dieksploitasi oleh penyerang jika mereka mengetahui endpoint dan parameter yang digunakan pada API. Setidaknya *web service* hendaknya memvalidasi permintaan dari pengguna dengan token misalnya guna melindungi data-data. Klien membutuhkan izin untuk mengaksesnya. Jika diizinkan, alat pertukaran informasi izin akan diberikan. Alat ini biasa disebut dengan akses token. Otentikasi token ini digunakan saat pengguna log in ke aplikasi klien. Teknologi yang biasanya digunakan dalam proses ini adalah dengan memanfaatkan JSON Web Token (JWT) [5].

Penelitian tentang penggunaan RESTful sebagai *web service* dengan baik dan benar telah banyak dilakukan oleh berbagai peneliti. Dalam penelitian [6] yang mengembangkan sebuah aplikasi reservasi tiket berbasis Android memamaparkan bahwa aplikasi dikembangkan menggunakan RESTful API pada *framework* Laravel, sebagai salah satu implementasi *web service*. Keamanan pertukaran data antara perangkat android dan layanan web menggunakan Laravel Passport, sebagai API generator token. Lalu penelitian [7] yang mengembangkan sebuah sistem Sandbox yang menyerupai dengan aplikasi dari pihak ketiga untuk mempermudah proses pengujian aplikasi dari PT. Semangat Gotong Royong yang melibatkan pihak ketiga. Sistem Sandbox akan dikembangkan menjadi REST API dan ditulis menggunakan bahasa pemrograman Golang. Dalam melakukan komunikasi dengan sistem lain, New Simple Queue (NSQ) juga digunakan yang dapat mendukung konkurensi dan mencegah kegagalan transmisi data. Akibatnya, sistem Sandbox dapat menerima permintaan dan akan memproses tanggapan yang mirip dengan fungsi dari pihak ketiga. Ada juga penelitian [8] yang memanfaatkan RESTful API sebagai media komunikasi antara aplikasi klien yang dibangun berdasarkan ReactJS dengan aplikasi server yang dibangun dengan NodeJS. Kedua aplikasi tersebut menggunakan bahasa Javascript dalam pembuatannya. Hal ini menunjukkan setidaknya sudah ada 3 buah bahasa pemrograman yang bisa digunakan sebagai media pembuatan RESTful API dari kutipan penelitian yang ditemukan.

Selanjutnya dikutip dari penelitian [9] yang berfokus pada keamanan dalam hal pertukaran data pada aplikasi dengan menggunakan JWT. Penggunaan autentifikasi JWT pada RESTful API yang dibangun membuat aplikasi menjadi lebih aman karena aplikasi tidak dapat diakses jika tidak menggunakan token. Pembahasan tentang JWT pada *web service* lebih lanjut ada pada penelitian [10] dimana menurut peneliti, JWT merupakan mekanisme otentikasi pada *web service*, namun penerapan JWT standar dengan algoritma *hashing* SHA-256 masih belum

optimal. Oleh karena itu, penelitian dilakukan untuk membahas tentang optimasi keamanan JWT dengan algoritma *hashing* SHA-512. Hasil dari penelitian tersebut adalah penggunaan SHA-512 menghasilkan waktu yang lebih baik 1% dibandingkan SHA-256, namun pada ukuran token SHA-512 lebih besar 2% dari SHA-256.

Dari penelitian-penelitian tersebut dan juga permasalahan yang terjadi pada penelitian ini dapat digambarkan proses rancang ulang keamanan *web service* yang akan dibangun nantinya akan menggunakan teknologi REST API dengan menggunakan *framework* Lumen yang merupakan *fast micro-framework* bagian dari Laravel namun dengan kecepatan load data yang lebih tinggi mengingat peruntukannya yang memang dikhususkan untuk mengembangkan *web service* [11]. Selain kecepatan, faktor yang dipertimbangkan dalam pemilihan Lumen sebagai framework adalah kemudahan *deploy* yang mendukung *shared hosting* di berbagai layanan *hosting*, tidak seperti NodeJS dan Golang yang tidak semua layanan *hosting* apalagi *shared hosting* memiliki layanan untuk *deploy* ke sisi production. Untuk keamanan yang juga ditekankan dari pemasalahan pada penelitian ini akan menggunakan token sebagai standar dari keamanan RESTful API.

3. Metode Penelitian

Metode Penelitian yang dilakukan pada penelitian kali ini terdiri dari beberapa tahap yang dimulai terlebih dahulu dengan tahap analisis sistem yang ada, analisis kebutuhan sistem yang diusulkan, implementasi sistem dan pengujian sistem.

3.1. Analisis Sistem yang Berjalan

Sebelum menyelesaikan permasalahan, terlebih dahulu dilakukan tinjauan dan analisa dari sistem yang sudah ada, dalam hal ini adalah *web service* yang dibikin dengan konsep REST API. Namun jika dilihat dari struktur URI *web service* yang sudah ada saja bisa kelihatan jika *web service* belum sepenuhnya menerapkan rancangan dari *guideline* URI pada REST API dengan benar [12]. Perhatikan gambar 1 yang merupakan proses uji coba REST API menggunakan aplikasi Postman.

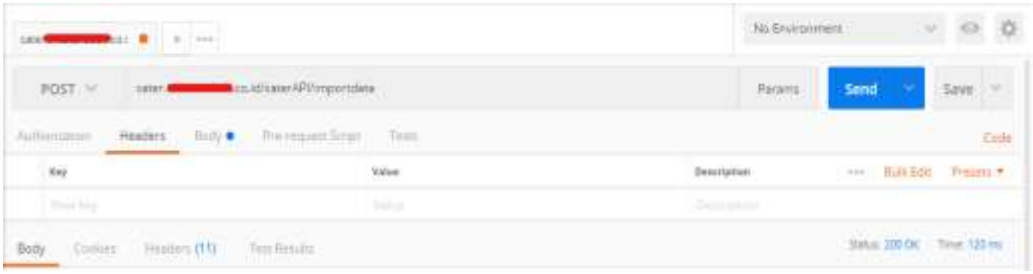
Gambar 1. REST API untuk login sebelum perbaikan

Jika dilihat pada gambar diatas, URI yang diberikan tidak menggunakan *names atau verbs* untuk menjelaskan tentang *resouces* yang dipanggil oleh REST API. Berdasarkan fungsinya URI REST API tersebut digunakan sebagai proses autentikasi atau *login*, namun jika dilihat berdasarkan URI kita tidak akan pernah mengerti jika URI tersebut digunakan untuk proses autentikasi, idealnya URI tersebut menggunakan verbs login / masuk misalnya sebagai penanda.

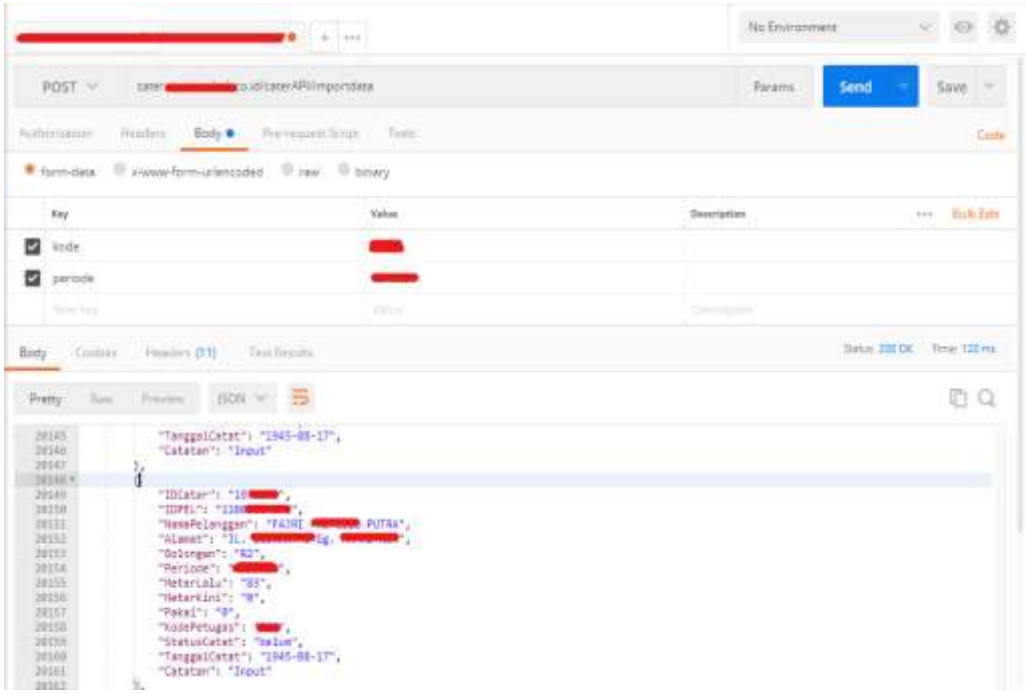
Dari segi URI *case*, terlihat jika URI menggunakan *Camel Case*. Walaupun dari segi kemudahan pembacaannya termasuk baik, kelemahan utamanya adalah efektif dalam konteks *case sensitive*. Sedangkan menurut RFC3986, URL bersifat *case sensitive* (kecuali untuk *schema* dan *host*). Namun, dalam praktiknya, *case sensitive* dapat menyebabkan disfungsi

dengan API yang dihosting di sistem Windows. Jadi lebih direkomendasikan untuk menggunakan *spinal case*. Dalam kasus ini URI pada gambar menjadi cater-api misalnya.

Lalu fokus terakhir dari penelitian ini adalah memperhatikan keamanan pada REST API yang menyangkut data-data penting yang seharusnya tidak terekspose oleh publik. Peneliti mencoba mengakses REST API untuk mengimport data pelanggan dari sistem dan berikut hasilnya dapat dilihat pada gambar 2 dan 3 berikut ini.



Gambar 2. REST API untuk import data bagian *Headers* sebelum perbaikan

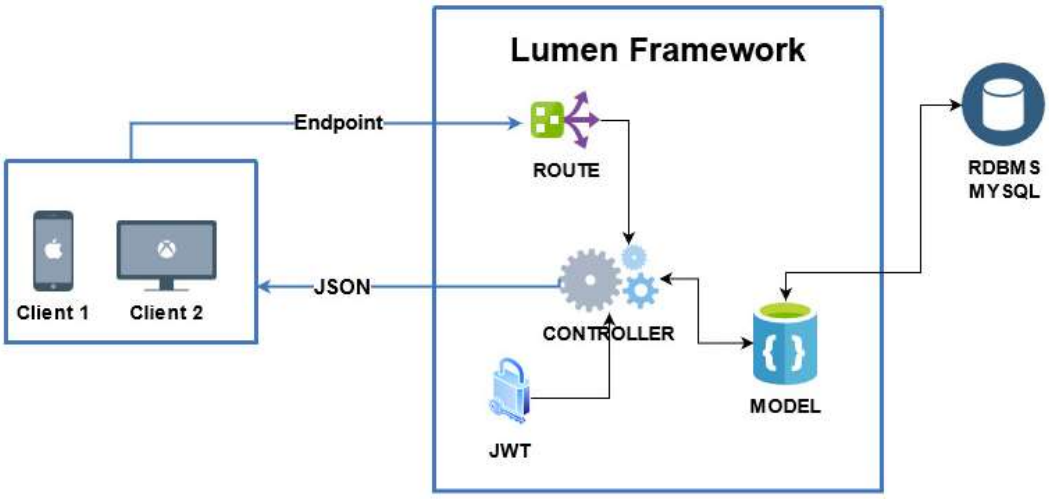


Gambar 3. REST API untuk import data bagian *form-data* sebelum perbaikan

Dari gambar 2 dilihat tidak ada *header* keamanan sama sekali, misalnya jika menggunakan JWT memasukkan 'Bearer' pada kolom *key* dan *token* pada kolom *value*. Dan pada gambar 3 parameter yang dikirim berupa kode dan periode menggunakan metode POST. Namun hasilnya REST API tidak menolak *request* dan tetap mengirim data pelanggan dalam bentuk json.

3.2. Analisis Kebutuhan Sistem yang Diajukan

Dari tahap analisa kebutuhan sebelumnya kita dapatkan fokus penelitian yaitu mengubah REST API sebelumnya mengikuti rancangan *guideline* REST API dan menerapkan keamanan JWT agar hanya pengguna yang telah terautentikasi saja yang bisa mengakses data-data dari REST API. Rancangan kebutuhan sistem yang diajukan dapat dilihat pada gambar 4 berikut ini.

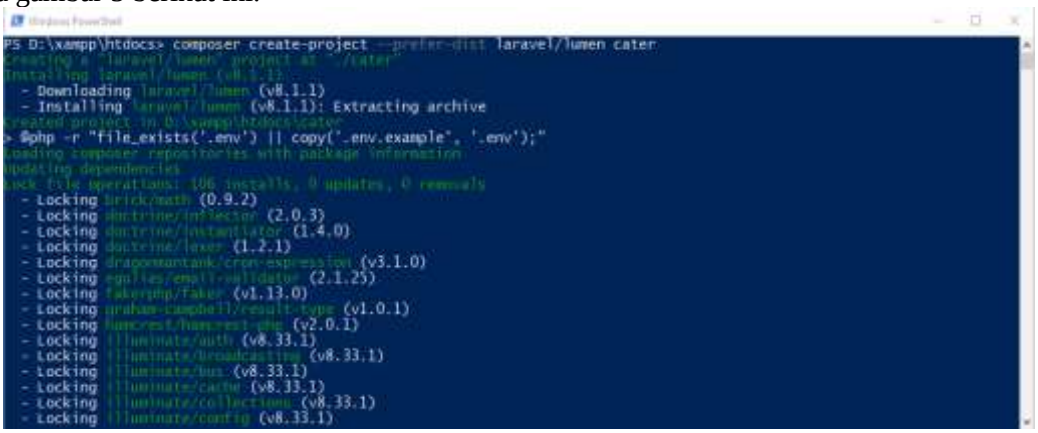


Gambar 4. Rancangan REST API Menggunakan Lumen dan JWT

Gambar 4 merupakan arsitektur dalam mengimplementasikan REST API pada framework Lumen. Alur pertama yang terjadi adalah *client* melakukan *request* dengan mengakses ke *endpoint* dari API. *Route* tersebut kemudian akan dihubungkan ke *controller* sesuai dengan *endpoint* yang diakses. *Controller* kemudian melakukan pemeriksaan apakah proses memerlukan otorisasi, jika memerlukan otorisasi maka akan melakukan pemeriksaan token terlebih dahulu dengan menggunakan JWT, jika tidak maka akan langsung mengakses *model* untuk mengambil data yang diperlukan di *database*, pada kasus ini menggunakan RDBMS MySQL. Data yang diambil tadi kemudian akan diteruskan oleh *controller* sebagai *response* berupa data JSON kepada *client*.

3.3. Implementasi Sistem

Pada tahap ini dilakukan implementasi sesuai dengan rancangan yang telah dibuat. REST API yang akan dibuat akan dibangun menggunakan bahasa pemrograman PHP dengan framework Lumen pada sisi *server*, serta MySQL sebagai *database* untuk menampung *resource*. REST API akan dibangun secara terpisah antara *client* dan *server*. Selain itu, juga diimplementasikan package JWT ke dalam sistem yang dibangun agar bisa mengimplementasikan keamanan bagi REST API. Langkah awal yang dilakukan dalam implementasi adalah melakukan instalasi Lumen di *web server*. *Web server* yang digunakan adalah Apache [13] dengan RDBMS MySQL [14]. Instalasi Lumen dilakukan menggunakan Composer. Dan proses implementasi dilakukan di lingkungan *local* sehingga tidak langsung menganggu proses yang ada di lingkungan *production*. Adapun proses instalasi dapat dilihat pada gambar 5 berikut ini.



Gambar 5. Instalasi Lumen

3.4. Pengujian Sistem

Pada tahap terakhir dari proses penelitian, dilakukan pengujian sistem untuk mencari tahu apakah sistem berjalan dengan baik dan mampu memberikan *response* saat *request* dijalankan. Proses pengujian menggunakan alat bantu Postman [15]. Postman biasa digunakan dalam pengujian berbagai *web service* termasuk juga REST API. Pengujian yang dilakukan berupa mengakses *resources* tanpa dan dengan token login untuk mengetahui keamanan REST API yang telah dibangun.

4. Hasil dan Pembahasan

Di bagian ini peneliti membangun tiap-tiap komponen dari framework Lumen, mulai dari *Route, Controller, Model*, bahkan juga memodifikasi *Middleware* [16] agar sesuai dengan kebutuhan. Pada gambar 6 berikut merupakan *route* sederhana yang peneliti gunakan dalam percobaan.

```
routes > web.php > ...
1   <?php
2   /** @var \Laravel\Lumen\Routing\Router $router */
3
4   /*
5   |--------------------------------------------------------------------------
6   | Application Routes
7   |--------------------------------------------------------------------------
8   |
9   | Here is where you can register all of the routes for an application.
10  | It is a breeze. Simply tell Lumen the URIs it should respond to
11  | and give it the Closure to call when that URI is requested.
12  |
13  */
14
15  $router->get('/', function () use ($router) {
16      return 'REST API CATER';
17  });
18
19  $router->post("/login", "AuthController@login");
20  $router->get("/customer", "CustomerController@index");
```

Gambar 6. Route REST API Menggunakan Lumen

Pada *route* yang ada peneliti membuat tiga buah *route*, dimana *route* pertama yaitu merupakan main URL yang menampilkan data return seperti pada gambar diatas. Lalu *route* kedua peneliti membuat *route* untuk login yang eksekusinya memanggil AuthController dengan method *login*. Dan terakhir adalah *route* untuk menampilkan seluruh data customer. URI pada URL telah menerapkan *guideline* dari rancangan REST API yang mana menyarankan untuk menggunakan kata penamaan (*names*) atau kata kerja (verbs).

Selanjutnya di gambar 7 berikut ini potongan *source code* dari Customercontroller yang bertugas untuk mengirim seluruh data *client* melalui REST API. Kelas menerapkan *constructor* yang bertugas memeriksa pengguna apakah sudah terautentikasi atau belum saat mengirim *request* ke sistem. Proses pemeriksaan ini dengan memanfaatkan *custom middleware* yang peneliti buat. Adapun custom middleware ditunjukkan pada gambar 8.

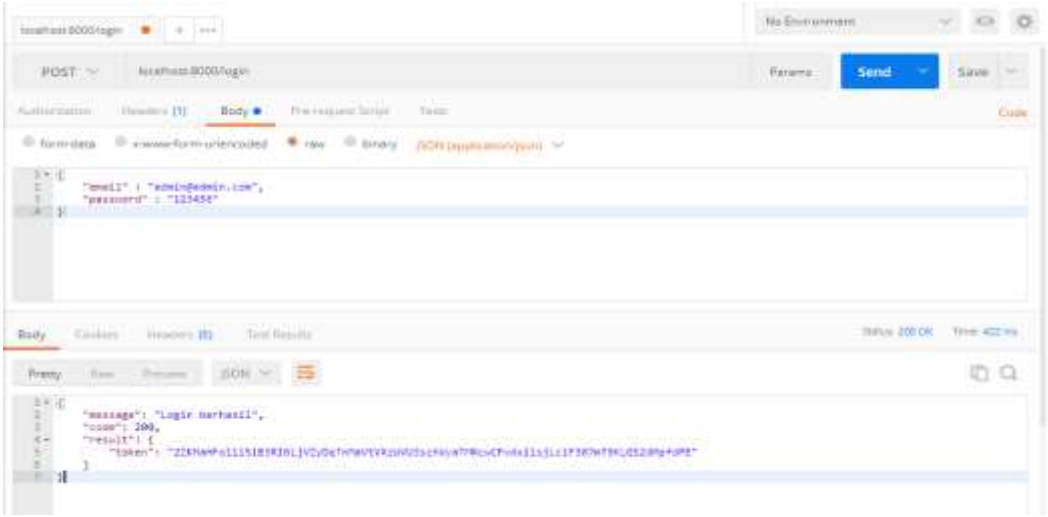
```
4
5   use Illuminate\Support\Facades\Hash;
6   use Illuminate\Http\Request;
7
8
9   class CustomerController extends Controller
10  {
11
12      public function __construct()
13      {
14          $this->middleware("login");
15      }
16
17      public function index()
18      {
19          $res = [];
20          $res['status'] = 'success';
21          $res['data'] = app('db')->select("SELECT * FROM pelanggan");
22          return response($res, 200);
23      }
24  }
```

Gambar 7. *Controller* REST API Menggunakan Lumen

```
14      * @param  \Closure  $next
15      * @return mixed
16      */
17     public function handle($request, Closure $next)
18     {
19         if ($request->input('token')) {
20             $check =  User::where('token', $request->input('token'))->first();
21             $res = [];
22             if (!$check) {
23                 $res['status'] = 'error';
24                 $res['message'] = 'Token Tidak Valid.';
25                 return response($res, 401);
26             } else {
27                 return $next($request);
28             }
29         } else {
30             $res['status'] = 'error';
31                 $res['message'] = 'Masukkan token terlebih dahulu.';
32                 return response($res, 401);
33         }
34     }
```

Gambar 8. *Middleware* REST API Menggunakan Lumen

Dari *middleware* tersebut dapat terlihat bahwa untuk mendapatkan *response* dari *controller* yang menerapkan *login middleware*, pengguna perlu juga untuk mengirimkan *request* berupa token. Jika token tidak ditemukan, maka REST API akan mengembalikan *response* berupa pesan *error* dengan kode HTTP 401. Begitu juga jika token yang dikirim tidak ditemukan di tabel user. Token yang dikirim diperoleh saat proses *login* dan token selalu di *generate* ulang saat pengguna melakukan proses *login* seperti pada gambar 9 berikut ini.



Gambar 9. Pengujian REST API Login Menggunakan Postman

Berikut pada tabel 1 dipaparkan hasil pengujian secara lengkap untuk proses login.

Tabel 1. Pengujian REST API Login

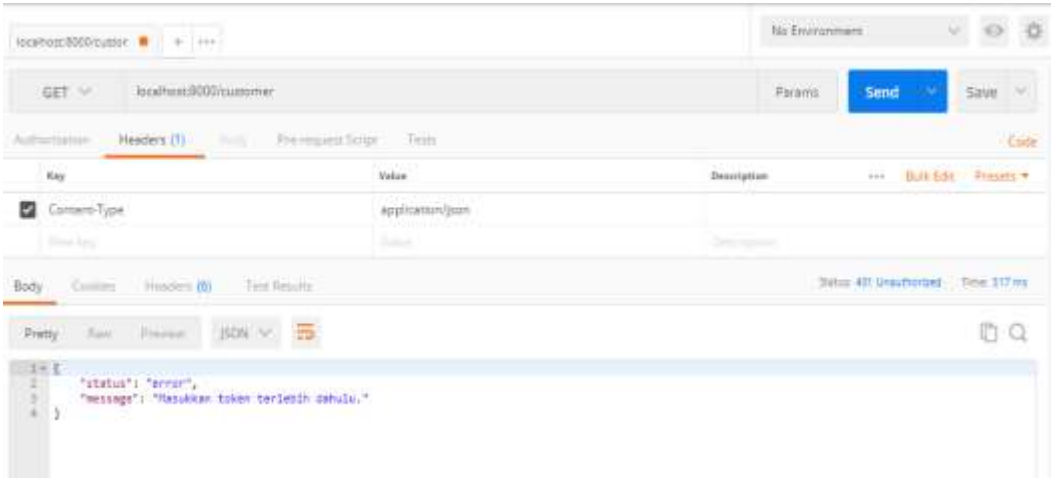
No.	Test Case	Hasil yang diharapkan	Hasil
1	Tidak mengirimkan parameter *email*/*password*	Mengirimkan *response* error	Berhasil
2	Mengirimkan parameter *email*/*password* tidak sesuai dengan akun	Mengirimkan *response* error	Berhasil
3	Mengirimkan parameter *email*/*password* sesuai dengan akun	Mengirimkan *response* berhasil bersama dengan token yang selalu di *generate* ulang	Berhasil

Dari hasil pengujian *login* tersebut didapatkan hasil bahwa setiap *test case* yang disimulasikan berhasil sesuai dengan hasil yang diharapkan. Dari token inilah nantinya akan digunakan sebagai proses autentikasi yang telah dibuat pada *custom middleware*, jika token yang dikirim ditemukan dan sesuai dengan yang ada tabel user maka pengguna bisa mendapatkan *response* dari *request* yang telah dilakukan. Untuk membuktikan keamanan yang dibuat pada *middleware* dilakukan pengujian lagi pada Postman dengan beberapa *test case* seperti pada tabel 2 berikut ini.

Tabel 2. Pengujian REST API Customer

No.	Test Case	Hasil yang diharapkan	Hasil
1	Tidak mengirimkan parameter token	Mengirimkan *response error*	Berhasil
2	Mengirimkan parameter token tidak sesuai dengan yang diberikan saat *login*	Mengirimkan *response error*	Berhasil
3	Mengirimkan parameter token sesuai dengan yang diberikan saat *login*	Mengirimkan *response* berhasil bersama dengan data pelanggan	Berhasil

Dari hasil pengujian pemanggilan REST API *customer* tersebut didapatkan hasil bahwa setiap *test case* yang disimulasikan juga berhasil sesuai dengan hasil yang diharapkan. Seperti yang dirancangkan, token berhasil mengamankan data yang ada dari akses yang tidak terautentikasi. Tidak sembarang token bisa digunakan untuk mengakses data-data yang di sisipkan dengan *custom middleware* yang telah dibuat. Adapun gambaran pengujian menggunakan Postman dapat dilihat pada gambar 10 berikut ini.



Gambar 10. Pengujian Keamanan REST API Menggunakan Postman

Dari gambar tersebut merupakan simulasi jika pengguna melakukan *request* ke REST API tanpa mengirimkan parameter berupa token, sehingga REST API mengirimkan *response* error dengan pesan “Masukkan token terlebih dahulu”.

5. Kesimpulan

Dari tahap implementasi dan hasil dari penelitian dapat disimpulkan bahwa untuk membuat sebuah *web service* berupa REST API dapat menggunakan framework Lumen. REST API yang dikembangkan telah mengikuti sesuai dengan *guideline* rancangan REST API berbanding terbalik dengan sebelum rancang ulang dilakukan, URL REST API lebih mudah dibaca dan terorganisir dengan lebih baik. REST API yang dibangun juga dilengkapi dengan

keamanan berupa token untuk mengakses setiap *endpoints* URL yang memerlukan proses otorisasi. Dengan rancang ulang REST API ini, membuat keamanan data pelanggan PT. XYZ lebih menjadi aman lagi, karena tidak bisa diakses secara sembarangan.

Daftar Pustaka

[1] T. D. Wismarini and A. Prihandono, "Rancang Bangun Aplikasi Android Terintegrasi Web Service Dengan Volley Untuk Layanan Publik," *Dinamik*, vol. 25, no. 1, pp. 10–19, 2020, doi: 10.35315/dinamik.v25i1.7515.

[2] B. Mulyadi, Jaroji, and A. T, "Aplikasi Sistem Pemesanan Jasa Laundry (E-Laundry) Berbasis Android," *Zo. J. Sist. Inf.*, vol. 1, no. 1, pp. 48–57, 2019, doi: 10.31849/zn.v1i1.2386.

[3] E. Viglianisi, M. Dallago, and M. Ceccato, "RESTTESTGEN: Automated Black-Box Testing of RESTful APIs," *Proc. - 2020 IEEE 13th Int. Conf. Softw. Testing, Verif. Validation, ICST 2020*, pp. 142–152, 2020, doi: 10.1109/ICST46399.2020.00024.

[4] F. Halili and E. Ramadani, "Web Services: A Comparison of Soap and Rest Services," *Mod. Appl. Sci.*, vol. 12, no. 3, p. 175, 2018, doi: 10.5539/mas.v12n3p175.

[5] 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.

[6] I. M. Sukarsa, I. N. Piarsa, and I. G. B. P. Putra, "Penerapan Arsitektur MVP dalam Pengembangan Aplikasi Pemesanan," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 1, no. 10, pp. 7–11, 2020.

[7] A. A. Kristanto, Y. Harjoseputro, and J. E. Samodra, "Implementasi Golang dan New Simple Queue pada Sistem Sandbox Pihak Ketiga Berbasis REST API," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 4, pp. 745–750, 2020.

[8] R. K. Safitri and H. P. Putro, "Implementasi REST API untuk Komunikasi Antara ReactJS dan NodeJS (Studi Kasus : Modul Manajemen User Solusi247)," *Automata*, vol. 2, no. 1, pp. 0–4, 2021, [Online]. Available: https://journal.uii.ac.id/AUTOMATA/article/view/17381.

[9] E. Edy, F. Ferdiansyah, W. Pramusinto, and S. Waluyo, "Pengamanan Restful API menggunakan JWT untuk Aplikasi Sales Order," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 3, no. 2, pp. 106–112, 2019, doi: 10.29207/resti.v3i2.860.

[10] A. Rahmatulloh, H. Sulastri, and R. Nugroho, "Keamanan RESTful Web Service Menggunakan JSON Web Token (JWT) HMAC SHA-512," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 7, no. 2, 2018, doi: 10.22146/jnteti.v7i2.417.

[11] I. G. S. Masdiyasa, G. S. Budiwitjaksono, H. A. M, I. A. W. Sampurno, and N. M. I. M. Mandenni, "Graph-QL Responsibility Analysis at Integrated Competency Certification Test System Base on Web Service," *Lontar Komput. J. Ilm. Teknol. Inf.*, vol. 11, no. 2, p. 114, 2020, doi: 10.24843/lkjiti.2020.v11.i02.p05.

[12] G. Levin, "5 Basic REST API Design Guidelines," 2016. https://blog.restcase.com/5-basic-rest-api-design-guidelines/ (accessed Mar. 21, 2021).

[13] Riswandi, Kasim, and M. F. Raharjo, "Evaluasi Kinerja Web Server Apache menggunakan Protokol HTTP2," *J. Eng. Technol. Appl. Sci.*, vol. 2, no. 1, pp. 19–31, 2020, doi: 10.36079/lamintang.jetas-0201.92.

[14] M. M. Eyada, W. Saber, M. M. El Genidy, and F. Amer, "Performance Evaluation of IoT Data Management Using MongoDB Versus MySQL Databases in Different Cloud Environments," *IEEE Access*, vol. 8, pp. 110656–110668, 2020, doi: 10.1109/ACCESS.2020.3002164.

[15] H. Du, P. Jones, E. L. Segarra, and C. F. Bandera, "Development Of A REST API For Obtaining Site-Specific Historical And Near-Future Weather Data In EPW Format,"

Build. Simul. Optim. Conf., no. September, pp. 629–634, 2018.

[16] Y. Miftahuddin, M. Ichwan, and A. Zaky, "Kajian Routing pada Framework Laravel 5.0 dari Perspektif Penggunaan," *MIND J.*, vol. 2, no. 1, pp. 59–67, 2018, doi: 10.26760/mindjournal.v2i1.59-67.