Pengembangan REST API untuk Sistem Pelaporan dan Pengaduan dengan Laravel

(Studi Kasus : Balai Besar Karantina Pertanian Belawan)

Bayu Aries Wicaksono
Jurusan Informatika, Fakultas Teknologi Industri
Universitas Islam Indonesia
Yogyakarta
18523281@students.uii.ac.id

Irving Vitra Paputungan
Jurusan Informatika, Fakultas Teknologi Industri
Universitas Islam Indonesia
Yogyakarta
irving@uii.ac.id

Abstract*—Korupsi merupakan tindak pidana yang masuk kedalam kelompok kejahatan luar biasa, karena menimbulkan kesenjangan sosial serta ekonomi. Penanganan korupsi bukan hanya dilakukan oleh pemerintah tapi juga dilakukan oleh masyarakat dan instansi, salah satunya BBKP-Belawan. BBKP-Belawan membutuhkan sebuah aplikasi yang dapat membantu mempermudah pelaporan dan penanganan kasus korupsi yang terjadi di lingkungan BBKP-Belawan. BBKP-Belawan berencana mengembangkan sebuah aplikasi yang mampu memenuhi kebutuhan tersebut. Aplikasi yang akan dikembangkan adalah aplikasi Ayo BerAksi. Merupakan aplikasi *multiplatform* yang berbasis pada *website* dan *mobile*. Makalah ini akan difokuskan kepada pengembangan aplikasi *website* dari Ayo BerAksi. Aplikasi *website* dapat melakukan komunikasi dan pertukaran data dengan aplikasi *mobile*. Teknologi *web service* dimanfaatkan untuk mempermudah pengembangan *website.* Selain itu aplikasi ini juga akan memanfaatkan REST API sebagai standar komunikasi, arsitektur REST (*Representational State Transfer*) menggunakan alamat URI tertentu untuk melakukan *request* dan penggunaan berbagai *method* operasi seperti POST, GET, DELETE. Serta menggunakan JSON (*JavaScript Object Nation*) sebagai format pertukaran data. Metode yang digunakan dalam mengembangkan aplikasi *website* terdiri dari Analisis, Desain, Implementasi, serta Pengujian. Hasil dari pengembangan *website* dengan memanfaatkan REST API dapat meningkatkan efisiensi *bandwidth* dan hemat waktu dalam pertukaran data, serta dapat membantu *developer* mengembangkan *website* yang mampu berkomunikasi dan bertukar data dengan aplikasi lain.

Keywords—website, API, REST, web service, JSON

I. PENDAHULUAN

Perkembangan teknologi pada saat ini sangatlah pesat dan banyak dimanfaatkan untuk membantu berbagai kegiatan manusia, terutama teknologi yang berkaitan dengan sistem informasi yang dapat mempermudah pekerjaan dalam berbagai hal. Tidak hanya berkembang teknologi juga berubah dalam waktu yang sangat cepat. Teknologi baru selalu ditemukan untuk menyempurnakan atau bahkan menggantikan teknologi sebelumnya. Perkembangan teknologi informasi sendiri banyak dimanfaatkan untuk berbagai macam tujuan seperti membantu dalam efisiensi pekerjaan, membantu dalam memecahkan permasalahan, serta membantu dalam membuat hal baru.

Korupsi merupakan sebuah tindakan pidana yang dikelompokan menjadi kejahatan luar bisa karena dapat menyebabkan kesenjangan sosial, ekonomi, dan berbagai masalah lainnya. Banyak tindakan yang telah diusahakan oleh pemerintah untuk memberantas korupsi. Salah satunya yaitu membuat sebuah Undang-Undang Tipikor, yang diatur dalam Undang-Undang Nomor 31 Tahun 1999 jo Undang-Undang Nomor 20 Tahun 2001. Undang-Undang ini mengelompokan tindakan pidana korupsi menjadi 7 kelompok, yaitu korupsi yang merugikan negara, perbuatan curang, benturan kepentingan dalam pengadaan, pemerasan, suap menyuap, dan gratifikasi [1]. Usaha pemberantasan korupsi bukan hanya dilakukan oleh pemerintah tapi juga dilakukan oleh masyarakat dan instansi, salah satunya adalah instansi BBKP-Belawan.

Balai Besar Karantina Pertanian Belawan (BBKP-Belawan) adalah lembaga perkarantinaan hewan dan tumbuhan yang berlokasi di Medan, Sumatera Utara. BBKP-Belawan memiliki sebuah regulasi dalam menangani kasus penyuapan dan gratifikasi yang terjadi di lingkungan BBKP-Belawan sebagai upaya dalam memberantas korupsi. Namun saat ini regulasi pelaporan dan penanganan tindak korupsi masih menggunakan cara konvensional, dan masih memakan waktu yang cukup lama dalam penanganannya. Oleh karena itu, BBKP-Belawan membutuhkan aplikasi yang dapat digunakan sebagai pelaporan dan penanganan pada kasus penyuapan, gratifikasi dan pengaduan. BBKP-Belawan berencana menciptakan sebuah aplikasi yang dapat membantu BBKP-Belawan dalam memenuhi kebutuhan tersebut.

Aplikasi tersebut bernama Ayo BerAksi, merupakan aplikasi berbasis *website* dan *mobile*. Aplikasi ini bertujuan untuk mendigitalisasi dan mempermudah proses pelaporan serta penanganan pada kasus penyuapan, gratifikasi, dan pengaduan yang terjadi di lingkungan BBKP-Belawan. Aplikasi ini akan tersedia pada dua *paltform* yaitu pada *platform* berbasis *android* yang akan digunakan untuk melaporkan kejadian penyuapan maupun gratifikasi, dan *platform website* yang akan digunakan untuk tim belawan dalam menangani laporan yang masuk, serta menjadi sebuah *web service* yang menangani *request* dari *client.*

Pengembangan aplikasi Ayo BerAksi akan memanfaatkan teknologi REST API. Karena teknologi ini memungkinkan sebuah fungsi dan layanan yang ada pada *web service* untuk diakses oleh *client* (*android*) tanpa mengetahui detail yang ada di dalamnya [2]. Sehingga mempermudah kedua aplikasi untuk berkomunikasi. Selain itu, REST API juga cocok untuk digunakan pada sistem berbasisi *android* karena REST API ramah akan infrastruktur nirkabel, memanfaatkan *bandwith* lebih sedikit, dan memiliki ukuran *transfer* data yang kecil [3].

Aplikasi ini juga akan dikembangkan dengan *framework* Laravel. Karena Laravel merupakan sebuah *framework* PHP

yang mendukung struktur MVC (*model, view, controller*) untuk pengembangan aplikasi web [4]. Selain itu, laravel juga memiliki sintaks yang sederhana, dan jelas, serta memiliki banyak *library* yang memudahkan *developer* pengembangan aplikasi [5].

Pada makalah ini akan berfokus pada pengembangan aplikasi Ayo BerAksi berbasis *website*. Aplikasi Ayo BerAksi diharapkan dapat mempermudah masyarakat dan pegawai BBKP-Belawan dalam melaporkan kasus geratifikasi maupun penyuapan, serta membantu pegawai dalam memproses laporan yang masuk.

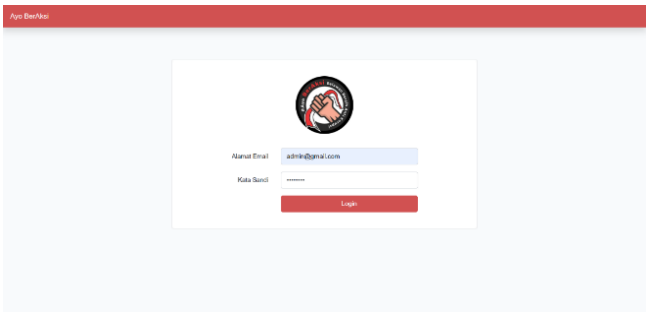
II. KAJIAN PUSTAKA

A. Aplikasi Ayo BerAksi

Aplikasi Ayo BerAksi merupakan aplikasi berbasis *multiplatform*, dimana aplikasi akan tersedia pada *website* dan *android* yang dikembangkan sebagai media dalam membantu proses penanganan laporan kasus penyuapan, gratifikasi, dan pengaduan.

Aplikasi Ayo BerAksi dapat digunakan oleh masyarakat dan pegawai BBKP-Belawan dalam melaporkan ketiga kasus tersebut kepada Tim Belawan. Tim Belawan merupakan tim yang ditunjuk untuk menyelidiki dan menangani laporan yang masuk. Setelah laporan diterima, terdapat tiga orang Tim Belawan yang akan menindaklanjuti laporan dari masyarakat. Tim tersebut ditugaskan untuk melakukan pengecekan terhadap laporan kasus.

Web dari Ayo BerAksi akan menjadi sebuah *web service* yang dapat diakses menggunakan API untuk komunikasi *server* (*website*) dengan *client* (*android*), dan memanfaatkan REST sebagai protokol komunikasi. Sehingga penyimpanan data dapat dilakukan pada satu *database* untuk kedua aplikasi serta untuk mempermudah dalam pengaksesan data yang ada. *Website* Ayo BerAksi memanfaatkan *framework* Laravel dalam pengembangannya. *Landing page* dari *website* dapat dilihat pada Gambar 1.



Gambar 1. Tampilan dari *website* Ayo BerAksi pada *landing page*

B. *REST*

Pada penulisan karya ilmiah ini, penulis menggali beberapa artikel, dan penelitian guna mendapatkan informasi yang akan menjadi landasan penulisan. Dimulai dari artikel [6], Menurut W3C yang merupakan lembaga penemu serta pengembang *web service*, *Representational State Transfer* (REST) diciptakan oleh *cofounder* dari Apache HTTP Server Project yaitu Roy Fielding. Menurut Fielding [7], REST merupakan sebuah standar web yang memanfaatkan protokol HTTP (*Hypertext Transfer Protocol*) dan URI (*Uniform Resource Identifier*) untuk berkomunikasi. Selain itu, menurut Fielding terdapat beberapa batasan dalam arsitektur REST diantara yaitu *client-server, cacheable, stateless, uniform interface*.

Pada Penelitian [8] yang membahas mengenai penerapan REST API pada sistem perpustakaan digital. Menjelaskan bahwa arsitektur REST bersifat *client-server* dimana *server* menyediakan layanan, mendengarkan *request* atas layanan tersebut dan mengirim kembali *response* kepada *client*. Serta REST bersifat *stateless* dimana setiap transaksi *request* bersifat independen. Selain itu, pada penelitian ini juga menjelaskan komunikasi antara *server* dan *client* memanfaatkan protokol HTTP dengan *method* GET, POST, PUT, dan DELETE untuk memanipulasi data seperti, mendapatkan, membuat, memperbarui, dan menghapus *resource.*

Dalam pengembangan *web service* terdapat dua *framework* populer yang sering digunakan yaitu SOAP (*Simple Object Access Protocol*) dan REST (*Representational State Transfer*). Berdasarkan penelitian [3], kedua *framework* memiliki ciri yang berbeda. Penelitian tersebut menjabarkan perbandingan antara SOAP dan REST seperti pada Tabel 1.

Tabel 1 Perbandingan REST dan SOAP

SOAP	**REST**
Merupakan teknologi lama.	Teknologi baru dibandingkan SOAP.
Cocok untuk *enterprise* dan B2B.	Belum cocok untuk *enterprise*, namun bisa diterapkan pada aplikasi perbankan.
Ketat dalam aturan *client-server.*	Lebih longgar dalam aturan *client-server*.
Jika pada *server* terdapat perubahan, maka *client* juga perlu perubahan.	Jika pada *server* terdapat perubahan, maka *client* tidak perlu perubahan.
Memiliki muatan data yang berat dibandingkan REST.	*Transfer* data sangat ringan karena menggunakan URI.
SOAP bukan infrastruktur nirkabel yang ramah	REST ramah infrastruktur nirkabel
Menggunakan lebih banyak *bandwith.*	Menggunakan lebih sedikit *bandwith*.
Lebih susah dikembangkan, dan membutuhkan *tools.*	Lebih mudah untuk dikembangkan.
SOAP *web service* mengembalikan data XML.	REST mendukung berbagai macam format data.
Sulit untuk membuat *cache response*.	Mudah untuk membuat *cache response.*

Selain itu pada penelitian [9] yang membahas mengenai Analisa Perbandingan Metode SOAP dan REST untuk Membangun *Web Service.* Dikatakan REST memiliki performa yang lebih tinggi dibandingkan dengan SOAP pada hasil pengujian *request* dan respon *web service.* Pada penelitian lain yang dilakukan oleh Asiyah et.al (2020), yang membahas mengenai Penerapan *Restful Web Service* untuk Optimalisasi Kecepatan Akses Android menyatakan bahwa REST memiliki *throughput* yang lebih sedikit dari SOAP, sehingga menyebabkan kecepatan akses pada REST jauh lebih cepat, bahkan hingga dua kali lipat dari kecepatan

akses SOAP [10]. Selain itu pada penelitihan [11] mengenai implementasi REST API untuk komunikasi antara ReactJS dan NodeJS, menjelaskan bahwa penggunaan REST API dalam pengembangan aplikasi mampu bekerja secara efisien karena tidak boros *bandwith* serta hemat dalam waktu pertukaran data.

Kemudian pada penelitian [2] membahas mengenai implementasi REST API untuk portal akademik berbasis android. Melalui metode ini, komunikasi antara aplikasi dengan *database server* memanfaatkan API dan *web service* dengan arsitektur REST yang memungkinkan *client* untuk mengakses *resource server* melalui *request* yang memanfaatkan protokol HTTP. Kelebihan dari penelitian tersebut adalah penggunaan REST API yang memungkinkan *client* menggunakan fungsi layanan yang ada pada *web service* tanpa mengetahui detail di dalamnya, sehingga dapat memberikan kemudahan untuk integrasi dan pengembangan selanjutnya. Sama seperti penelitian sebelumnya, penelitihan [12] yang menjelaskan tentang Pembangunan *web service* data masyarakat menggunakan REST API, komunikasi antara aplikasi kembali memanfaatkan API dan juga menggunakan *web service* namun untuk mendapatkan hasil respon atau *output* penelitian tersebut menggunakan JSON sebagai format pertukaran pesan.

Dalam pengimplementasian REST API terdapat berbagai macam *framework* yang dapat digunakan, seperti pada penelitian [9] yang menggunakan *framework* Flusk untuk membangun *web service*. Pemilihan Flusk sebagai *framework* karena Flusk lebih ringan dan dapat dengan cepat membuat *website* meski dengan *library* sederhana. Penelitian lain [5] yang menjelaskan tentang Implementasi REST API untuk Membangun Aplikasi *multiplatform*. Penelitian tersebut memanfaatkan *framework* Laravel. Keunggulan dari *framework* ini yaitu memiliki *template layout* yang ringan, memiliki banyak *library object oriented*, dan mendukung *framework* MVC.

Oleh karena itu, berdasarkan penelitian tersebut REST API paling sesuai untuk penelitian ini, dikarenakan REST API punya beberapa kelebihan diantaranya memiliki *transfer* data yang ringan, *bandwith* yang lebih kecil, waktu *response* yang lebih kecil, mendukung infrastruktur nirkabel, dan mudah untuk dikembangkan. Sehingga sesuai untuk pengembangan aplikasi berbasis *multiplatform* yang memanfaatkan API sebagai jembatan komunikasi antara *server* dan *client.*

III. METODOLOGI

Terdapat beberapa tahapan dalam penerapan REST API yaitu tahap analisis, desain, implementasi, dan pengujian. Berikut merupakan detail dari tahapan-tahapan di atas:

A. Analisis

Analisis tahap awal dilakukan oleh salah satu Tim *Developer* yaitu *Project Manager* (PM), dimana PM akan melakukan diskusi secara langsung dengan *client* dari pihak BBKP-Belawan untuk mencari tahu kebutuhan dari *client*. Pada tahapan ini, penulis yang berposisi sebagai *Back End Developer* tidak melakukan analisis secara langsung kepada *client*, karena tahapan ini dikerjakan secara langsung oleh PM. Hasil dari analisis tersebut akan menjadi data yang disusun menjadi sebuah dokumen pengembangan. Selanjutnya, PM akan meminta Tim *Developer* pada bagian UI/UX dan juga *Back End Developer* untuk melanjutkan tahapan pengembangan pada tahap desain sesuai dengan dokumen pengembangan yang sudah dibuat..

B. Desain

Pada tahap ini penulis sebagai *Back End Developer*, melanjutkan pengembangan berdasarkan dokumen pengembangan dari hasil analisis. Pengembangan dilanjutkan dengan membuat desain *use case diagram*. Selain itu penulis juga melakukan perancangan API untuk sistem Ayo-Beraksi menggunakan arsitektur REST yang dibangun diatas aplikasi berbasis *website*.

C. Implementasi

Pada tahapan implementasi akan lebih dijelaskan bagaimana penulis mengembangkan sebuah sistem dengan mengimplementasikan hasil desain yang telah dibuat pada tahapan sebelumnya. Sistem yang dibangun berupa API *web service* dengan bahasa pemrograman php.

D. Pengujian

Pada tahap pengujian, penulis akan melakukan pengujian terhadap sistem untuk memastikan API akan bekerja dengan baik. Pengujian akan dilakukan dengan memanfaatkan sebuah *tools* yaitu aplikasi Postman. Penulis akan memasukan alamat URI dari fungsi yang ada pada *web service*, kemudian menjalankan *request* dengan mengatur *field-field* data yang diperlukan kepada *web service.*

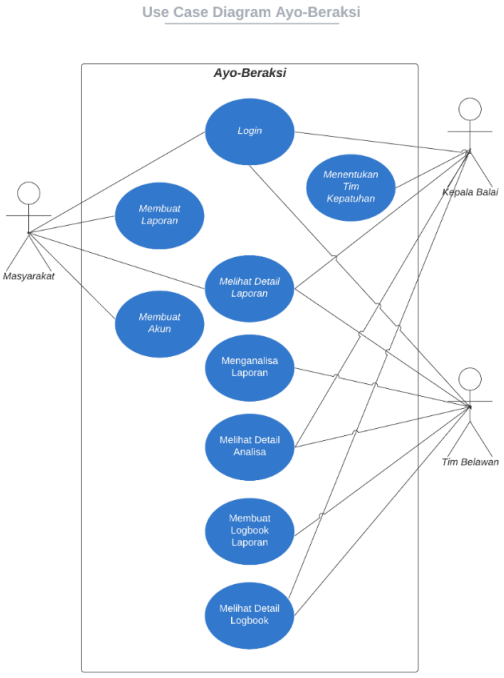
IV. HASIL DAN PEMBAHASAN

Pada bab ini, akan dibahas secara jelas hasil dari tahapan-tahapan yang telah disebutkan pada bab sebelumnya.

A. Desain

1) Use Case Diagram

Use case diagram digunakan untuk menggambarkan hubungan antara sistem dan *stakeholder*. Sistem yang penulis kembangkan memiliki 3 *stakeholder*, yakni masyarakat, kepala balai, dan tim kepatuhan. Untuk lebih jelas mengenai peran dari stakeholder dapat dilihat pada Gambar 2.



Gambar 2. Use Case Diagram

Use Case Diagram pada Gambar 2 menggambarkan kegiatan apa saja yang bisa dilakukan oleh masyarakat, kepala balai, dan juga tim kepatuhan di dalam sistem. Seluruh *stakeholder* perlu melakukan *login* terlebih dahulu sebelum mengakses sistem. Masyarakat dapat membuat akun, membuat laporan, dan melihat detail dari laporan yang akan berupa status laporan.

Kepala Balai dapat melihat detail laporan, kemudian menentukan tim kepatuhan, melihat detail dari hasil analisa laporan, dan melihat detail dari logbook laporan tim.

Tim Kepatuhan dapat melihat detail dari laporan yang ditunjuk untuk mereka selidiki, menganalisa laporan yang diselidiki, melihat detail analisa laporan, membuat logbook terkait laporan, dan melihat detail logbook laporan.

2) Perancangan REST API

Tahapan ini akan menjelaskan hasil dari perancangan REST API yang telah dibuat. REST API yang penulis kembangkan akan mempresentasikan hubungan antara *server* dengan *client* pada sistem. Komunikasi antara keduanya akan dijembatani oleh API yang dibuat dengan mengimplementasikan arsitektur REST. Beberapa *method* yang digunakan untuk melakukan *request* antara lain:

a. *Request method* GET yang digunakan untuk membaca data *resource*.
b. *Request method* POST yang digunakan untuk menambah data *resource*.
c. *Request method* PUT yang digunakan untuk mengubah data *resource*.
d. *Request method* DELETE yang digunakan untuk menghapus data *resource*.

B. Implementasi

Tahapan implementasi akan menjelaskan bagaimana pengembangan dilanjutkan dengan mengimplementasikan tahap sebelumnya. Pada tahapan ini API akan diimplementasikan dengan memanfaatkan Laravel Passport.

1) Melakukan Konfigurasi API

Konfigurasi API dimulai dengan menginstall *package passport*, kemudian memanggil *package* tersebut pada folder config/app.php, dan juga melakukan perubahan pada folder config/auth.php. Tujuan dari dari konfigurasi ini untuk menginstruksi *project* Laravel yang akan menjadi REST *server* untuk menggunakan *package* Laravel Passport saat mengautentikasi *request* API dari *client*.

2) Membuat *Routes* API

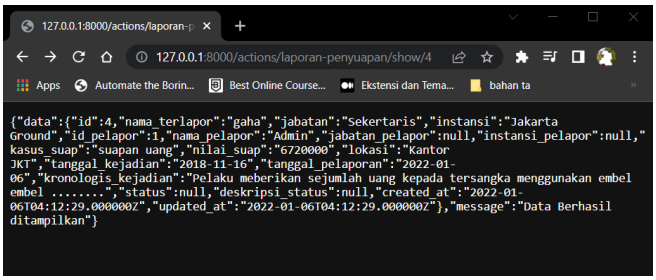
Pembuatan *route* API akan didefinisikan pada folder routes/api.php yang sudah disediakan oleh *framework* Laravel. *Routes* akan memanfaatkan beberapa *method* dari arsitektur REST yang telah didukung oleh protokol HTTP. *Method* yang digunakan, antara lain yaitu POST, PUT, GET, dan DELETE. Sebagian daftar *endpoint* API dapat dilihat pada Tabel 2.

Tabel 2. Endpoint API

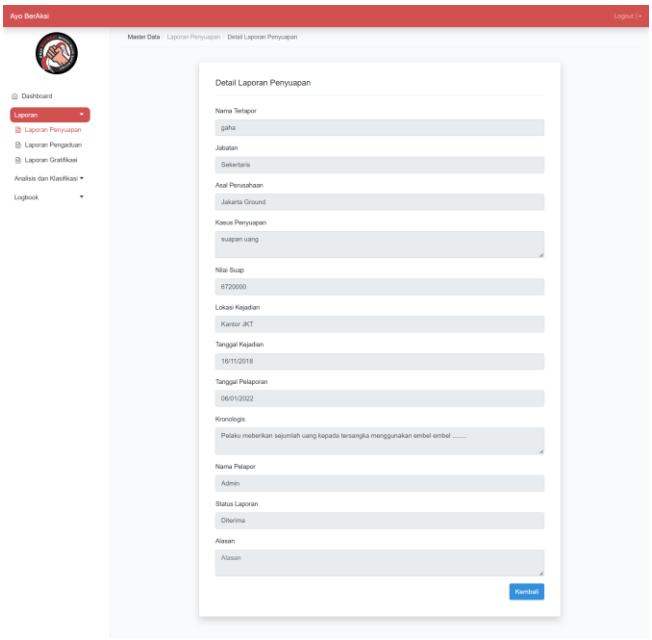
No	***Request Method***	***Endpoint***	**Keterangan**
1	POST	/register	Membuat akun
2	POST	/login	Login
3	POST	/actions/laporan-penyuapan/add	Membuat laporan penyuapan baru
4	PUT	/actions/laporan-penyuapan/update/{id}	Mengubah laporan penyuapan berdasarkan ID
5	GET	/actions/laporan-penyuapan/get-list	Menampilkan daftar laporan penyuapan
6	GET	/actions/laporan-penyuapan/show/{id}	Menampilkan detail laporan penyuapan berdasarkan ID
7	DELETE	/actions/laporan-penyuapan/delete/{id}	Menghapus laporan berdasarkan ID

Endpoint API akan mengakses *controller* yang sudah dibuat, dimana *controller* dibuat agar dapat mencakup semua *endpoint* sehingga akan lebih efektif untuk pertukaran data. Contoh *request* dengan alamat http://127.0.0.1:8000/api/actions/laporan-penyuapan/show/4. Pada tahap ini implementasi masih berjalan menggunakan *database* lokal, oleh karena itu nama *domain* masih menggunakan "127.0.0.1" atau "*localhost*", dengan nomor *port* "8000".

Keluaran yang dihasilkan dari *request* di atas berupa data dengan format JSON. Gambar 3 menunjukan *output* atau keluaran balik yang akan diterima *client* dengan format JSON dan Gambar 4 menunjukan hasil *request* dari alamat yang disebutkan di atas.



Gambar 3 Hasil Percobaan *Request*



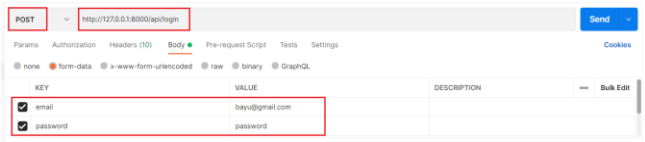
Gambar 4 Output Request pada *Website*

C. Pengujian

Tahapan pengujian merupakan tahapan terakhir dimana akan dilakukan pengujian kelayakan REST API yang sebelumnya sudah dibuat. Tahapan ini akan memastikan bahwa REST API yang dibuat sudah sesuai dan dapat bekerja dengan baik. Pengujian dilakukan dengan menggunakan sebuah *tools* yang berfungsi sebagai REST CLIENT yaitu Postman.

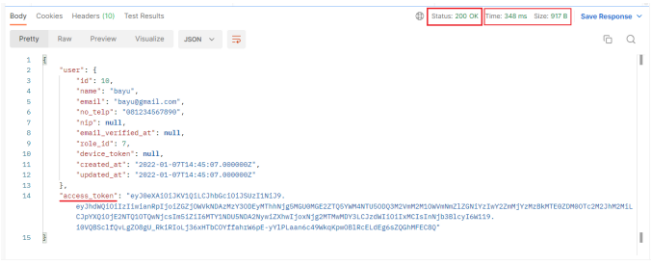
Pengujian REST dilakukan dengan mengakses *web server* yang sudah dibuat, menggunakan beberapa parameter diantaranya, POST, GET, dan DELETE. Dalam penggunaan Postman penulis perlu menyesuaikan *method* yang akan digunakan, memasukan URI *endpoint* dari fungsi yang akan dicoba, dan mengisi *field-field* yang diperlukan.

Sebagian pengujian API dapat dilihat pada gambar dibawah ini:



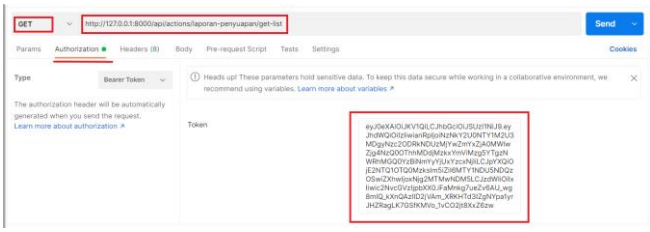
Gambar 5. Pengujian API */api/login*

Pengujian pada Gambar 5, digunakan untuk menguji *login* pengguna melalui *endpoint* /api/login. Parameter POST digunakan untuk pengujian *request* dan *response* POST pada REST, yang mana penulis perlu mengisi *field* data email dan *password* sebelum menjalankan pengujian.



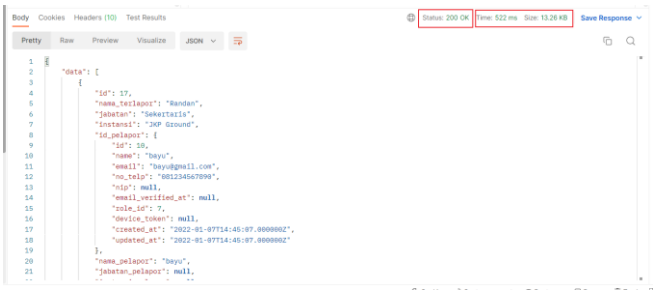
Gambar 6 Hasil *Response* API */api/login*

Setelah berhasil mengirim *request*, REST *server* akan memberikan keluaran berupa data dengan format JSON seperti pada Gambar 6, *status code* 200 yang menandakan *request* berhasil dikerjakan dengan waktu 348 ms, serta besar *bandwidth* sebesar 917 byte.



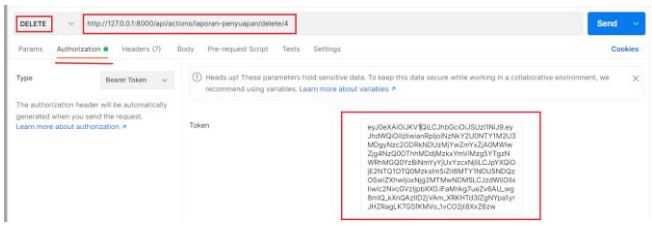
Gambar 7 Pengujian API */api/actions/laporan-penyuapan/get-list*

Kemudian pada pengujian lain seperti pada Gambar 7. Parameter GET digunakan untuk pengujian *request* dan *response* GET pada REST. Pengujian dilakukan dengan meminta *request* melalui *endpoint* /api/actions/laporan-penyuapan/get-list, dan memasukan akses *token* dari hasil *login* maupun *register user* yang memiliki *permission* untuk mengakses data tersebut.



Gambar 8 Hasil *Response* API */api/actions/laporan-penyuapan/get-list*

Hasil dari pengujian parameter GET untuk mengambil seluruh data laporan penyuapan yang ada pada *server* dengan format JSON seperti pada Gambar 8, *status code* 200 yang menandakan *request* berhasil dijalankan dengan waktu 522ms dan *bandwidth* sebesar 13.26 KB.



Gambar 9 Pengujian API */api/actions/laporan-penyuapan/delete/4*

Kemudian pada pengujian parameter DELETE seperti pada Gambar 9 untuk pengujian *request* dan *response* DELETE pada REST. Pengujian dilakukan seperti pada pengujian parameter GET, hanya saja *request* melalui *endpoint* /api/actions/laporan-penyuapan/delete/4 perlu menambahkan ID pada akhir *endpoint*, selain itu penulis juga perlu memasukan akses *token* dari *user* yang memiliki *permission* untuk mengakses data tersebut.

Gambar 10 Hasil *Response* API */api/actions/laporan-penyuapan/delete/4*

Hasil pengujian parameter DELETE untuk menghapus data laporan penyuapan berdasarkan ID seperti pada Gambar 10 Hasil *request* berhasil dimana *respon* menghasilkan *status code* 200, dengan waktu 491 ms, dan *bandwidth* sebesar 883 byte, serta hasil pesan yang menyatakan data berhasil dihapus. Hasil dari pengujian lainnya dengan menguji *endppoint* sebanyak lima kali untuk masing-masing *method* dapat dilihat pada tabel dibawah ini.

Tabel 3 Hasil Pengujian *Method POST* dengan 5 *Sample*

No	*Request Method*	Keterangan	Waktu (ms)	*Bandwidth* (byte)
1	POST	Membuat akun	412	925
2			378	925
3			501	925
4			501	925
5			454	925
Rata-rata			449,2	925

Tabel 4 Hasil Pengujian *Method PUT* dengan 5 *Sample*

No	*Request Method*	Keterangan	Waktu (ms)	*Bandwidth* (Kbyte)
1	PUT	Mengubah laporan penyuapan berdasarkan ID	616	1,08
2			507	1,08
3			629	1,09
4			624	1,08
5			1301	1,08
Rata-rata			735,4	1,082

Tabel 5 Hasil Pengujian *Method GET* dengan 5 *Sample*

No	*Request Method*	Keterangan	Waktu (ms)	*Bandwidth* (Kbyte)
1	GET	Menampilkan daftar laporan penyuapan	616	18,89
2			298	18,89
3			336	18,89
4			445	18,89
5			484	18,89
Rata-rata			435,8	18,89

Tabel 6 Hasil Pengujian *Method DELETE* dengan 5 *Sample*

No	*Request Method*	Keterangan	Waktu (ms)	*Bandwidth* (byte)
1	DELETE	Menghapus laporan berdasarkan ID	405	888
2			385	888
3			361	889
4			479	889
5			414	888
Rata-rata			408,8	888,4

Berdasarkan hasil pengujian dan pembahasan REST API sudah sesuai dengan penjelasan pada tahap desain hingga pengujian. *Server* REST API dapat dengan baik menerima *request* yang diminta oleh *client* untuk melakukan pertukaran data dengan memanfaatkan protokol HTTP. selain itu, hasil pengujian menggunakan *method* POST, PUT, GET, dan DELETE. *Size* dari *bandwith* pada proses *request* tergolong kecil dan cepat dalam menangani *request* untuk pertukaran data sehingga dapat dikatakan efisien karena tidak boros dalam penggunaan *bandwidth* dan hemat waktu.

Namun, pada saat ini penggunaan REST API masih belum memanfaatkan *cache* untuk menghindari akses data yang sama berulang-ulang. Hal tersebut dikhawatirkan akan menjadi sebuah halangan di masa depan.

V. KESIMPULAN DAN SARAN

Kesimpulan yang dapat diambil dari pembahasan bahwa API yang telah dibuat dengan menggunakan arsitektur REST telah berhasil mencapai tujuan, dimana sistem sudah bekerja dengan baik serta mampu berkomunikasi secara baik dengan *client* melalui seluruh *endpoint* API. Selain itu sistem juga mampu bekerja secara efisien karena penggunaan *bandwidth* yang tidak boros, dan hemat waktu.

Dengan memanfaatkan protokol HTTP, *client* dapat mengakses *resource* yang ada pada *server* REST API. Menggunakan *method* POST, PUT, GET, DELETE yang dimiliki operasi HTTP, serta penggunaan format JSON yang menjadi hasil keluaran karena format ini lebih mudah diolah dan lebih sederhana. Penggunaan REST API juga membantu *developer* untuk mengembangkan aplikasi yang dapat berkomunikasi dengan aplikasi lainnya.

Adapun saran untuk meningkatkan hasil penelitian selanjutnya yaitu, diperlukan penelitian lanjut untuk mengimplementasikan *cache* pada sistem terutama untuk menghindari akses data sama berulang kali pada *server*.

REFERENCES

[1] B. Waluyo, *Pemberantasan Tindak Pidana Korupsi (Strategi dan Optimalisasi)*. 2016.

[2] Y. K. Kurniawan, Y. Oslan, and H. Kristanto, "Implementasi Rest - Api Untuk Portal Akademik Ukdw Berbasis Android," *J. EKSIS*, vol. 6, pp. 29–40, 2013.

[3] K. Wagh and R. Thool, "A Comparative Study of SOAP Vs REST Web Services Provisioning Techniques for Mobile Host," *J. Inf. Eng. Appl.*, vol. 2, no. 5, pp. 12–16, 2012, [Online]. Available: http://www.iiste.org/Journals/index.php/JIEA/article/view/2063.

[4] Laravel, "Installation - Laravel - The PHP Framework For Web Artisans.," *Laravel*, 2019. https://laravel.com/docs/9.x/passport (accessed May 16, 2022).

[5] R. Choirudin and A. Adil, "Implementasi Rest Api Web Service dalam Membangun Aplikasi Multiplatform untuk Usaha Jasa," *MATRIK J. Manajemen, Tek. Inform. dan Rekayasa Komput.*, vol. 18, no. 2, pp. 284–293, May 2019, doi: 10.30812/matrik.v18i2.407.

[6] W3C, "Web Services Architecture," in *Architecting Web Services*, 2001, pp. 25–65.

[7] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," 2000.

[8] I. Mohidin and S. B. Musa, "PENERAPAN TEKNOLOGI REST API PADA APLIKASI PERPUSTAKAAN DIGITAL POLITEKNIK GORONTALO," vol. 10, no. 1, pp. 34–39, 2021.

[9] M. G. L. Putra and M. I. A. Putera, "Analisis Perbandingan Metode Soap Dan Rest Yang Digunakan Pada Framework Flask Untuk Membangun Web Service," *SCAN - J. Teknol. Inf. dan Komun.*, vol. 14, no. 2, pp. 1–7, 2019, doi: 10.33005/scan.v14i2.1480.

[10] L. N. Asiyah, M. P. T. Sulistyanto, and A. Aziz, "Penerapan Restful Web Service Untuk Optimalisasi Kecepatan Akses Pada Aplikasi Berbasis Android," *JOINTECS (Journal Inf. Technol. Comput. Sci.*, vol. 5, no. 2, p. 129, 2020, doi: 10.31328/jointecs.v5i2.1260.

[11] H. P. Safitri, R.K., dan 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.

[12] M. I. Perkasa and E. B. Setiawan, "Pembangunan Web Service Data Masyarakat Menggunakan REST API dengan Access Token," *J. Ultim. Comput.*, vol. 10, no. 1, pp. 19–26, 2018, doi: 10.31937/sk.v10i1.838.