API Gateway Menggunakan SlimPHP pada Aplikasi Kantin Amikom

API Gateway Using SlimPHP on Cafetaria Application in Amikom

Arif Dwi Laksito
Program Studi Informatika Universitas Amikom Yogyakarta
Jl. Ringroad Utara, Condong Catur, Depok, Sleman

arif.laksito@amikom.ac.id

Naskah diterima: 21 Mei 2019, direvisi: 28 Mei 2019, disetujui: 29 Juni 2019

Abstract

API has been widely used as a programming interface to integrate one device into another. The API Gateway is the layer that can communicate with the client. At this layer, the API Gateway can handle requests from the client and as a security layer, checking whether each request from the client is allowed to continue or not. Citramas Amikom has obstacles in implementing the canteen application, to reach all users it is necessary to use an internet connection in accessing the application. Whereas the previous system was in the local environment. The API Gateway has been successfully built to overcome these obstacles. In this research the API Gateway was developed using the SlimPHP framework and unit testing using the PHPUnit framework.

Keywords: *REST API, API Gateway, SlimPHP, PHPUnit.*

Abstrak

API telah banyak digunakan sebagai antarmuka pemrograman untuk mengintegrasikan perangkat satu ke perangkat yang lainnya. API Gateway merupakan layer yang menjadi satu-satunya gerbang bagi client. Pada lapisan ini, API Gateway dapat menangani request dari client dan sebagai lapisan keamanan, melakukan pengecekan apakah setiap request dari client diperbolehkan untuk dilanjutkan atau tidak. Pada koperasi Citramas Amikom terdapat kendala dalam mengimplementasikan aplikasi pemesanan online kantin, dimana untuk menjangkau seluruh pengguna perlu digunakan koneksi internet dalam akses aplikasi. Sedangkan sistem sebelumnya berada di lingkungan lokal. API Gateway telah berhasil dibangun untuk mengatasi kendala tersebut. Pada penelitian ini dilakukan pengembangan API Gateway menggunakan framework SlimPHP dan pengujian unit menggunakan framework PHPUnit.

Kata kunci: *REST API, API Gateway, SlimPHP, PHPUnit*

PENDAHULUAN

Application Programming Interface atau API telah banyak digunakan sebagai antarmuka pemrograman untuk mengintegrasikan perangkat satu ke perangkat lainnya. Arsitektur API yang saat ini banyak digunakan adalah REST (*Representational State Transfer*) yang menawarkan kemudahan dalam implementasinya seperti pada *standard HTTP* (*Hypertext Transfer Protocol*) yang selama ini telah digunakan. Beberapa penggunaan REST API telah diimplementasikan untuk layanan lokasi yang menyediakan informasi area secara abstraksi level tinggi sehingga dapat memenuhi kebutuhan spesifik aplikasi LBS (*Location Based Service*) (Akbar 2018) atau implementasi REST API untuk *Modular Home Automation* (Hasibuan et al. 2016). Selain itu REST API juga telah digunakan untuk *Association rule mining* dimana layanan ini dapat mendukung untuk jumlah transaksi 100.000 (Boonchuay, Intasorn, and Rattanaopas 2017).

Dengan meningkatnya jumlah aplikasi dalam instansi di mana tuntutan ketersediaan layanan API yang tinggi, akan muncul juga kendala dalam melakukan standarisasi dan konfigurasi masing-masing layanan tersebut. Sebagian besar pekerjaan dalam keamanan di *web service* telah memberikan solusi hanya untuk memastikan otentikasi, kerahasiaan, dan integritas informasi pada lapisan *network* bukan pada lapisan aplikasi (Rajaram, Babu, and Kishore Kumar 2014). Qiang Zhang telah melakukan penelitian dengan membangun *Unified API Gateway* untuk *high-availability* (HA) *cluster*. Penelitian tersebut menunjukan bahwa *API Gateway* dapat digunakan untuk mengelola layanan pada HA *cluster* yang berbeda dan juga dapat mengintegrasikan HA pada *platforms* yang berbeda (Zhang and Chu 2013). *API Gateway* berbasis SOA juga telah diterapkan untuk menyatukan HA *cluster* yang beragam (Li et al. 2013). *API Gateway* berfungsi sebagai gerbang dari beberapa API untuk manajemen, perlindungan data, dan meminimalkan *down-time* dari layanan (Akbar 2018).

Beberapa bahasa pemrograman dapat digunakan untuk membangun API, seperti PHP, Java, Python, atau Node.js berbasis Javascript. Penelitian terhadap perbadingan kinerja PHP dan Java menunjukan bahwa kinerja Java lebih baik pada jumlah *thread* di atas 100, sedangkan kinerja PHP lebih baik pada jumlah *thread* di bawah 100 (Pramana 2018). Sementara itu pada penelitian tentang perbandingan kinerja antara PHP, Python dan Node.js menunjukan bahwa Node.js memiliki durasi respon yang paling cepat, di mana PHP menyediakan performa yang paling konsisten di antara ketiga aplikasi, serta menyediakan keseimbangan antara konsumsi sumber daya dan kecepatan respon (Rompis and Aji 2018). Saat ini PHP merupakan bahasa pemrograman yang telah banyak digunakan untuk membangun aplikasi web termasuk juga mengimplementasikan REST API. PHP menawarkan kemudahan dalam pengembangan dan implementasi di *server* berbasis Linux atau Windows. Salah satu *framework* PHP yang mendukung dalam pengembangan API adalah SlimPHP. SlimPHP telah digunakan untuk mengembangkan Sistem Informasi Akademik di Politeknik kota Malang (Wijonarko and Mulya 2018), atau digunakan dalam mengembangkan API yang ditujukan untuk aplikasi Android Pendaftaran dan Transaksi di Klinik Hewan Bandung (Sabila, Rosely, and Nugroho 2018). Selain itu SlimPHP juga digunakan untuk membangun REST API untuk komunikasi pada IoT (*Internet of Things*) dan menunjukan bahwa sistem ini bisa mendukung komunikasi pada perangkat-perangkat yang berbeda standar (Setiawan et al. 2017).

Citramas merupakan koperasi yang berada di Universitas Amikom Yogyakarta dengan jumlah anggota lebih dari 200 orang yang terdiri dari staf pendidik, staf akademik, dan staf dari badan usaha Amikom. Terdapat layanan utama dari koperasi Citramas, yaitu kantin dan swalayan Citramart. Untuk menunjang layanan tersebut telah diimplementasikan aplikasi *point of sales* yang terintegrasi dengan sistem akuntansi dan sistem *inventory*. Untuk meningkatkan layanan

kepada anggota dan civitas Universitas Amikom, pengurus koperasi akan membangun aplikasi pemesanan online makanan kantin yang akan digunakan oleh para *tenant* sebagai penjual makanan dan seluruh civitas Universitas Amikom sebagai pembeli. Sistem utama dari koperasi Citramas telah dibangun sebelumnya pada lingkungan lokal intranet, di mana terdapat satu *server database* dan beberapa aplikasi berbasis desktop yang terinstal di komputer pengguna. Kendala yang dihadapi pengurus Citramas adalah ketika mengimplementasikan aplikasi pemesanan kantin online, di mana untuk menjangkau seluruh pengguna perlu digunakan koneksi internet dalam akses aplikasi, sedangkan sistem sebelumnya berada di lingkungan lokal.

Pada tahun 2000, Roy Fielding pada disertasinya memperkenalkan *Representational State Transfer (REST),* yaitu seperangkat aturan yang digunakan pada standar HTTP dan URI (*Uniform Resource Identifier)* (Bojinov 2016). Konsep utama dari *REST* adalah *resource* sebagai komponen dari suatu aplikasi yang memiliki alamat tertentu. *REST* dapat mengintegrasikan dengan lebih sederhana dan ringan dengan fokus pada *resource* (Tanaem, Manongga, and Iriani 2016). Terdapat 5 prinsip dalam membangun REST yaitu (Bojinov 2016):

1. Segalanya adalah *resource*. Setiap satuan data yang ada di internet mempunyai format untuk menidentifikasikan *content-type*. Sebagai contoh gambar JPEG, video MPEG, HTML, XML dan *file* binari didefiniskan dengan tipe: *images/jpeg, video/mpeg, text/html, text/xml dan application/octet-stream*.
2. Setiap *resource* dapat diidentifikasi dengan alamat yang unik. Dalam internet terdapat *resource* yang melimpah, dan setiap *resource* harus dapat diakses menggunakan URI yang unik.
3. Menggunakan standar metode HTTP. Terdapat 8 protokol HTTP sesuai dengan dokumen (RFC 2612) yaitu: *GET, POST, PUT, DELETE, HEAD, OPTIONS, TRACK*, dan *CONNECT*.
4. *Resource* dapat memiliki banyak representasi. Kunci utama dari *resource* pada *REST* adalah dapat direpresentasikan pada bentuk yang berbeda dengan menyesuaikan pada kemampuan *client*, contohnya: HTML, XML dan JSON.
5. Komunikasi yang *Stateless*. Dalam arsitektur *REST*, tidak membolehkan menyimpan *state* atau penanda dari *client* pada *server*. Seperti pada *session*, di mana *session* merupakan penanda *client* yang disimpan pada *server*. Keuntungan dari *stateless* ini adalah *server* dapat melayani masing-masing *request* secara mandiri dan *server* tidak perlu bertanggung jawab terhadap *state* pada *client*.

API *Gateway* merupakan layer yang menjadi satu-satunya gerbang bagi *client*. Pada lapisan ini, *API Gateway* dapat menangani *request* dari *client* dan sebagai lapisan keamanan, melakukan pengecekan apakah setiap *request* dari *client* diperbolehkan untuk dilanjutkan atau tidak.

Beberapa keunggulan menggunakan *API Gateway* adalah sebagai berikut (Sanchez and Vilariño 2017):

1. Aplikasi memiliki satu akses, di mana akan menghilangkan masalah dari *client* yang memerlukan masing-masing layanan *microservice*
2. Dapat mengakses penuh bagaimana layanan digunakan, dan dapat menyediakan *end-point* yang sesuai untuk kebutuhan *client*.
3. Mengurangi jumlah permintaan, yaitu hanya dengan satu permintaan saja *client* bisa mendapatkan informasi dari beberapa layanan.

SlimPHP adalah salah satu *framework* PHP dengan keunggulan penulisan kode yang ringkas dalam mengembangkan aplikasi web atau API. Menurut penciptanya, Josh Lockhart, Slim adalah PHP *micro framework* di mana fokus pada kebutuhan pokok dari aplikasi web seperti

menerima *HTTP request*, meneruskan *request* ke kode yang sesuai dan mengembalikan *HTTP response*. *Micro framework* biasanya digunakan untuk proyek skala kecil yang memiliki tujuan khusus dan tingkat kompleksitas yang rendah, oleh karena itu kinerja *micro framework* lebih cepat, lebih efisien dan lebih ringan jika dibandingkan dengan *fullstack framework* seperti Laravel atau Symfony. Slim ideal untuk membangun API, juga untuk membuat *prototype* aplikasi web. Dengan kompleksitas *framework* yang cukup ringkas, tidak memerlukan waktu yang lama untuk memahami *framework* ini. Ketika bekerja menggunakan Slim, kita akan menggunakan *Request* dan *Response* sebagai objek. Objek tersebut merepresentasikan *HTTP request* dan *HTTP response* pada arsitektur HTTP. Slim terdiri dari *route* yang akan merespon spesifik *HTTP request.* Setiap *route* akan meminta kembalian dalam bentuk *HTTP response*. Dalam implementasinya yang perlu dilakukan pertama kali adalah melakukan instansiasi dan konfigurasi Slim. Selanjutnya adalah mendefinisikan masing-masing *route* dan terakhir adalah menjalankan aplikasi Slim. Pada gambar 1 dibawah ini adalah contoh penulisan code di *framework* Slim (Lockhart, Smith, and Allen, n.d.).

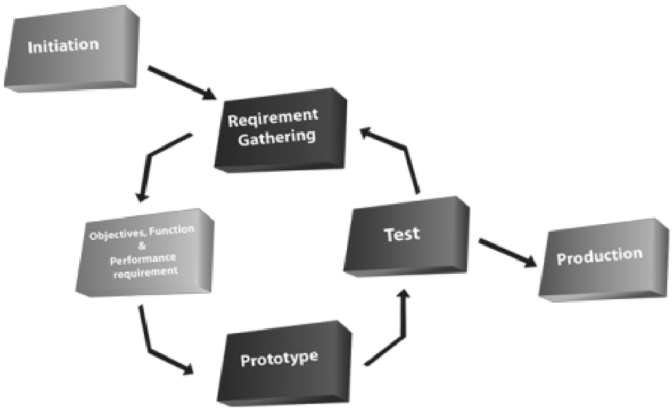
```php
<?php
// Create and configure Slim app
$config = ['settings' => [
    'addContentLengthHeader' => false,
]];
$app = new \Slim\App($config);

// Define app routes
$app->get('/hello/{name}', function ($request, $response, $args) {
    return $response->write("Hello " . $args['name']);
});

// Run app
$app->run();
```

Gambar 1. Contoh penulisan code di Slim (Lockhart, Smith, and Allen, n.d.)

Dari kendala yang dihadapi oleh Citramas maka perlu adanya sebuah jembatan untuk menghubungkan sistem kantin online dan aplikasi yang telah ada di lokal. Pada penelitian ini akan digunakan API *Gateway* menggunakan *protocol* komunikasi REST yang dibangun menggunakan *framework* SlimPHP untuk mengatasi masalah di Koperasi Citramas Amikom.

METODE

Peneliti melakukan perancangan aplikasi yang akan dibuat menggunakan pendekatan metode *prototype*. Pada metode *prototype* tidak menggunakan aturan yang kaku dalam pengembangan sistem, yaitu dengan tahapan awal membuat rancangan *prototype*, melakukan pengujian dan mengerjakan *design* ulang sampai kebutuhan sesuai. Metode ini mempunyai keunggulan dalam beberapa hal, yaitu kesalahan dapat dikenali dengan mudah, pengguna dapat berpartisipasi aktif dalam pengembangan, dan metode yang mudah dipahami (Saxena and Upadhyay 2016). Siklus dalam metode ini dijelaskan seperti pada Gambar 2 di bawah ini:

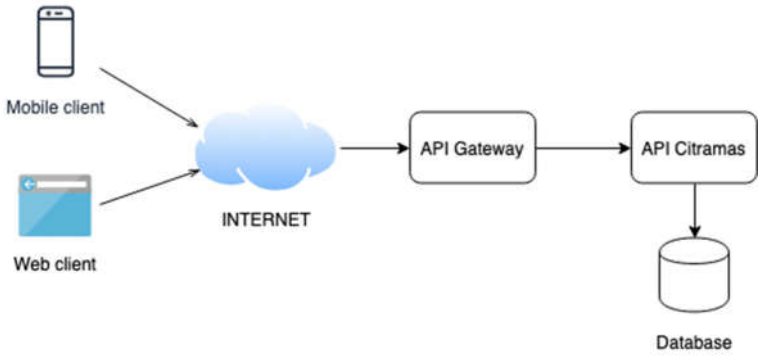


Gambar 2. Siklus metode *prototype* (Saxena and Upadhyay 2016)

Tahapan pada penelitian ini adalah sebagai berikut:

1. Identifikasi kebutuhan
2. Membangun *prototype*
3. Pengujian *prototype*
4. *Production*

Seperti penjelasan sebelumnya, saat ini sistem utama pada koperasi Citramas telah diimplementasikan dengan adanya API yang menunjang untuk aplikasi pemesanan kantin online. *API* Gateway ini akan diimplementasikan pada salah satu *server* di Universitas Amikom di mana dapat menghubungkan koneksi publik dengan lokal sistem koperasi Citramas. Arsitektur desain sistem terlihat seperti pada Gambar 3 berikut:



Gambar 3. Arsitektur desain sistem

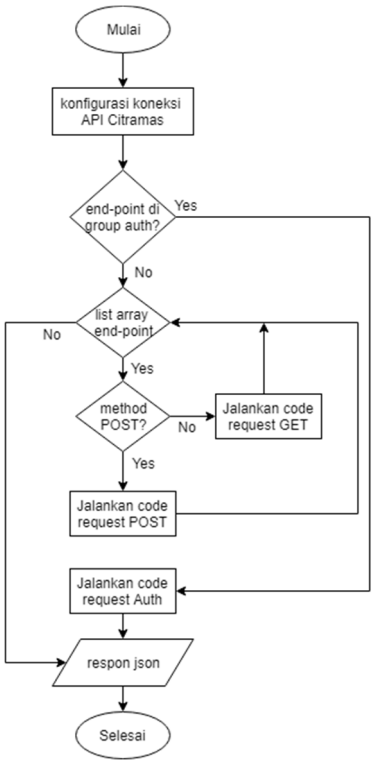
HASIL DAN PEMBAHASAN

API Gateway yang dibangun ini dikembangkan pada *server* dengan sistem operasi Unix dan menggunakan bahasa pemrograman PHP versi 5.6.40. *Framework* yang digunakan adalah SlimPHP di mana mempunyai keunggulan dalam kecepatan dan ringan saat dijalankan. Pada API di Citramas terdapat 2 method HTTP yang digunakan, yaitu GET dan POST dan terdapat 15 *end-point* yang terbagi menjadi 5 grup, yaitu:

Tabel 1. Daftar *End-point* pada API Citramas

No	*Group*	*End-point*	*Method*
1	Tenant	api/tenant/get_all	GET
2		api/tenant/get_by_status	GET
3	Menu	api/menu/get_all?tenant_id={id}	GET
4	Auth	api/auth/access_token	POST
5		api/auth/access_token/tenant	POST
6	Order	api/order/save	POST
7		api/order/proses	POST
8		api/order/cancel_order	POST
9		api/order/get_history_order? order_id={id}&include_detail=true	GET
10		api/order/get_history_order_with_pagging? customer_id={id}&include_detail=true& page_number=1&page_size=10	GET
11		api/order/get_history_order_today? customer_id={id}&include_detail=true	GET
12		api/order/get_history_order_tenant_today? tenant_id={id} &include_detail=true	GET
13		api/order/get_history_order_detail? order_id={id}	GET
14		api/order/get_history_order_tenant_with_pagging? tenant_id={id}&include_detail=true& page_number=1&page_size=10	GET
15	Customer	api/customer/get_deposit? customer_id={id}	GET

Berdasarkan dari kebutuhan akses *end-point* ke API di koperasi Citramas, maka aplikasi *API Gateway* yang dibangun ini mempunyai alur kerja seperti pada Gambar 4 *flowchart* di bawah ini:



Gambar 4. *Flowchart code API Gateway*

Konfigurasi koneksi pada *framework* Slim dilakukan dengan membuat variable *array* dengan *array index* "domain", selain itu bisa juga ditambahkan *array index* "*display Error Details*" untuk menampilkan *error* atau tidak, konfigurasi ini diperlukan untuk memisahkan lingkungan pengembangan atau *production.* Kode untuk konfigurasi terlihat seperti pada Gambar 5.

```
$config = array(
    'domain' => "http://172.16.47.3:4000/api",
    'displayErrorDetails' => false
);

$app = new \Slim\App(['settings' => $config]);
$container = $app->getContainer();
```

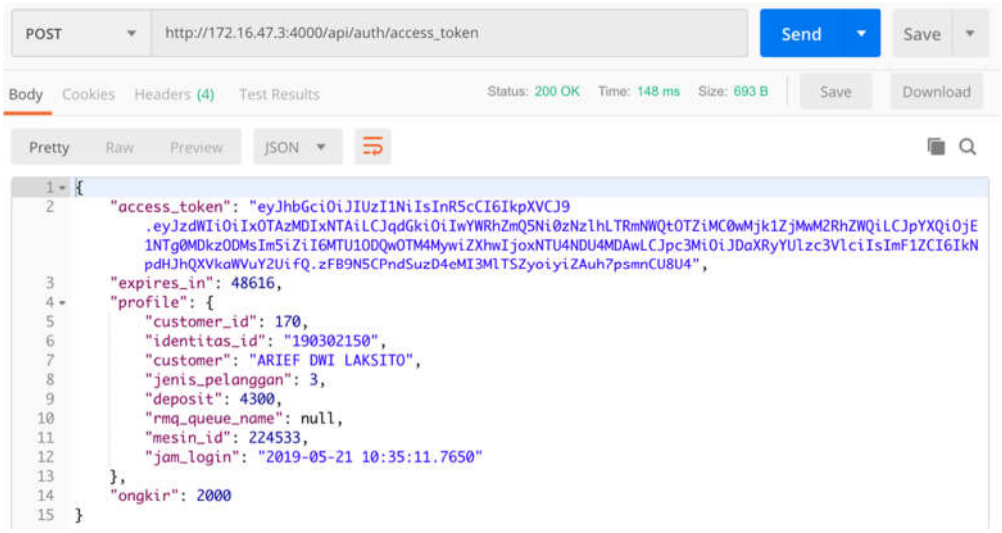
Gambar 5. Kode konfigurasi koneksi ke API

Selanjutnya untuk *request* pada group *auth* yaitu *end-point* api/auth/access_token dan api/auth/access_token/tenant, digunakan kode PHP yang diperlukan untuk melakukan enkripsi pada *input username* dan *password*. Selain *request* dari *group auth*, proses akan diteruskan ke tahap selanjutnya yaitu pemilihan dari *end-point* yang di *request* dengan menggunakan perulangan dari daftar *end-point* yang telah di definisikan. Berikutnya akan dilakukan pemilihan kode dimana *request* menggunakan *method* POST atau GET, seperti pada kode di Gambar 6. Untuk melakukan *request* ke API di koperasi Citramas pada kode PHP ini digunakan *library* RestClient yang dibuat oleh Travis Dent pada halaman *Github* untuk memudahkan dalam penggunaannya (Dent 2017).

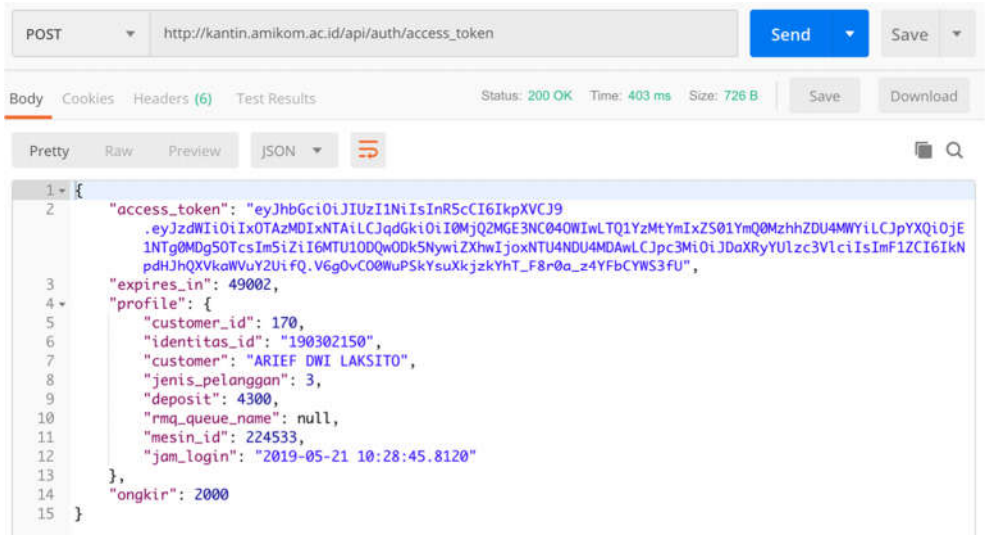
```
$header = $_SERVER['REDIRECT_REDIRECT_HTTP_AUTHORIZATION'];

$api = new RestClient([
    'base_url' => $config['domain'],
    'headers' => ['Authorization' => $header]
]);

if ($item['method'] == 'post') {
    $val = $req->getBody();
    $result = $api->post($item['endpoint'], $val,
        array('Content-Type' => 'application/json'));

} else {
    $query = $req->getUri()->getQuery();
    $result = $api->get( url: $item['endpoint'] . '?' . $query);
}
```

Gambar 6. Kode *request* API *method* POST dan GET

Pengujian *API Ga*teway ini menggunakan metode *blackbox* dan *whitebox.* Pengujian *blackbox* akan dilakukan uji kesesuaian antara akses ke API di koperasi Citramas dengan akses ke *API Gateway* yang dilakukan oleh *front-end developer,* yaitu pengembang aplikasi pada bagian *mobile client* dan *web client*. Pengujian ini dilakukan menggunakan *tools* Postman berdasarkan 15 *end-point* yang ada. Pada pengujian di API koperasi Citramas untuk *end-point api/auth/access_token* didapatkan hasil seperti pada Gambar 7 sedangkan pada pengujian di *API Gateway* dengan *end-point* yang sama menampilkan hasil seperti pada Gambar 8.



Gambar 7. Hasil *request* end-point api/auth/access_token pada API koperasi Citramas



Gambar 8. Hasil *request end-point* api/auth/access_token pada API *Gateway*

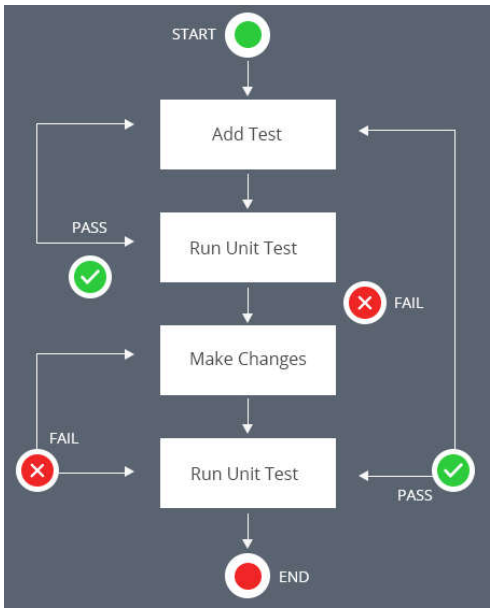
Response dari *request* ke end-*point* pada grup *auth* menghasilkan *access*_token di mana nilai tersebut digunakan sebagai *authorization* pada akses *end-point* yang lainnya. Pengujian dari masing-masing *end-point* dengan menggunakan Postman didapatkan hasil seperti pada Tabel 2.

Tabel 2. Uji kesesuaian API masing- masing *end-point*

No	*End-point*	Hasil
1	api/tenant/get_all	Sesuai
2	api/tenant/get_by_status	Sesuai
3	api/menu/get_all?tenant_id={id}	Sesuai
4	api/auth/access_token	Sesuai
5	api/auth/access_token/tenant	Sesuai
6	api/order/save	Sesuai
7	api/order/proses	Sesuai
8	api/order/cancel_order	Sesuai
9	api/order/get_history_order? order_id={id}&include_detail=true	Sesuai

No	*End-point*	Hasil
10	api/order/get_history_order_with_pagging? customer_id={id}&include_detail=true& page_number=1&page_size=10	Sesuai
11	api/order/get_history_order_today? customer_id={id}&include_detail=true	Sesuai
12	api/order/get_history_order_tenant_today? tenant_id={id} &include_detail=true	Sesuai
13	api/order/get_history_order_detail? order_id={id}	Sesuai
14	api/order/get_history_order_tenant_with_pagging? tenant_id={id}&include_detail=true& page_number=1&page_size=10	Sesuai
15	api/customer/get_deposit? customer_id={id}	Sesuai

Pada pengujian *whitebox* digunakan *framework* PHPUnit untuk melakukan uji kesesuaian algoritma dalam aplikasi API *Gateway*. PHPUnit merupakan *tools* unit *testing* yang unggul dalam hal fungsionalitas, efisiensi, kehandalan dan portabilitas dibandingkan dengan *tools* lain seperti *Codeception* dan *SimpleTest* (Sandin, Yassin, and Mohamad 2016). Kata Unit mengacu pada blok kode program, metode atau *class independent.* Pengujian unit adalah proses pengujian perangkat lunak di mana blok kode program diperiksa untuk memastikan apakah hasilnya sesuai dengan harapan (Nawaz 2018). Pengujian unit ini umumnya dapat dilakukan secara otomatis, tetapi bisa juga dimplementasikan secara manual.



Gambar 9. Alur kerja pengujian unit (Nawaz 2018)

PHPUnit menggunakan *assertion* untuk melakukan pengujian pada unit. Terdapat beberapa jenis *assertion* yang dapat digunakan seperti: *assertEquals* untuk membandingkan nilai ekspetasi dan nilai hasil pengujian, *assertNotEmpty* untuk melakukan validasi apakah hasil pengujian tidak kosong, *assertIsInt* untuk melakukan validasi apakah hasil pengujian berupa *integer*, *assertTrue* untuk validasi hasil pengujian bernilai *Boolean true*, *assertIsArray* untuk validasi apakah hasil pengujian berupa *array* dan beberapa jenis *assertion* lainnya.

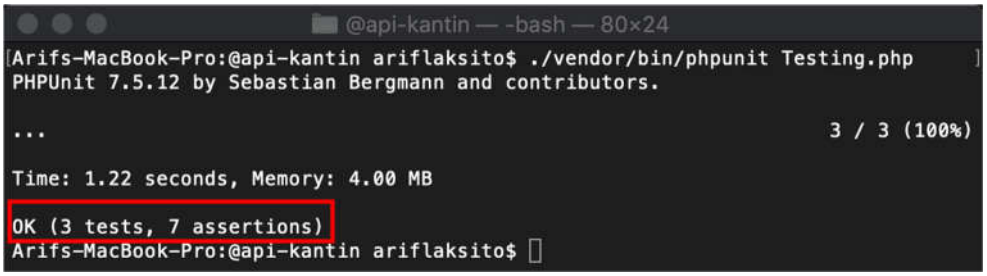
Pengujian *whitebox* ini dilakukan oleh peneliti atau *programmer* yang mengembangkan API *Gateway* dengan membuat kode PHP untuk melakukan pengujian terhadap 3 unit proses

yang ada di API *Gateway,* yaitu: *request* GET, *request* POST dan *request* Auth seperti yang telah disajikan pada gambar 4 diatas. Pada pengujian ketiga unit tersebut di PHPUnit perlu menggunakan *method* yang mewakili pengujian unit. Ketiga *method* tersebut dibuat dengan nama *testAuth*( ), *testGet*( ) dan *testPost*( ). Pada *testAuth*( ) akan digunakan 3 *assertion*, yaitu: *assertEquals*, *assertNotEmpty* dan *assertIsInt*. Adapun contoh *end-point* yang digunakan dalam *testAuth*( ), yaitu api/auth/access_token. Di tabel 3 berikut menampilkan detail dari *assertion* pada tiap-tiap unit pengujian beserta kegunaannya.

Tabel 3. Unit pengujian dengan *assertion* PHPUnit.

No	Unit Pengujian	Assertion	Keterangan
1	testAuth( )	assertEquals	Validasi apakah status *response code* dari API bernilai 200/Ok
		assertNotEmpty	Validasi *response token* dari API tidak kosong
		assertIsInt	Validasi *response* data customer_id berupa data *integer*/angka
2	testGet( )	assertEquals	Validasi apakah status *response code* dari API bernilai 200/Ok
		assertIsArray	Validasi *output* pada daftar *tenant* berupa data dalam bentuk array
3	testPost( )	assertEquals	Validasi apakah status *response code* dari API bernilai 200/Ok
		assertTrue	Validasi *output* pada status proses order menunjukan tipe *Boolean* bernilai true

Setelah dijalankan proses pengujian dengan PHPUnit melalui *command line* atau *terminal* menggunakan perintah /vendor/bin/phpunit Testing.php, menunjukan tidak terdapat *error* pada setiap *assertion* di unit tersebut. Gambar 10 berikut ini menunjukan *output* dari hasil pengujian di PHPUnit.



Gambar 10. Hasil pengujian di PHPUnit

Tahap terakhir di metode *prototype* adalah melakukan implementasi pada lingkungan *production*. Pada aplikasi berbasis web atau API istilah ini biasanya disebut dengan tahap *deployment*, yaitu proses melakukan pemindahan kode dari lingkungan pengembangan ke lingkungan *production.* Di sini peneliti menggunakan *tools* PHPloy yang dikembangankan oleh Baki Goxhaj untuk memudahkan proses *deployment* tersebut. PHPloy dibangun menggunakan bahasa pemrograman PHP dan bekerja berdasarkan *version control Git* di mana akan dilakukan otomatisasi pengiriman *file-file* yang telah diubah ke *production* (Goxhaj, n.d.). *Tools* ini memudahkan dalam pengembangan metode *prototype* di mana proses pengujian dan pengembangan ulang akan sering dilakukan.

Langkah awal menggunakannya adalah dengan men-*download* paket PHPloy dengan paket manager *composer* di alamat paket "banago/phploy". Selanjutnya perlu dilakukan konfigurasi pada *file* dengan nama “phploy.ini”. Konfigurasi ini digunakan untuk menyimpan informasi server *production* serta protokol untuk mengirimkan data ke server seperti *ftp* atau

sftp beserta informasi *user*, *password* dan *path* direktori. Pada gambar 11 disajikan contoh isi dari *file* konfigurasi *phploy*.ini

```
[production]
user = username
pass = s0m3p455w0rd
host = yourdomain.com
path = /public_html/folder
exclude[] = 'phploy'
exclude[] = 'phploy.ini'
exclude[] = 'composer.json'
exclude[] = 'composer.lock'
port = 21
passive = true
```

Gambar 11. Konfigurasi PHPloy pada file phploy.ini.

Tools PHPloy ini dijalankan menggunakan *command line* atau terminal pada direktori bin di mana terdapat *file executable* phploy. Untuk melihat daftar *file* yang di-*update* menggunakan perintah phploy -l, sedangkan untuk menjalankan proses *deployment* menggunakan perintah *phploy*. Sementara itu untuk mengembalikan atau membatalkan proses *deployment* sebelumnya digunakan perintah *phploy -rollback*.

PENUTUP

Dari tahapan pengembangan dan uji coba tersebut dapat diambil kesimpulan bahwa API Gateway menggunakan *framework* SlimPHP berhasil diimplementasikan sebagai penghubung antara API yang berada pada lingkungan lokas di koperasi Citramas dengan aplikasi kantin online yang berjalan menggunakan koneksi internet. Pada aplikasi API *Gateway* tersebut terdapat kode untuk memisahkan metode *request POST* dan *GET*. Pengujian menggunakan metode *blackbox* menunjukan bahwa masing-masing *end-point* dapat diakses dengan hasil yang sesuai. Adapun pada pengujian menggunakan metode *whitebox* berhasil dilakukan pengujian pada 3-unit dengan beberapa assertion pada setiap unit. Penelitian selanjutnya perlu dilakukan pengujian untuk kecepatan akses dalam penggunaan API *Gateway*. Selain itu perlu adanya lagi fitur yang lain dalam implementasi API *Gateway* ini seperti manajemen aplikasi atau manajemen akses untuk aplikasi *client* yang menggunakan API tersebut.

UCAPAN TERIMA KASIH

Kami ucapkan terimakasih kepada Fakultas Ilmu Komputer Program Studi Informatika dan Lembaga Penelitian Universitas Amikom Yogyakarta yang telah mendukung dalam melaksanakan penelitian ini.

DAFTAR PUSTAKA

Akbar, Muhammad. “Pengembangan Restful Api Untuk Application Specific High Level Location Service.” UNIVERSITAS ISLAM INDONESIA, 2018.

Bojinov, Valentin. *RESTful Web API Design with Node.js*. Second Edi. Birmingham, UK: Packt Publishing Ltd, 2016.

Boonchuay, Kesinee, Youppadee Intasorn, and Kritwara Rattanaopas. “Design and Implementation a REST API for Association Rule Mining.” *ECTI-CON 2017 - 2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, (2017): 668–71. doi:10.1109/ECTICon.2017.8096326.

Dent, Travis. “PHP REST Client.” 2017. https://github.com/tcdent/php-restclient.

Goxhaj, Baki. “PHPloy.” Accessed May 21, 2019. https://github.com/banago/PHPloy.

Hasibuan, Atas, Muhammad Mustadi, Ir Eniman Y. Syamsuddin, and Ir M.Anis Rosidi. “Design and Implementation of Modular Home Automation Based on Wireless Network, REST API, and WebSocket.” *International Symposium on Intelligent Signal Processing and Communication Systems, ISPACS 2015*. IEEE, (2016): 362–67. doi:10.1109/ISPACS.2015.7432797.

Li, Mingyu, Qian Zhang, Hanyue Chu, Xiaohui Hu, and Fanjiang Xu. “ConHA: An SOA-Based API Gateway for Consolidating Heterogeneous HA Clusters.” *IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, (2013): 1–3. doi:10.1109/CLUSTER.2013.6702645.

Lockhart, Josh, Andrew Smith, and Rob Allen. “Documentation - Slim Framework.” Accessed May 18, 2019. http://www.slimframework.com/docs/.

Nawaz, Shahroze. “PHP Unit Testing Using PHPUnit Framework.” 2018. https://www.cloudways.com/blog/getting-started-with-unit-testing-php/.

Pramana, I Putu Arya Sabha. “Perbandingan Performa Middleware Java dan Php pada Arsitektur Tiga Layer Berbasis Rest.” Universitas Islam Indonesia, 2018.

Rajaram, A. Kanchana, B. Chitra Babu, and C. R. Kishore Kumar. “API Based Security Solutions for Communication among Web Services.” *5th International Conference on Advanced Computing (ICoAC*) IEEE, (2013): 571–575. doi:10.1109/ICoAC.2013.6922014.

Rompis, Anugerah Christian, and Rizal Fathoni Aji. "Perbandingan Performa Kinerja Node. Js, PHP, Dan Python Dalam Aplikasi REST." *CogITo SMart Journal 4, no. 1* (2018): 171–87.

Sabila, Tiara, Ely Rosely, and Heru Nugroho. "Aplikasi Pendaftaran Dan Transaksi Pasien Klinik Hewan Di Bandung Berbasis Web." *eProceedings of Applied Science* 4, no. 3 (2018): 1499–1511.

Sanchez, Carlos Perez, and Pablo Solar Vilariño. *PHP Microservices*. Birmingham, UK: Packt Publishing Ltd, 2017.

Sandin, Easter Viviana, Noraniah Mohd Yassin, and Radziah Mohamad. "Comparative Evaluation of Automated Unit Testing Tool for PHP." *International Journal of Software Engineering and Technology* 3, no. 2 (2016): 7–11. doi:10.3345/kjp.2014.57.4.164.

Saxena, Aayushi, and Priya Upadhyay. "Waterfall vs . Prototype : Comparative Study of SDLC." *Imperial Journal of Interdisciplinary Research* 2, no 6 (2016); 1012–15.

Setiawan, Arif, I Wayan Mustika, Teguh Bharata Adji. "Mekanisme Otentikasi Berbasis Token Untuk Komunikasi REST Ada Internet of Things." *Prosiding Seminar Nasional Geotik, 2017* (39–44).

Tanaem, Penidas Fiodinggo, Danny Manongga, and Ade Iriani. "RESTFul Web Service Untuk Sistem Pencatatan Transaksi." *Jurnal Teknik Informatika Dan Sistem Informasi* 2, no. 1 (2016): 2443–2229.

Wijonarko, Dwi, and Betta Wahyu Retna Mulya. "Pengembangan Antarmuka Pemrograman Aplikasi Menggunakan Metode RESTful Pada Sistem Informasi Akademik Politeknik Kota Malang." *Smatika Jurnal* 8, no. 2 (2018): 63–66. doi:10.32664/smatika.v8i02.202.

Zhang, Qian, and Hanyue Chu. "A Unified API Gateway for High Availability Clusters." *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, IEEE, (2013): 2321–2325. doi:10.1109/MEC.2013.6885428.