

PENGEMBANGAN BACKEND BERBASIS REST API PADA ATSOFT E-GOVERNMENT DI CV. ATSOFT TEKNOLOGI



Disusun Oleh:

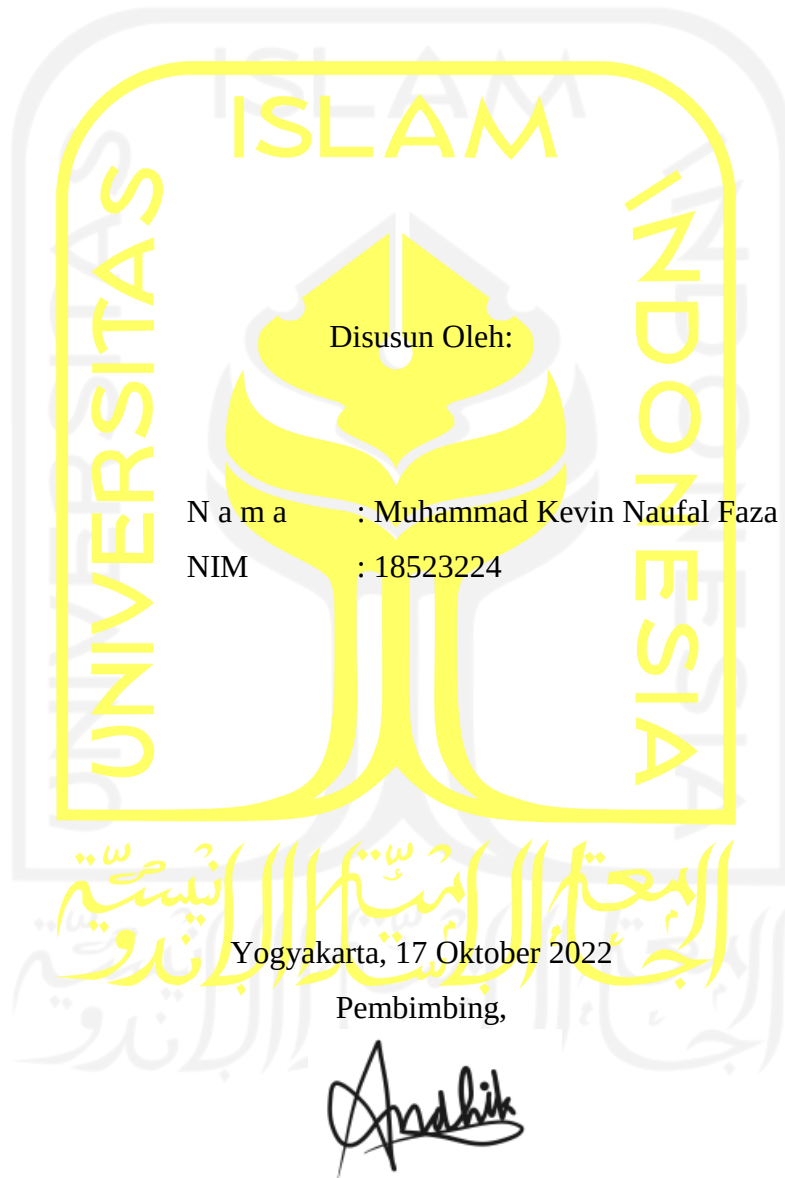
N a m a : Muhammad Kevin Naufal Faza Affrianto
NIM : 18523224

**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2022**

HALAMAN PENGESAHAN DOSEN PEMBIMBING

**PENGEMBANGAN BACKEND BERBASIS REST API PADA
ATSOFT E-GOVERNMENT DI CV. ATSOFT TEKNOLOGI**

TUGAS AKHIR JALUR MAGANG



(Andhik Budi Cahyono, S.T., M.T.)

HALAMAN PENGESAHAN DOSEN PENGUJI

**PENGEMBANGAN BACKEND BERBASIS REST API PADA
ATSOFT E-GOVERNMENT DI CV. ATSOFT TEKNOLOGI****TUGAS AKHIR JALUR MAGANG**

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 17 Oktober 2022

Tim Penguji

Andhik Budi Cahyono, S.T., M.T.

**Anggota 1**

Izzati Muhimmah, S.T., M.Sc., Ph.D.

**Anggota 2**

Arrie Kurniawardhani, S.Si., M.Kom.



Mengetahui,

Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia



(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Muhammad Kevin Naufal Faza Affrianto
NIM : 18523224

Tugas akhir dengan judul:

PENGEMBANGAN BACKEND BERBASIS REST API PADA ATSOFT E-GOVERNMENT DI CV. ATSOFT TEKNOLOGI

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 17 Oktober 2022



(Muhammad Kevin Naufal Faza Affrianto)

HALAMAN PERSEMBAHAN

Assalamualaikum Warahmatullahi Wabarakatuh. *Alhamdulillahirobbil'alam*, puji syukur kepada Allah SWT Yang maha Pengasih lagi Maha Penyayang dan tidak lupa salam serta shalawat kepada Nabi Muhammad Sallallahu Alaihi Wassalam. Sehingga, saya dapat menyelesaikan laporan Tugas Akhir ini tepat pada waktunya. Laporan ini saya persembahkan kepada orang tua yang telah mendoakan, menyemangati, mendukung, dan memotivasi saya. Laporan ini juga saya persembahkan untuk kerabat dan sahabat saya. Terimakasih telah menyemangati saya dan senantiasa memberi dukungan saat saya membutuhkan. ***Wassalamualaikum Warahmatullahi Wabarakatuh.***



HALAMAN MOTO

“Angin tidak berhembus untuk menggoyangkan pepohonan, melainkan menguji kekuatan akarnya.”

– Ali bin Abi Thalib

“Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya.”

– QS Al Baqarah 286

“Sukses berjalan dari satu kegagalan ke kegagalan yang lain, tanpa kita kehilangan semangat.”

– Abraham Lincoln



KATA PENGANTAR

Segala puji dan syukur selalu dihaturkan kepada Allah *Subhanahu Wata'ala*. atas berkat dan rahmat limpah-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul “PENGEMBANGAN BACKEND BERBASIS REST API PADA ATSOFT E-GOVERNMENT DI CV. ATSOFT TEKNOLOGI”.

Laporan ini dibuat untuk memenuhi persyaratan tugas akhir jalur magang di Fakultas Teknologi Industri jurusan Informatika Universitas Islam Indonesia. Penulis menyadari dalam penyusunan laporan Tugas Akhir penulis telah mendapatkan bantuan, arahan, dan bimbingan dari banyak pihak. Untuk itu, pada kesempatan ini penulis ucapkan rasa hormat dan terimakasih kepada semua pihak yang membantu. Pihak-pihak yang terkait diantaranya sebagai berikut:

1. Allah *Subhanahu Wata'ala* yang telah memberikan kehidupan, keselamatan serta kesehatan jasmani dan rohani untuk dapat melaksanakan magang dengan baik.
2. Bapak dan Ibu penulis yang telah memberikan do'a dan dukungan selama kegiatan magang berlangsung.
3. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku Ketua Program Studi Teknik Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Andhik Budi Cahyono, S.T., M.T., selaku dosen pembimbing yang telah bersedia membimbing dan mengarahkan penyusunan laporan ini.
5. CV. Atsoft Teknologi, beserta seluruh pimpinan, staf, dan karyawan yang telah memberi penulis kesempatan magang, dan membantu dalam menyelesaikan laporan tengah ini.
6. Bapak Andrey Feryyian sebagai pembina yang membimbing penulis secara teknis dalam tugas-tugas yang penulis kerjakan.

Selama proses penyusunan laporan Tugas Akhir penulis menyadari bahwa laporan ini masih belum sempurna. Sehingga mengharapkan kritik dan saran yang membangun dari pembaca. Akhir kata, semoga laporan ini dapat memberikan manfaat kepada penulis dan pembaca lainnya.

Yogyakarta, 17 Oktober 2022



(Muhammad Kevin Naufal Faza Affrianto)

SARI

Atsoft E-Government (ATGOV) merupakan proyek lanjutan dari SENGGUH (Sistem Evaluasi Pertanggungjawaban Pembangunan Daerah) yang dikembangkan oleh CV. Atsoft Teknologi. Kendala pada Sistem Informasi SENGGUH adalah sisi *client* tidak dapat mengambil data dari *server* secara langsung karena belum memiliki API. Dari permasalahan tersebut, dikembangkanlah Sistem Informasi ATGOV dengan menggunakan API berarsitektur *Representational State Transfer* (REST) sebagai *backend* dari sistem informasi tersebut. REST API dikembangkan menggunakan *framework* Phalcon dan menerapkan standar pengembangan seperti REST *server* dan REST *client*. ATGOV menggunakan REST API untuk mempercepat proses pertukaran data antara *client* dan *server*. Terdapat beberapa aktivitas yang dilakukan selama pengembangan REST API pada ATGOV mulai dari instalasi teknologi yang digunakan, perancangan, implementasi, dan pengujian REST API. Pada proses pengembangan, REST API menggunakan *endpoint* untuk memproses HTTP dengan method yang ada seperti GET, POST, dan DELETE dan *Output* yang dihasilkan menggunakan format *JavaScript Object Nation* (JSON) sehingga mudah dibaca oleh sisi *client* dan fleksibel. Hasilnya, pengembangan *backend* dengan implementasi REST API mampu memisahkan dan menyederhanakan interaksi antara *client* dan *server*. Perubahan yang terjadi pada *server* tidak mempengaruhi *client* sehingga kinerja *client* dan *server* menjadi lebih efisien karena kedua sisi dapat mengembangkan modul tanpa mempengaruhi proses pertukaran data.

Kata kunci: ATGOV, REST API, CV. Atsoft Teknologi.

GLOSARIUM

REST API	arsitektur perangkat lunak untuk pertukaran data
URI	alamat untuk mengidentifikasi sumber
JSON	format pertukaran data yang mudah dibaca oleh manusia
Resource	sumber daya yang digunakan dalam aplikasi perangkat lunak atau sistem
Framework	kerangka kerja pada pengembangan aplikasi berbasis desktop atau website
Frontend	segala hal yang berhubungan dengan tampilan antarmuka (UI) untuk pengguna pada suatu website atau aplikasi
Backend	segala hal yang berhubungan dengan sistem dibalik layar yang mengolah database dan server

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN DOSEN PEMBIMBING	ii
HALAMAN PENGESAHAN DOSEN PENGUJI	iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR	iv
HALAMAN PERSEMBAHAN	v
HALAMAN MOTO	vi
KATA PENGANTAR	vii
SARI.....	viii
GLOSARIUM	ix
DAFTAR ISI	x
DAFTAR TABEL	xii
DAFTAR GAMBAR.....	xiii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Ruang Lingkup.....	2
1.3 Tujuan	3
1.4 Manfaat	3
1.5 Sistematika Penulisan	4
BAB II LANDASAN TEORI	5
2.1 Sistem Informasi <i>E-Government</i>	5
2.2 Sistem Informasi Atsoft E-Government	6
2.3 Rencana Pembangunan Jangka Menengah Daerah.....	6
2.4 Rencana Strategis	7
2.5 <i>Application Programming Interface (API)</i>	7
2.6 REST API	8
2.7 <i>JavaScript Object Notation (JSON)</i>	8
2.8 Phalcon Framework	9
BAB III PELAKSANAAN MAGANG	12
3.1 Aktivitas Magang.....	12
3.1.1 Aktivitas di Bulan September.....	12
3.1.2 Aktivitas di Bulan Oktober dan November	13
3.1.3 Aktivitas di Bulan Desember, Januari	16
3.2 Pengembangan REST API pada Sistem Informasi Atsoft <i>E-Government</i>	16
3.2.1 Analisa Kebutuhan	17
3.2.2 Desain Arsitektur REST API	18
3.2.3 Implementasi REST API.....	19
3.2.4 Pengujian REST API.....	24
3.3 Tampilan antarmuka Aplikasi berbasis <i>website</i> Atsoft <i>E-Government</i> (ATGOV) 33	
3.3.1 Halaman Manajemen Rencana Pembangunan Jangka Menengah Daerah (RPJMD) Periode	33
3.3.2 Halaman tambah data Rencana Pembangunan Jangka Menengah Daerah (RPJMD) Periode	34
3.3.3 Halaman edit data Rencana Pembangunan Jangka Menengah Daerah (RPJMD) Periode	34
3.3.4 Halaman manajemen Rencana Strategis (Renstra) Program.....	34
3.3.5 Halaman tambah data Rencana Strategis (Renstra) program	35
3.3.6 Halaman edit data Rencana Strategis (Renstra) program.....	36
3.4 Mekanisme komunikasi antara atgov-fw dan atgov-api	36

	BAB IV REFLEKSI PELAKSANAAN MAGANG	40
4.1	Teknis.....	40
4.2	Non Teknis.....	41
	BAB V PENUTUP.....	42
5.1	Kesimpulan	42
5.2	Saran.....	42
	DAFTAR PUSTAKA	44
	LAMPIRAN.....	46



DAFTAR TABEL

Tabel 3.1 Aktivitas Magang.....	12
Tabel 3.2 Aktivitas bulan Desember, dan Januari	16
Tabel 3.3 Proses wawancara untuk pengumpulan data	18
Tabel 3.4 Salah satu daftar endpoint REST API pada proyek ATGOV	20
Tabel 3.5 Controller untuk mengelola Renstra program	22
Tabel 3.6 Daftar <i>endpoint</i> REST API RPJMD Periode.....	29
Tabel 3.7 Daftar <i>endpoint</i> API Renstra Program.....	33



DAFTAR GAMBAR

Gambar 2.1 Keuntungan menggunakan API	7
Gambar 2.2 Contoh penulisan dalam format JSON.....	9
Gambar 2.3 Perbandingan Framework Phalcon dan Symfony2	11
Gambar 3.1 Tampilan aplikasi desktop Github	15
Gambar 3.2 Tampilan aplikasi Discord	15
Gambar 3.3 Tahap pengembangan REST API	17
Gambar 3.4 Arsitektur REST API	18
Gambar 3.5 Konfigurasi docker-compose.yml pada proyek ATGOV	20
Gambar 3.6 Konfigurasi database ATGOV.....	21
Gambar 3.7 Contoh penggunaan method GET pada <i>controller</i>	21
Gambar 3.8 Output Data Dengan Format JSON pada Postman	22
Gambar 3.9 Model pada renstra program	23
Gambar 3.10 Kode <i>request</i> dengan method GET pada renstra program	24
Gambar 3.11 Tampilan login untuk mendapatkan token pengujian	25
Gambar 3.12 <i>Response</i> membaca data seluruh user berhasil	26
Gambar 3.13 <i>Response</i> membaca data berdasarkan <i>id</i> periode berhasil	26
Gambar 3.14 Langkah membuat RPJMD periode baru (Kiri) dan <i>Response</i> menambah data RPJMD periode berhasil (Kanan).....	27
Gambar 3.15 Langkah mengubah data RPJMD periode (Kiri) dan <i>Response</i> edit data RPJMD periode berhasil (Kanan).....	28
Gambar 3.16 Langkah menghapus data sesuai <i>id</i> RPJMD periode (Kiri), <i>Response</i> hapus data sesuai <i>id</i> RPJMD berhasil (Kanan).....	28
Gambar 3.17 <i>Response</i> ketika data hapus data RPJMD periode tidak ditemukan	29
Gambar 3.18 <i>Response</i> menampilkan seluruh data Renstra program.....	30
Gambar 3.19 Langkah mengambil data berdasarkan <i>id</i> Renstra program (Kiri) dan <i>Response</i> mengambil data berdasarkan <i>id</i> Renstra program (Kanan).....	30
Gambar 3.20 Langkah menambah data Renstra program (Kiri) dan <i>Response</i> menambah data Renstra program berhasil (Kanan).....	31
Gambar 3.21 Langkah mengubah data berdasarkan <i>id</i> Renstra program (Kiri) dan <i>Response</i> mengubah data berdasarkan <i>id</i> Renstra program berhasil (Kanan)	32
Gambar 3.22 Langkah menghapus data berdasarkan <i>id</i> Renstra program (Kiri) dan <i>Response</i> menghapus data berdasarkan <i>id</i> Renstra program berhasil (Kanan).....	32

Gambar 3.23 <i>Response</i> menghapus data berdasarkan <i>id</i> Renstra program tidak ditemukan...	32
Gambar 3.24 Tampilan halaman manajemen RPJMD periode	33
Gambar 3.25 Tampilan halaman tambah data RPJMD periode	34
Gambar 3.26 Tampilan halaman edit data RPJMD periode	34
Gambar 3.27 Halaman manajemen program Renstra	35
Gambar 3.28 Halaman tambah data Renstra program	35
Gambar 3.29 Halaman edit data Renstra program	36
Gambar 3.30 Struktur folder pada atgov-api (Kiri) dan atgov-fw (Kanan)	36
Gambar 3.31 Halaman manajemen program Renstra	37
Gambar 3.32 Form tambah data program Renstra	37
Gambar 3.33 Fungsi <code>insertAction()</code> pada <i>RenstraProgramController</i>	38
Gambar 3.34 Fungsi <code>insertAction()</code> pada <i>RenstraProgramController</i> atgov-api	38



BAB I PENDAHULUAN

1.1 Latar Belakang

Badan Perencanaan Pembangunan Daerah (BAPEDDA) DIY adalah pejabat teknis daerah yang bertanggung jawab kepada Walikota melalui Sekretaris Daerah pada bidang perencanaan pembangunan daerah. BAPEDDA DIY telah diresmikan dan ditetapkan dengan Keputusan Kepala Daerah Tingkat I DIY Nomor 341 Tahun 1974 tentang Pembentukan Badan Perencanaan Pembangunan Daerah Tingkat I DIY. BAPEDDA DIY membutuhkan aplikasi berbasis website yang dapat meningkatkan pengelolaan serta untuk mewujudkan perencanaan pembangunan daerah yang berkualitas. Sehingga CV. Atsoft Teknologi menawarkan solusi kepada BAPEDDA DIY berupa aplikasi berbasis *website* yang bernama Atsoft *E-Government* (ATGOV). BAPEDDA DIY mengharapkan ATGOV dapat bermanfaat untuk mempermudah penyusunan rencana strategis pembangunan daerah serta merealisasikan sasaran rencana strategis program yang telah direncanakan secara akurat, *real-time*, dan fleksibel.

Perusahaan CV. Atsoft Teknologi mengembangkan sistem informasi Atsoft *E-Government* (ATGOV) untuk melakukan pencatatan dan perancangan pembangunan daerah dengan memanfaatkan teknologi REST API. ATGOV merupakan proyek lanjutan dari SENGGUH (Sistem Evaluasi Pertanggungjawaban Pembangunan Daerah) yang dikembangkan oleh CV. Atsoft Teknologi. Terdapat 2 fitur yang membutuhkan tenaga kerja untuk merealisasikan fitur tersebut. Atsoft Teknologi. Fitur tersebut adalah RPJMD (Rencana Pembangunan Jangka Menengah Daerah) dan Renstra (Rencana Strategis). Pada fitur RPJMD, terdapat beberapa sub fitur RPJMD yang belum diselesaikan pada ATGOV yaitu RPJMD periode, RPJMD program, RPJMD sasaran, dan RPJMD tujuan. Sedangkan, pada fitur Rencana Strategis (Renstra) beberapa sub fitur yang belum diselesaikan adalah Renstra periode, dan Renstra program.

ATGOV dikembangkan dengan menggunakan pendekatan REST API. Penggunaan REST API pada aplikasi ATGOV bertujuan untuk memudahkan pertukaran data antara *client* dan *server* karena memiliki konfigurasi yang mudah dipahami dan mampu menyederhanakan interaksi antara *client* dan *server*. Perubahan yang terjadi pada *backend* atau *server* tidak mempengaruhi *client* sehingga kinerja *server* menjadi lebih efisien karena kedua sisi dapat mengembangkan modul tanpa mempengaruhi proses pertukaran data. Pada REST API terdapat REST *server* yang memberikan jalur untuk mengakses *resource* atau data dan REST *client*

yang dapat mengakses *resource* untuk menggunakan atau menampilkan data atau dapat dikatakan pertukaran antar *state* dimana *state* tersebut merupakan halaman *web* yang akan dipanggil oleh *browser* melalui tautan HTTP.

Laporan ini akan membahas pengembangan ATGOV dari sisi *backend* dengan mengimplementasikan REST API. Dikarenakan *client* dan *server* adalah bagian yang terpisah, maka REST API yang dikembangkan akan memiliki kemampuan untuk pertukaran data antara *client* (atgov-fw) dan *server* (atgov-api). REST API digunakan karena memiliki kelebihan dalam performa dan kecepatan transfer data dibandingkan dengan teknologi SOAP. Keluaran dari REST lebih flexibel yakni XML atau JSON sedangkan SOAP hanya XML.

1.2 Ruang Lingkup

Kegiatan resmi magang dilaksanakan selama 6 bulan dari awal bulan September 2022 sampai dengan awal bulan Maret 2023 yang diposisikan sebagai *Backend Developer*. Selama magang berlangsung terdapat satu proyek yang dikerjakan yaitu Atsoft *E-Government* (ATGOV). Pada proyek tersebut, tugas yang diberikan meliputi pengembangan fitur Rencana Pembangunan Jangka Menengah Daerah (RPJMD) dan fitur Rencana Strategis (Renstra) serta melakukan pengujian REST API pada kedua fitur tersebut. Fitur RPJMD berfungsi untuk memudahkan Satuan Kerja Perangkat Daerah (SKPD) dalam mengelola periode rencana pembangunan daerah selama 1 (satu) sampai 5 (lima) tahun sesuai dengan masa jabatan Kepala daerah. Sedangkan, fitur Renstra berfungsi untuk mempermudah SKPD dalam dokumentasi rumusan program untuk keberhasilan perencanaan strategis pembangunan daerah. Kedua fitur tersebut dipecah menjadi sub fitur yang lebih kecil, antara lain:

- a. Sub fitur RPJMD periode, yaitu sub fitur yang berfungsi untuk menampilkan periode Pemerintah daerah selama 1 (satu) sampai 5 (lima) tahun sesuai dengan masa pemerintahan kepala daerah
- b. Sub fitur RPJMD program, yaitu sub fitur yang berfungsi untuk menampilkan rumusan program rencana pembangunan jangka menengah daerah sesuai amanat dari kepala daerah dan berlangsung selama periode 5 (lima) tahun sesuai dengan berlangsungnya masa pemerintahan kepala daerah
- c. Sub fitur RPJMD sasaran, yaitu sub fitur yang berfungsi untuk menampilkan sasaran target BAPEDDA DIY dari program atau kegiatan selama 5 (lima) tahun mendatang yang mendukung pencapaian tujuan

- d. Sub fitur RPJMD tujuan, yaitu sub fitur yang berfungsi untuk menampilkan penjabaran tujuan dari tindakan perencanaan pembangunan daerah selama 5 (lima) tahun mendatang sesuai misi BAPEDDA DIY
- e. Sub fitur Renstra periode, yaitu sub fitur yang berfungsi untuk mempermudah SKPD dalam dokumentasi perencanaan strategis yang telah disusun selama 1 (satu) sampai 5 (lima) tahun sesuai dengan periode jabatan SKPD
- f. Sub fitur Renstra program, yaitu sub fitur yang berfungsi untuk mempermudah SKPD dalam dokumentasi rumusan program untuk keberhasilan perencanaan strategis pembangunan daerah

Dikarenakan pola pengembangan sub fitur lainnya mirip, hanya dua sub fitur utama yang akan dilaporkan pada laporan tugas akhir yaitu: RPJMD periode dan Renstra program. Sedangkan, sub fitur lain yang telah dikembangkan akan ditampilkan pada bagian lampiran.

1.3 Tujuan

Tujuan dari pengembangan *backend* berbasis REST API pada Sistem Informasi Atsoft *E-Government* (ATGOV) di CV. Atsoft Teknologi adalah sebagai berikut:

- a. Memisahkan dan menyederhanakan interaksi antara *client* dan *server* pada fitur Rencana Pembangunan Daerah Jangka Menengah Daerah (RPJMD) dan fitur Rencana Strategis (Renstra).
- b. Mengembangkan fitur Rencana Pembangunan Jangka Menengah Daerah (RPJMD) untuk memudahkan pemerintah daerah menjabarkan kebijakan pembangunan yang merespon kebutuhan masyarakat serta visi dan misi kepala daerah terpilih dalam tujuan, sasaran, dan program.
- c. Mengembangkan fitur Rencana Strategis (Renstra) untuk memudahkan pemerintah daerah menentukan strategi atau arahan serta pengambilan keputusan alokasi sumber daya.

1.4 Manfaat

Manfaat yang dihasilkan dari pengembangan REST API pada tersebut adalah kinerja *client* dan *server* menjadi lebih efisien karena kedua sisi dapat mengembangkan fitur tanpa mempengaruhi proses pertukaran data.

1.5 Sistematika Penulisan

Sistematika Penulisan ini menjelaskan secara keseluruhan dari setiap materi yang terdapat dalam laporan ini. Sistematika penulisan yang terdiri dari lima bab yaitu:

a. Pendahuluan

Bab ini menjelaskan dan menggambarkan latar belakang, ruang lingkup magang, tujuan, manfaat, dan sistematika penulisan.

b. Landasan Teori

Bab ini berupa pengertian dan definisi yang diambil dari media internet dan buku yang berkaitan dengan penyusunan laporan.

c. Pelaksanaan Magang

Bab ini menggambarkan kegiatan dan dokumentasi tugas yang dilaksanakan selama proses kegiatan magang.

d. Refleksi Pelaksanaan Magang

Bab ini berupa penjelasan refleksi dari hasil yang diperoleh selama pengembangan aplikasi *website* Atsoft E-Government, fitur Rencana Pembangunan Jangka Menengah Daerah (RPJMD), Rencana Strategis (Renstra).

e. Kesimpulan dan Saran

Bab ini berupa kesimpulan dan saran dari seluruh materi yang telah dijelaskan pada bab-bab sebelumnya.

BAB II

LANDASAN TEORI

2.1 Sistem Informasi *E-Government*

Menurut Mulyanto dan Kuswara dan Kusmana (2017:18), Sistem Informasi merupakan kumpulan suatu system yang terdiri dari *hardware*, *software*, dan *brainware* yang memproses informasi menjadi sebuah output yang bermanfaat untuk suatu perusahaan atau organisasi. *E-Government* adalah pemanfaatan teknologi informasi untuk meningkatkan ikatan antara pemerintah dengan pihak-pihak lain yang melibatkan otomatisasi dan komputerisasi. Sehingga, Sistem Informasi *E-Government* merupakan suatu teknologi informasi yang dapat memudahkan pelayanan yang berkaitan dengan pemerintahan.

Menurut Setiawan (2017), tujuan dari dibentuknya *E-Government* adalah untuk meningkatkan kualitas pelayanan masyarakat oleh pemerintah terutama dalam hal proses dan akses interaksi masyarakat. *E-Government* juga dapat meningkatkan transparansi pemerintah dengan cara memberikan akses informasi kepada publik. Selanjutnya, manfaat yang didapatkan dari adanya *E-Government* adalah dapat meningkatkan efektifitas dan kualitas kinerja pemerintah dalam hal memberikan layanan dan informasi kepada masyarakat.

Menurut Irawan (2020), suatu sistem informasi dapat dikatakan sebagai Sistem informasi *E-Government* apabila memenuhi karakteristik sebagai berikut:

- 1. *Government to Customer (G2C)***

Penyampaian layanan atau informasi satu arah dari pemerintah ke masyarakat dan memungkinkan adanya interaksi antara masyarakat dan pemerintah. Contohnya adalah layanan jaminan sosial, pajak online, dokumen pribadi, layanan kesehatan, beasiswa.

- 2. *Government to Business (G2B)***

Layanan transaksi elektronik yang disediakan oleh pemerintah kepada kalangan bisnis untuk melakukan interaksi dengan pemerintah sehingga dapat meningkatkan proses bisnis pemerintah. Contohnya adalah peluang bisnis, pendaftaran perusahaan, Hukum bisnis, pelelangan dan penjualan yang diselenggarakan pemerintah.

- 3. *Government to Government (G2G)***

Layanan yang memungkinkan komunikasi dan interaksi antar departemen atau lembaga pemerintahan melalui basis data yang terintegrasi. Contohnya adalah pendidikan online, pelayanan masyarakat terpadu, *blogging* untuk kalangan legislatif.

2.2 Sistem Informasi Atsoft E-Government

Sistem Informasi Atsoft *E-Government* (ATGOV) adalah aplikasi berbasis *website* yang menggunakan *Phalcon framework* dan merupakan lanjutan dari Sengguh (Sistem Evaluasi Pertanggungjawaban Pembangunan Daerah) yang telah dibuat CV. Atsoft Teknologi. Perusahaan tersebut mengubah Sengguh menjadi ATGOV karena Sengguh masih memiliki arsitektur yang kaku dan bersifat monolith (belum memiliki *web service* atau *Application Program Interface* (API)). Fungsi dari ATGOV adalah untuk meningkatkan pengelolaan serta pelayanan untuk melakukan perencanaan dan pencatatan pembangunan daerah di Daerah Istimewa Yogyakarta agar mencapai target Badan Perencanaan Pembangunan Daerah (BAPEDDA) DIY. BAPEDDA DIY menginginkan sistem informasi ATGOV menggunakan teknologi REST API sehingga dapat mempermudah pengembangan pada beberapa platform seperti: *website* dan *mobile*.

Penggunaan REST API pada sistem informasi ATGOV bertujuan untuk berbagi data dengan aplikasi lain dengan cara yang lebih efisien. REST API memiliki konfigurasi yang mudah dipahami dan bersifat open source sehingga dapat lebih cepat untuk mengambil data. Pada REST API terdapat REST *server* yang memberikan jalur untuk mengakses *resource* atau data dan REST *client* yang dapat mengakses *resource* untuk menggunakan atau menampilkan data atau dapat dikatakan pertukaran antar *state* dimana *state* tersebut merupakan halaman *web* yang akan dipanggil oleh *browser* melalui tautan HTTP.

2.3 Rencana Pembangunan Jangka Menengah Daerah

Rencana Pembangunan Jangka Menengah Daerah (RPJMD) merupakan dokumen perencanaan untuk jangka periode selama lima tahun yang berisi penjabaran visi dan misi kepala daerah terpilih dalam tujuan, sasaran, program dan kebijakan pembangunan yang merespon kebutuhan masyarakat dan pembangunan daerah dalam jangka waktu lima tahun kedepan. Tata cara penyusunan rumusan RPJMD terdapat pada *Permendagri Nomor 86 Tahun 2017* yaitu: tahapan awal pelaksanaan persiapan penyusunan meliputi penyusunan rancangan keputusan kepala daerah tentang pembentukan tim penyusun RPJMD, tujuan RPJMD, penyusunan program kerja tim penyusun RPJMD, sasaran perencanaan pembangunan daerah berdasarkan SKPD dan penyusunan rancangan RPJMD (Feradis, 2021).

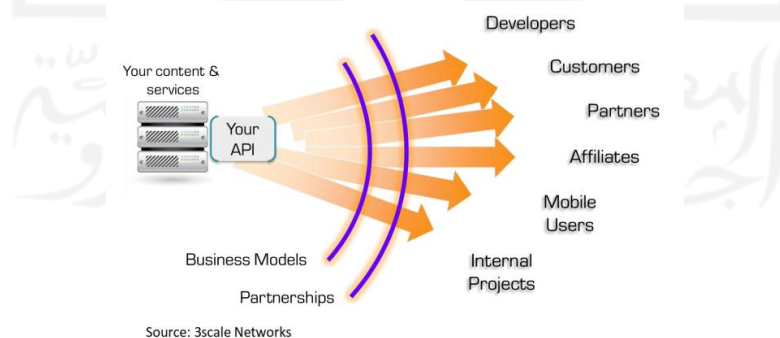
2.4 Rencana Strategis

Recana Strategis (Renstra) adalah suatu dokumen perencanaan suatu organisasi/lembaga yang digunakan untuk menentukan strategi atau arahan dan digunakan sebagai dasar dalam pengambilan keputusan untuk mengalokasi sumber daya selama satu sampai lima tahun sehubungan dengan tugas dan fungsi SKPD. Dokumen Rencana strategis berupa tujuan, sasaran dan strategi, kebijakan, program, dan kegiatan sesuai dengan tugas dan fungsi perangkat daerah berpedoman pada RPJMD serta bersifat indikatif (Bappeda, 2017).

2.5 *Application Programming Interface* (API)

Application Programming Interface (API) adalah Bahasa dan format pesan yang digunakan aplikasi atau program untuk berkomunikasi dengan *Operational System* (OS) atau program kontrol lainnya seperti manajemen database. Implementasi API dengan cara menulis fungsi yang ada dalam program yang menyediakan tautan pada instruksi program yang akan dieksekusi. API sering disebut dengan *web service* karena API memberikan layanan *service* pada sisi *client*. Adapun fungsi pada API yaitu: JSON, XML, JavaScript, SOAP, REST (Pahlevy, 2017).

Application Programming Interface (API) merupakan antarmuka yang digunakan untuk mengakses layanan atau aplikasi dari suatu program. API memungkinkan developer tidak perlu mengulang dari awal pengembangan sistem dengan memanfaatkan fungsi yang sudah ada dari aplikasi lain. **Gambar 2.5** adalah keuntungan menggunakan API yaitu: penggunaan API fleksibel yang artinya dapat digunakan di sistem operasi mana saja, dan bahasa yang digunakan mudah dimengerti seperti JSON (Hasanudin, 2022).



Gambar 2.1 Keuntungan menggunakan API (Sumber: 3scale Networks)

2.6 REST API

Teknologi Representational state transfer (REST) diciptakan oleh Roy Fielding dari University of California (Fielding, 2000). Aplikasi REST sering digunakan untuk pengembangan layanan berbasis web atau mobile karena sangat sederhana dan ringan dibandingkan dengan SOAP. Kinerja, skalabilitas, kesederhanaan, portabilitas, dan kemampuan modifikasi merupakan prinsip utama di balik desain REST.

REST API merupakan sebuah interface yang memungkinkan komunikasi antara dua program perangkat lunak. REST API menggunakan HTTP untuk menentukan *request* dan *response* dan keluaran yang berupa JSON (Deby, 2020). Beberapa method dan tipe request yang tersedia untuk melakukan komunikasi data, antara lain:

1. GET, untuk mendapatkan data dari server.
2. POST, untuk melakukan *create* data
3. UPDATE, untuk memodifikasi data yang sudah ada pada database.
4. DELETE, untuk menghapus data yang ada pada database.

2.7 JavaScript Object Nation (JSON)

JavaScript Object Nation (JSON) adalah format pertukaran data yang mudah diterjemahkan oleh komputer, ringan, sederhana dan ditulis oleh manusia. Format ini merupakan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 – Desember 1999 (Yani, 2017). JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.

Penulis menggunakan metode JSON dalam pengiriman data yang dilakukan, karena JSON memiliki beberapa kelebihan - kelebihan dibandingkan XML, kelebihan – kelebihan tersebut adalah:

- Format Penulisan

Untuk merepresentasikan sebuah struktur data yang rumit dan berbentuk hirarkis penulisan JSON relatif lebih terstruktur dan mudah. Contoh penulisan dalam format JSON dapat dilihat pada **Gambar 2.6**

```
{
  "id": 2,
  "name": "Just as I Am",
  "author": "Cicely Tyson",
  "type": "non-fiction",
  "price": 20.33,
  "current-stock": 0,
  "available": false
}
```

Gambar 2.2 Contoh penulisan dalam format JSON

- Ukuran

Ukuran karakter yang dibutuhkan JSON lebih kecil dibandingkan XML untuk data yang sama. Hal ini tentu berpengaruh pula pada kecepatan pertukaran data, walaupun tidak signifikan untuk data yang kecil, namun cukup berarti jika koneksi yang digunakan relatif lambat untuk mengakses aplikasi web kaya fitur yang memanfaatkan pertukaran data. Di sini JSON lebih unggul dibandingkan XML, kecuali jika data dikompresi terlebih dahulu sebelum dikirimkan, perbedaan JSON dan XML yang telah dikompresi tidaklah signifikan.

- Browser Parsing

Proses parsing merupakan proses pengenalan token atau bagian-bagian kecil dalam rangkaian dokumen XML/JSON. Contohnya, terdapat data text dalam format JSON. Data tersebut harus di-parsing terlebih dahulu sebelum dapat diakses dan dimanipulasi. Browser parsing berarti proses parsing yang terjadi pada sisi client/browser. Melakukan browser parsing pada JSON lebih sederhana dibandingkan pada XML, JSON menggunakan function JavaScript `eval()` untuk melakukan 18 parsing. Sementara dokumen XML di-parsing oleh `XMLHttpRequest`. Rata-rata survei menobatkan JSON sebagai pemenang jika diadu kecepatan parsingnya (Budi, 2017).

2.8 Phalcon Framework

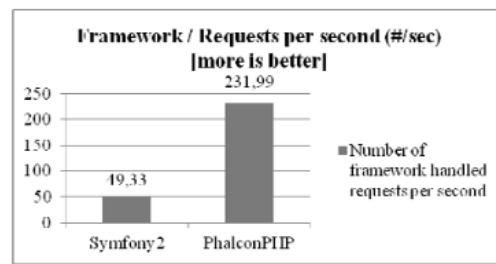
Phalcon telah diresmikan sejak tahun 2012 oleh Andres Gutierrez dan memasuki tahap *stable release* pada tahun 2014. Framework ini memiliki karakteristik sebagai berikut: bersifat open source dan semua komponen ditulis dalam bahasa pemrograman C dan Zephir; memiliki versi berbeda untuk beberapa sistem operasi seperti Linux, Windows dan Mac. *Framework* ini mengimplementasikan Bahasa C menggunakan teknologi Object Relation Mapping (ORM) untuk berinteraksi dengan database (Babakan, 2019). ORM memiliki tugas untuk mengubah data dari Bahasa pemrograman berorientasi objek (OOP) ke lingkup database

relasional serta sebagai penghubung yang memudahkan developer untuk mengembangkan aplikasi yang menggunakan database relational sehingga tugas lebih efisien (Petrosyan, 2021).

Phalcon memiliki sifat *loosely coupled* yang memiliki arti bahwa developer dapat membuat direktori dimanapun. Direktori utama pada aplikasi buatan Phalcon disebut sebagai folder *root*. Folder yang terletak di luar direktori *root* tidak termasuk dalam lingkup kerja Phalcon dan tidak dapat diakses server *web*. Terdapat beberapa fungsi folder pada aplikasi buatan Phalcon antara lain (Ariata, 2018):

1. App: Folder app memuat file inti yang digunakan dalam pengembangan aplikasi berbasis website. Seluruh script folder dan file seperti *config*, *controller*, *library*, *migration*, *models*, dan *view* terletak di dalam folder app.
2. Config: Folder berisi konfigurasi yang akan dialihkan ke dalam subfolder. Beberapa kode dan *library*, *third party*, dan koneksi database dapat ditambahkan ke dalam folder config.
3. Controllers: Folder yang berisi seluruh *request* dari program aplikasi yang dikembangkan. Hasil proses *request* ditampilkan dalam bentuk tampilan program.
4. Library: Folder yang berisi seluruh library eksternal yang digunakan.
5. Migration: Folder yang memuat file akan diedit, dan ditambahkan ketika melakukan migrasi data.
6. Models: Folder yang digunakan untuk konfigurasi pada database.
7. Views: Folder yang berisi tampilan data pengguna. Seluruh informasi terkait view seperti halaman dan konten disimpan pada folder Views.
8. Cache: Folder ini mencakup seluruh data yang berkaitan dengan caching program. Caching berperan penting untuk mempertahankan performa *benchmark*.
9. Public: Folder yang berisi file asset pada pengembangan aplikasi *web* seperti CSS, Javascript, *upload* gambar, meta-data dan data sementara yang digunakan untuk mengembangkan aplikasi.

Tujuan dikembangkannya Phalcon adalah untuk membuat eksistensi ke suatu web server yang meningkatkan kecepatan eksekusi, memastikan proses request lebih tinggi daripada *framework* lainnya serta mengurangi biaya resource. Keunggulan yang dimiliki Phalcon dari *framework* lainnya adalah pemanfaatan memori yang efisien. Menurut pengujian pada **Gambar 2.7**, Phalcon adalah salah satu kerangka kerja PHP tercepat dan memiliki performa kinerja yang tinggi serta biaya sumber daya server yang rendah dibandingkan dengan *framework* Symfony2 (Prokofyeva, 2021).



Gambar 2.3 Perbandingan Framework Phalcon dan Symfony2 (Sumber: Riga, Latvia, 2016)



BAB III

PELAKSANAAN MAGANG

3.1 Aktivitas Magang

Kegiatan magang telah dilakukan selama kurang lebih lima bulan dimulai dari awal bulan September 2021 hingga bulan Januari 2022 sebagai *Backend Developer* di CV. Atsoft Teknologi. Terdapat tiga aktivitas yang telah dilalui selama kurun waktu tersebut. Adapun penjelasan secara detail aktivitas magang pada **Tabel 3.1** tersebut terdapat pada subbab berikutnya.

Tabel 3.1 Aktivitas Magang

No	Aktivitas	Waktu Mulai	Durasi
1	Aktivitas di bulan September	1 September 2021	1 Bulan 7 hari
2	Aktivitas di bulan Oktober, November	8 Oktober 2021	2.5 Bulan
3	Aktivitas di bulan Desember, Januari	22 Desember 2021	2 Bulan

3.1.1 Aktivitas di Bulan September

Pada awal pelaksanaan magang, *Project Manager* menjelaskan lingkungan kerja perusahaan mulai dari jam kerja luring maupun daring, skenario magang, dan beberapa *tools* yang digunakan selama kegiatan magang berlangsung. *Tools* tersebut perlu disiapkan sebelum memasuki tahap pengembangan proyek. Beberapa *tools* yang diperlukan pada kegiatan magang, yaitu:

1. Visual Studio Code sebagai *code editor* yang digunakan selama pengembangan proyek,
2. Memasang *framework* Phalcon,
3. Docker Desktop sebagai alat untuk menjalankan aplikasi dan *container* API,
4. Postman sebagai alat untuk pengujian API,
5. Github untuk melakukan pengecekan dan sebagai tempat penyimpanan *source code*,
6. Discord sebagai alat komunikasi selama kegiatan magang berlangsung.

Project Manager merekomendasikan untuk melakukan pendalaman materi selama kegiatan magang pada salah satu akun youtube yang bernama Owen Seller. Akun tersebut

berguna untuk meningkatkan pemahaman fundamental *framework* Phalcon sehingga dapat memudahkan pelaksanaan pengembangan proyek selama magang berlangsung.

Setelah menyelesaikan pendalaman materi mengenai *framework* Phalcon, *Project Manager* memberikan *link* dokumentasi REST API pada Phalcon *framework*. Dokumentasi tersebut berisi tentang cara mengembangkan REST API menggunakan *framework* Phalcon dan akan terus digunakan bersamaan dengan pengembangan proyek magang hingga waktu kontrak magang berakhir.

3.1.2 Aktivitas di Bulan Oktober dan November

Kegiatan berlanjut dengan memasuki proyek yang bernama Atsoft *E-Government* (ATGOV). ATGOV adalah aplikasi berbasis *website* yang merupakan lanjutan dari SENGGUH (Sistem Evaluasi Pertanggungjawaban Pembangunan Daerah). ATGOV menggunakan REST API untuk mengurangi penggunaan bandwidth dan dapat mempercepat proses pertukaran data. Proyek ini bertujuan untuk membantu BAPEDDA DIY dalam meningkatkan pengelolaan dan pelayanan dalam pembangunan daerah serta mendesain ulang *website* yang mengalami beberapa perubahan utama sebagai upaya untuk meningkatkan kualitas dan memperkuat pengendalian kinerja. Aktivitas magang yang dilakukan dalam proyek ATGOV berada pada posisi *Backend Developer*. Proses pengembangan proyek ATGOV melibatkan peran selain *Backend Developer*, diantaranya: Bapak Andrey Ferriyan sebagai *Project Manager*, Hafidh dan Pradipta sebagai *Frontend Web Developer*, Anwarudin Kurniawan dan Dimas Wahyu sebagai *UI/UX*, dan Bapak Sofyan sebagai *Project Leader*.

Pada proyek ATGOV, mentor memberikan tugas untuk mengembangkan fitur *Create, Read, Update, Delete* (CRUD) pada *master data* ATGOV menggunakan *framework* Phalcon. Pertama kali yang harus dilakukan ketika mengembangkan aplikasi tersebut adalah instalasi proyek. Sebelum melakukan instalasi, folder pada *local* komputer perlu dihubungkan pada aplikasi Github. Dengan terhubungnya folder *local* komputer dengan Github, mentor dapat memonitoring kode yang sudah dibuat oleh pemagang. Setelah melakukan instalasi, pemagang perlu menkonfigurasi Docker untuk menjalankan aplikasi *website* ATGOV dan *container* API. Setelah proses konfigurasi selesai, mentor akan memberikan tugas yang harus dikerjakan dan diserahkan pada jangka waktu satu minggu per fiturnya. Setelah mengerjakan per fitur, mentor dan sebagian karyawan CV. Atsoft Teknologi melakukan pengecekan yang telah dibuat dengan cara menguji fungsi fitur menggunakan aplikasi Postman. Pemagang perlu membuka aplikasi Postman untuk menguji fitur yang dikerjakan menggunakan *endpoint* REST API. Ketika melakukan pengujian, pemagang perlu berkomunikasi melalui aplikasi Discord dengan

Frontend Developer. *Frontend Developer* akan melakukan pengecekan pada tampilan halaman fitur yang dikerjakan melalui *endpoint* REST API yang telah dibuat oleh *Backend Developer*. Dengan adanya REST API, dapat memungkinkan *client* untuk mengambil data pada penyimpanan *server* secara tidak langsung. Dengan komunikasi menggunakan REST API, pemegang atau developer dapat membuat banyak fitur aplikasi, seperti: menambah data, menampilkan data, mengubah data dan menghapus data.

Terdapat 2 tugas utama yang dikerjakan selama proses pengembangan ATGOV, yaitu: pembuatan REST API pada fitur RPJMD dan fitur Renstra serta melakukan pengujian pada REST API yang telah dibuat. Apabila REST API yang dikerjakan berjalan dengan baik, maka mentor akan memberikan tugas selanjutnya. Sebaliknya, pemegang harus merevisi REST API hingga berfungsi dengan baik. Pengendalian dan pemantauan proyek ATGOV akan dijelaskan pada sub bab dibawah ini.

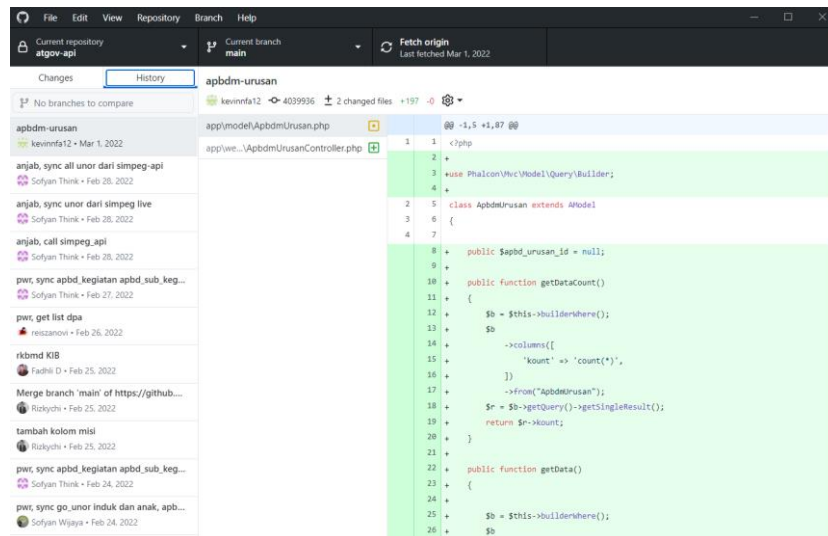
1. Pengendalian dan Pemantauan Proyek Atsoft E-Government (ATGOV)

Selama kegiatan magang berlangsung, terdapat dua teknologi yang digunakan untuk pengendalian dan Pemantauan proyek ATGOV yaitu: Github dan Discord.

a. Github

Github merupakan platform web dan layanan berbasis *cloud* untuk mengelola dan menyimpan kode developer. Github dapat memudahkan developer untuk memonitor dan mendokumentasikan perubahan kode yang dibutuhkan sesuai dengan kebutuhan selama pengembangan proyek. Platform ini merupakan salah satu bagian dari *storehouse online* terbesar di dunia. Dengan platform ini, semua orang pada berbagai tempat dapat bekerja sama untuk mengembangkan proyek bersama-sama (Faradila A, 2021). Github digunakan pada proyek ATGOV untuk menyimpan dan berbagi *source code* sebagai arsip perusahaan CV. Atsoft Teknologi. Tampilan aplikasi *desktop* github pada proyek ATGOV dapat dilihat pada

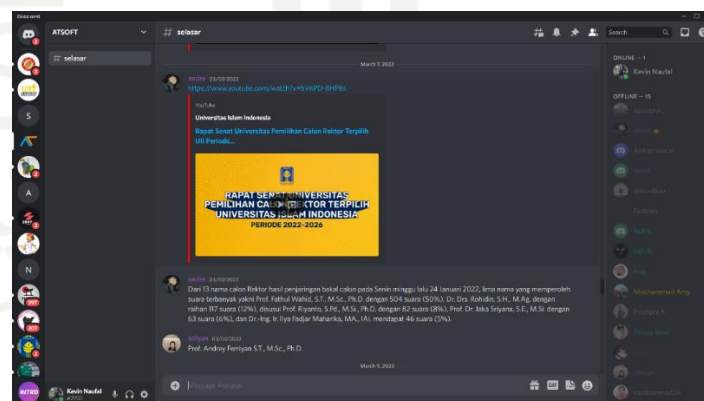
Gambar 3.1



Gambar 3.1 Tampilan aplikasi desktop Github

b. Discord

Discord adalah platform yang didesain untuk menciptakan suatu komunitas karena dapat digunakan penggunaanya untuk melakukan aktivitas sosial. Pada platform ini, pengguna berkomunikasi melalui pesan suara, video, pesan teks, serta pengiriman file. Discord digunakan untuk media komunikasi dengan seluruh karyawan di CV. Atsoft Teknologi. Discord juga digunakan sebagai media untuk memberikan tugas oleh *Project manager* dan pengumpulan tugas yang telah dikerjakan oleh pemegang selama kegiatan magang dengan CV. Atsoft Teknologi. Tampilan aplikasi Discord dapat dilihat pada **Gambar 3.2**.



Gambar 3.2 Tampilan aplikasi Discord

3.1.3 Aktivitas di Bulan Desember, Januari

Aktivitas dimulai dengan melakukan perbaikan dan melanjutkan pengembangan pada sub fitur yang telah dikerjakan. Pada Tabel 3.2 mencantumkan seluruh pekerjaan yang dilakukan selama kurang lebih 7 minggu kegiatan magang dengan CV. Atsoft Teknologi

Tabel 3.2 Aktivitas bulan Desember, dan Januari

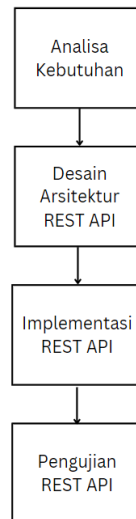
No	Pekan	Pekerjaan	Durasi
1	1	Pengembangan API sub fitur RPJMD Periode beserta tampilannya	6 hari
2	2-3	Pengujian sub fitur RPJMD Periode	3 hari
3		Pengembangan API pada sub fitur RPJMD Program	6 hari
	3	Revisi sub fitur RPJMD program	4 hari
4		Pengujian sub fitur RPJMD program	3 hari
5	4	Pengembangan API pada sub fitur RPJMD sasaran	6 hari
6		Pengujian sub fitur RPJMD sasaran	3 hari
7	5-6	Pengembangan API pada sub fitur RPJMD tujuan	6 hari
8		Pengujian sub fitur RPJMD tujuan	3 hari
9	6	Pengembangan API pada sub fitur Renstra periode	6 hari
10	7	Pengujian sub fitur Renstra periode	3 hari
11		Pengembangan API pada sub fitur Renstra program	6 hari
12	8	Pengujian sub fitur Renstra program	3 hari
13		Dokumentasi fungsi proyek melalui Postman	3 hari

Aktivitas telah memasuki tahap akhir proyek, namun masih terdapat fitur yang belum selesai dikerjakan. Oleh karena itu, pengembangan Sistem Informasi Atsoft *E-Government* (ATGOV) masih berlanjut. Masih terdapat fitur yang belum dikembangkan seperti fitur Manajemen jabatan, Evaluasi Jabatan, Manajemen kegiatan, dan Manajemen Program. Hal ini terjadi karena pemangag hanya terlibat pada pengembangan ATGOV dan berkontribusi sampai kontrak magang berakhir, sehingga proses pengembangan ATGOV tidak dilanjutkan pemangag sementara waktu. Tidak menutup kemungkinan bahwa pengembangan ATGOV dilanjutkan oleh pemangag selanjutnya.

3.2 Pengembangan REST API pada Sistem Informasi Atsoft *E-Government*

Pengembangan REST API pada Sistem Informasi Atsoft *E-Government* (ATGOV) bertujuan untuk memisahkan dan menyederhanakan interaksi antara *client* dan *server*.

Penyederhanaan interaksi antara *client* dan *server* membuat perubahan yang terjadi pada *server* tidak mempengaruhi *client*. Sehingga, kinerja *client* dan *server* menjadi lebih efisien karena kedua sisi dapat mengembangkan fitur tanpa mempengaruhi proses pertukaran data. Terdapat beberapa tahap pengembangan REST API yang dapat dilihat pada **Gambar 3.3**



Gambar 3.3 Tahap pengembangan REST API

Gambar tersebut merupakan bagan dari tahapan yang dilakukan pada pengembangan REST API. Adapun hasil dari setiap tahapan pengembangan REST API yang akan dijelaskan pada sub bab di bawah ini.

3.2.1 Analisa Kebutuhan

Tahapan pertama dalam proses pengembangan REST API adalah diskusi dengan tim yang terlibat dalam pengerjaan proyek. Diskusi dilakukan dengan *Project Manager* (PM) untuk menentukan analisa data dan identifikasi masalah. Analisa yang dilakukan bertujuan untuk menetapkan solusi desain arsitektur REST API.

1. Analisa data dan Identifikasi masalah

Proses analisis data dan informasi adalah proses untuk menentukan karakteristik dari pengguna ATGOV. Proses ini bertujuan untuk mengidentifikasi masalah dari pengguna dan organisasi kemudian dapat menghasilkan solusi desain REST API. Pada proses ini, penulis melakukan proses wawancara dengan *Project Manager* (PM) untuk mendapatkan data profil pengguna dan mengidentifikasi masalah pada sistem. Proses wawancara dirangkum dalam sebuah tabel yang dapat dilihat pada **Tabel 3.3**

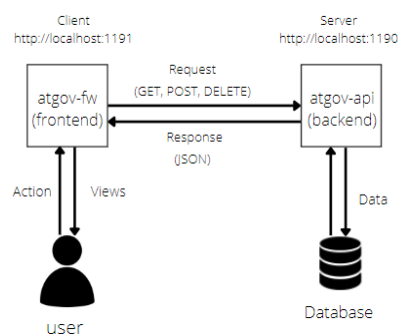
Tabel 3.3 Proses wawancara untuk pengumpulan data

No	Pertanyaan	Hasil
1	Siapa yang menggunakan Sistem Informasi <i>Atsoft E-Government</i> ?	Pengguna utama dari Sistem Informasi <i>Atsoft E-Government</i> adalah BAPEDDA DIY
2	Apa permasalahan dari <i>Atsoft E-Government</i> ?	Permasalahan dari sistem informasi <i>Atsoft E-Government</i> adalah belum memiliki <i>web service</i> atau <i>Application Program Interface</i> (API). Sehingga perlu dikembangkannya API berteknologi REST yang dapat memisahkan dan menyederhanakan interaksi antara <i>client</i> dan <i>server</i> .
3	REST API seperti apa yang diharapkan?	REST API yang dapat mempermudah sisi <i>client</i> untuk mengambil data ke <i>database server</i> . Sehingga, kinerja <i>client</i> dan <i>server</i> menjadi lebih efisien karena kedua sisi dapat mengembangkan modul tanpa mempengaruhi proses pertukaran data

3.2.2 Desain Arsitektur REST API

Pada tahap ini, akan dijelaskan hasil desain arsitektur REST API. Bahasa pemrograman yang digunakan dalam mendesain REST API adalah PHP dan *framework* Phalcon. Setelah desain REST API telah dibuat, dilanjutkan oleh *Frontend developer* untuk membuat tampilan antarmuka. REST API juga berguna untuk pertukaran data antara *client* dan *server* pada ATGOV seperti yang dijelaskan pada **Gambar 3.4**. Agar sisi *client* dan sisi *server* dapat saling bertukar data, maka diperlukan API yang dapat menghubungkan keduanya. Berikut *method* yang digunakan antara lain:

1. GET, method yang digunakan untuk membaca data dari *database*
2. POST, method yang digunakan untuk mengirim data ke *database*
3. DELETE, method yang digunakan untuk menghapus data dari *database*



Gambar 3.4 Arsitektur REST API

Berdasarkan arsitektur pada **Gambar 3.4**, atgov-fw berfungsi sebagai *frontend* yang menggunakan pemrograman JavaScript, atgov-api berfungsi sebagai *backend* yang menggunakan pemrograman PHP dengan *framework* Phalcon. HTTP Request dilakukan pada sisi *client* dengan memanfaatkan *method* GET, POST, DELETE pada *server* yang menyimpan *resource* melalui *endpoint*. Selanjutnya, REST server memberikan *feedback* berupa HTTP response seperti permintaan dari sisi *client*. Server membalas response kepada sisi *client* dengan format JSON. Output yang berformat JSON akan diganti sehingga sisi *client* dapat membaca format dengan jelas. Dengan adanya proses HTTP Request dan proses HTTP Response, *client* dan *server* menjadi dua proses yang terpisah, sehingga kinerja *server* menjadi lebih efisien karena kedua sisi dapat mengembangkan modul tanpa mempengaruhi proses pertukaran data

3.2.3 Implementasi REST API

Pada tahapan implementasi REST API, atgov-api adalah sisi *server* (REST server) dan atgov-fw adalah sisi *client* (REST client) sesuai dengan yang telah dijelaskan pada **Gambar 3.4**. E-Government beroperasi menggunakan Create, Read, Update, Delete (CRUD) dengan fitur tambahan lainnya seperti *limit show*, pencarian, dan *pagination*. Dengan menggunakan REST API pertukaran data antar *client* dan *server* dapat menjadi lebih mudah. Adapun tiga tahapan dalam melakukan implementasi REST API yaitu: persiapan, REST Server dan REST Client. Tahapan tersebut akan dijelaskan pada sub bab dibawah ini.

1. Persiapan

Sebelum memasuki tahap pengembangan REST *client* dan REST *server*, developer perlu mempersiapkan Docker desktop sebagai container API. Penggunaan Docker berguna untuk mengurangi pemakaian memori, prosesor dan storage pada komputer karena Docker tidak membangun mesin virtualnya sendiri.

a. Konfigurasi Docker pada Proyek ATGOV

Pertama, developer perlu menyiapkan image pada docker. Image adalah *executable package* yang terdiri dari app, library, runtime, environment variable, dan file konfigurasi. Proyek ATGOV menggunakan image Nginx yang bisa didapatkan pada <https://hub.docker.com/r/sabrecode/nginxphp7phalcon4>. CV. Atsoft Teknologi menggunakan Nginx karena stabil, memiliki performa yang tinggi, hemat sumber daya, dan memiliki komunitas yang luas sehingga mudah untuk melakukan *troubleshoot* secara mandiri.

Kedua, developer perlu menyiapkan container pada Docker. Container adalah sebuah *environment* yang digunakan untuk menyimpan dan mengemas aplikasi seperti *system tools*,

code, library, runtime, dan file konfigurasi dan hanya bisa mengakses sumber daya yang telah ditentukan pada Docker Image. Pada Container, terdapat docker compose yang berguna untuk mempermudah developer dalam proses pembuatan container. **Gambar 3.5** merupakan konfigurasi docker-compose.yml pada proyek ATGOV.

```
version: "3.7"
services:
  web:
    container_name: atgov-api
    #image: nimdasx/sf-phalcon4
    #image: sabrecode/nginxphp7phalcon4
    #image: nimdasx/sf-nginx-php74
    image: ghcr.io/nimdasx/docker-nginx-php7-phalcon4:latest
    restart: unless-stopped
    ports:
      - 1190:80
    volumes:
      #apache
      #- ../var/www/html
      #nginx
      - ./nginx/10-location-
root.conf:/opt/docker/etc/nginx/vhost.common.d/10-location-root.conf
      - ./nginx/vhost.conf:/opt/docker/etc/nginx/vhost.conf
      - ./php/php-nimdasx.ini:/usr/local/etc/php/conf.d/php-nimdasx.ini
      - ../app:/app
```

Gambar 3.5 Konfigurasi docker-compose.yml pada proyek ATGOV

Dari gambar tersebut menunjukkan konfigurasi docker-compose.yml pada proyek ATGOV sudah berhasil. Pada proyek ATGOV container diberi nama atgov-api dengan menggunakan image Nginx yang sudah ditentukan sebelumnya serta menggunakan port 1190:80.

2. REST Server

Pada arsitektur REST API terdapat beragam *method* yang digunakan pada protokol HTTP, namun pada proyek ini hanya memerlukan tiga *method* yaitu GET, POST, dan DELETE. Pada **Tabel 3.4** merupakan salah satu daftar *endpoint* REST API pada proyek ATGOV yang sedang dikembangkan

Tabel 3.4 Salah satu daftar endpoint REST API pada proyek ATGOV

Endpoint	Method
/renstra_program/get	GET
/renstra_program/get_list	GET
/renstra_program/insert	POST
/renstra_program/update	POST
/renstra_program/delete	DELETE

Selanjutnya, developer perlu melakukan konfigurasi database pada file *konfigurasi.php* seperti **Gambar 3.6** untuk melakukan akses pada database ATGOV yaitu MySQL.

```
<?php
return new Phalcon\Config([
    'db' => [
        'adapter' => 'Mysql',
        'host' => 'dev.atsoft.co.id',
        'username' => 'devatsoft_kevin',
        'password' => 'lgDkJU5fAq9Yxk*****',
        'dbname' => 'devatsoft_atgov_anjab',
        'charset' => 'utf8',
        'port' => '1190',
    ],
    'app_id' => 'atgov-api',
    'version' => '0.1',
    'printNewLine' => true,
    'display_errors' => true,
    'debug' => true,
    'peran_id publik' => 'publik',
    'username publik' => 'publik',
]);
```

Gambar 3.6 Konfigurasi database ATGOV.

Berdasarkan potongan kode pada **Gambar 3.6**, baris 3 – 11 merupakan logika pemrograman untuk deklarasi database ATGOV. *adapter* merupakan jenis database yang digunakan pada proyek ATGOV yaitu MySQL, *Host* merupakan penyimpanan yang digunakan yaitu *dev.atsoft.co.id*. *username* dan *password* digunakan untuk login kedalam *database* proyek ATGOV yang bernama *devatsoft_atgov_anjab*. Nomor *port* pada proyek ATGOV adalah 1190.

Selanjutnya, perlu membuat *controller* untuk menentukan proses pengiriman data agar data dapat diakses oleh pengguna. *Controller* terdiri dari seluruh *endpoint* REST API, sehingga pertukaran data menjadi lebih efektif. **Gambar 3.7** adalah salah satu contoh implementasi dari *endpoint /renstra_program/get_list* pada *controller* menggunakan method GET.

```
public function getlistAction()
{
    $p = $this->request->getJsonRawBody();

    //njikuk data
    $hs = new RenstraProgram();
    $hs->setParams($p);
    //filter data
    $hs->go_unor_id_opd = $p->go_unor_id_opd;
    $hs->renstra_periode_id = $p->renstra_periode_id;
    $data = $hs->getData();

    //ngitung jumlah record terfilter
    $filtered_record = $hs->getDataCount();

    //ngitung jumlah record total
    $stot = new RenstraProgram();
    $total_record = $stot->getDataCount();

    $other_data = new StdClass();
    $other_data->total_record = $total_record;
    $other_data->filtered_record = $filtered_record;

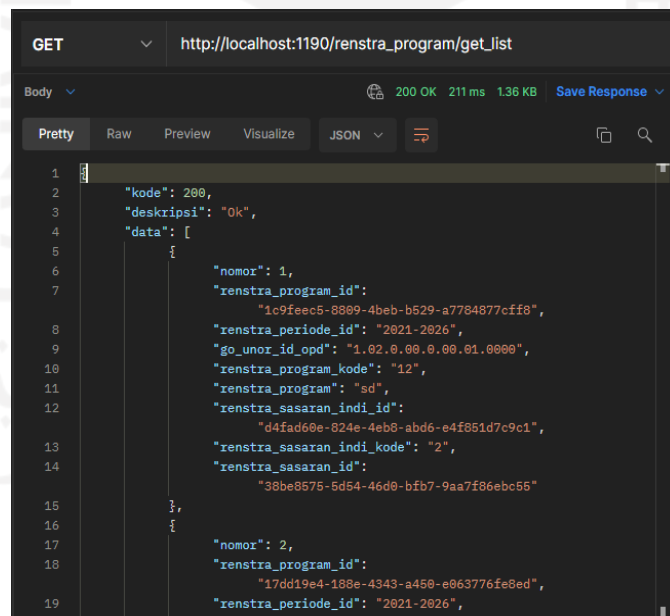
    return $this->setJsonRespon(null, null, $data, $other_data);
}
```

Gambar 3.7 Contoh penggunaan method GET pada *controller*

Gambar 3.7, penggunaan method GET berfungsi untuk menampilkan semua data rencana strategis program dengan cara mengakses alamat sesuai *endpoint* yang dibuat. Pada contoh ini, Alamat yang digunakan adalah http://localhost:1190/renstra_program/get_list. Bagian pertama merupakan nama domain. Proses implementasi masih menggunakan database lokal sehingga dituliskan “localhost”. Penomoran *port* dari *server* dituliskan dengan angka “1190”. Selanjutnya merupakan direktori sebagai penanda bahwa layanan API telah dapat diakses. Pada bagian akhir, merupakan data yang terdapat dalam *controller*. **Tabel 3.5** merupakan *controller* untuk mengelola renstra program. Pada **Gambar 3.8**, merupakan hasil eksekusi *url* diatas menggunakan Postman. *Output* yang didapatkan berupa format JSON.

Tabel 3.5 Controller untuk mengelola Renstra program

Nama Controller	Fungsionalitas	Request Method	Fungsi
RenstraProgramController	Memuat method yang berhubungan dengan pengelolaan renstra program pada aplikasi ATGOV	GET	getListAction()
		GET	getAction()
		POST	insertAction()
		POST	updateAction()
		DELETE	deleteAction()



Gambar 3.8 Output Data Dengan Format JSON pada Postman

Penulisan output data dengan format JSON seperti pada **Gambar 3.8**, memudahkan sisi *client* untuk membaca data karena memiliki struktur kode yang sederhana dibandingkan dengan format XML.

Selanjutnya, pada Phalcon tabel database direpresentasikan menggunakan model. Salah satu tabel yang dibuat pada proyek ATGOV adalah tabel renstra program. **Gambar 3.9** merupakan contoh model pada tabel Renstra program.

```
public function getData()
{
    $b = $this->builderWhere();
    $b
        ->columns([
            'RenstraProgram.renstra_program_id',
            'RenstraProgram.renstra_periode_id',
            'RenstraProgram.go_unor_id_opd',
            'RenstraProgram.renstra_program_kode',
            'RenstraProgram.renstra_program',
            'RenstraSasaranIndi.renstra_sasaran_indi_id',
            'RenstraSasaranIndi.renstra_sasaran_indi_kode'
        ])
        ->from("RenstraProgram")
        ->leftJoin("GoUnor",
            "GoUnor.go_unor_id=RenstraProgram.go_unor_id_opd")
        ->leftJoin("RenstraPeriode",
            "RenstraPeriode.renstra_periode_id=RenstraProgram.renstra_periode_id")
        ->leftJoin("RenstraSasaranIndi",
            "RenstraSasaranIndi.renstra_sasaran_indi_id=RenstraProgram.renstra_sasaran_indi_id");
}
```

Gambar 3.9 Model pada renstra program

Seperti yang dilihat pada **Gambar 3.9**, dapat diuraikan beberapa penjelasan sebagai berikut:

- Baris pertama merupakan deklarasi fungsi `getData` yang berguna untuk mengambil data dari database dan ditampilkan pada `atgov-api`
- Baris 4, terdapat fungsi `builderWhere` untuk membangun *query* SQL tanpa harus membuat pernyataan SQL menggunakan Phalcon
- Baris 6 – 16, adalah pernyataan kolom yang dipanggil dari tabel Renstra program.
- Baris 17, adalah pernyataan tabel Renstra program LEFT JOIN dengan tabel GoUnor. Pernyataan ini, digunakan untuk menampilkan `go_unor_id_opd` yang ada pada tabel GoUnor
- Baris 19, adalah pernyataan tabel Renstra Program LEFT JOIN dengan tabel RenstraPeriode. Pernyataan ini, digunakan untuk menampilkan `renstra_periode_id` pada tabel RenstraPeriode
- Baris 21, adalah pernyataan tabel Renstra Program LEFT JOIN dengan tabel RenstraSasaranIndi. Pernyataan ini, digunakan untuk menampilkan `renstra_sasaran_indi` pada tabel RenstraSasaranIndi.

3. REST Client

Pada sisi *client* diimplementasikan dengan atgov-fw menggunakan Bahasa pemrograman PHP dan JavaScript dengan *framework* Phalcon sebagai salah satu *framework* berbasis *website*.

Gambar 3.10. Menampilkan salah satu *source code* yaitu renstra program pada saat *client* melakukan HTTP *request* untuk memanggil data yang telah disimpan di dalam *database* menggunakan *method* GET.

```
function getListAction() {
    $order_arr = [
        'RenstraProgram.renstra_program_kode',
        'RenstraProgram.renstra_program_kode',
        'RenstraSasaranIndi.renstra_sasaran_indi_kode',
        'RenstraProgram.renstra_program'
    ];
    $api_table = 'renstra_program/get_list';
    $data['go_unor_id_opd'] = $this->session->get("go_unor_id");
    $data['renstra_periode_id'] = $this->session->get("renstra_periode_id");
    return $this->getFormatDatatables($order_arr, $api_table, $data);
}
```

Gambar 3.10 Kode *request* dengan *method* GET pada renstra program

Pada **Gambar 3.10** diketahui bahwa baris 1 terdapat fungsi `getListAction` yang digunakan untuk memanggil data renstra program menggunakan *method* GET. Selanjutnya, baris 2-7 terdapat variabel `order_arr` berbentuk array yang berfungsi untuk menampung data yang ada pada tabel Renstra program. Selanjutnya, pada baris 8 terdapat variabel `api_table` pada variabel tersebut atgov-fw akan menarik data pada API untuk meminimalisir akses langsung pada database. Selanjutnya, baris 9-10 variabel data berfungsi untuk memanggil data `go_unor_opd_id` dan `renstra_periode_id` yang akan ditampilkan pada sisi *client*. Selanjutnya pada baris 11 terdapat fungsi `getFormatData` yang berguna menyimpan dan menampilkan *data table* renstra program yang ada pada sisi *server*.

3.2.4 Pengujian REST API

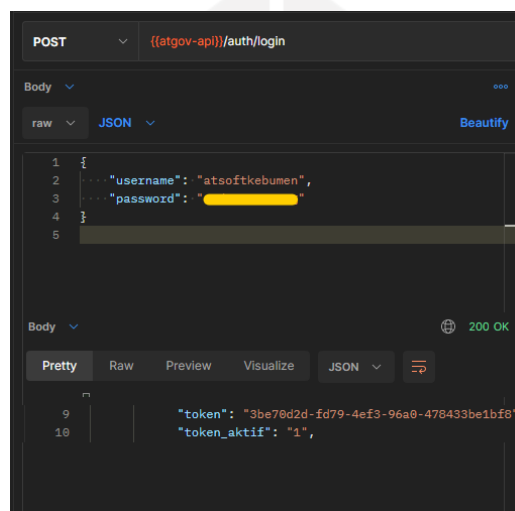
Pengujian REST API dilakukan dengan menguji fungsionalitas API pada enam sub fitur. Pengujian dilakukan untuk membuktikan kelayakan REST API yang telah dibuat sehingga dapat digunakan secara optimal untuk fitur yang ada pada ATGOV. Hasil *Response* yang diharapkan adalah saat pengujian memiliki *status* kode 200 jika berhasil, 404 jika data tidak ditemukan, dan 500 apabila *server error*. Dikarenakan pola pengujian yang mirip, hanya dua sub fitur saja yang akan dilaporkan pada laporan tugas akhir ini. Kedua sub fitur yang akan dilaporkan meliputi sub fitur RPJMD periode dan sub fitur Renstra program. Penjelasan secara jelas tentang pengujian kedua sub fitur tersebut akan dijelaskan pada sub bab dibawah ini. Sedangkan, penjelasan tentang pengujian sub fitur lainnya dapat dilihat pada lampiran.

1. Pengujian Rencana Pembangunan Jangka Menengah Daerah (RPJMD) Periode

Pengujian dilakukan menggunakan Postman untuk memastikan kelayakan REST API pada sub fitur RPJMD periode dengan menyusun *method* yang akan diuji serta memasukkan *endpoint* sebagai berikut:

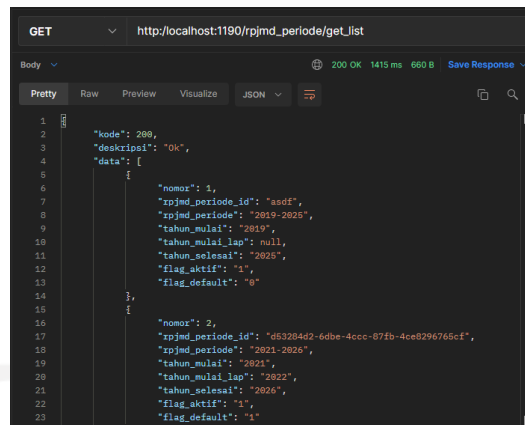
a. Response membaca seluruh data Rencana Pembangunan Jangka Menengah Daerah (RPJMD) Periode

Endpoint `rpjmd_periode/get_list` digunakan untuk mengambil seluruh data RPJMD Periode yang terdapat pada server. Pengujian ini dilakukan dengan mengirimkan request data ke *endpoint* tersebut dengan menggunakan *method* GET dan memasukkan *url* http://localhost:1191/rpjmd_periode/get_list. Sebelum melakukan pengujian, developer perlu melakukan *authorize* pada auth/login yang dapat dilihat pada **Gambar 3.11** untuk mendapatkan kode *token* yang akan terus digunakan selama pengujian dilakukan.



Gambar 3.11 Tampilan login untuk mendapatkan token pengujian

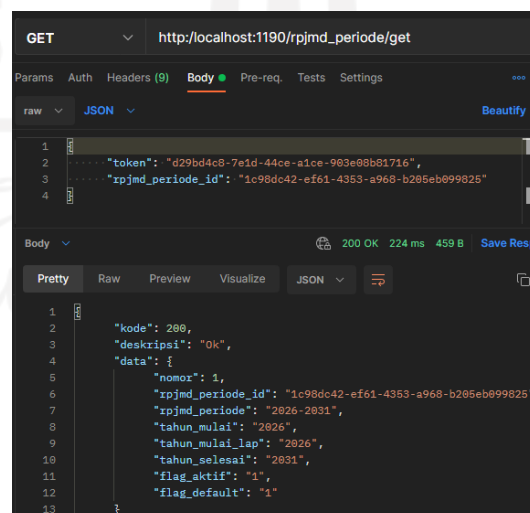
Selanjutnya, untuk membaca seluruh data periode, perlu mengisi *token* pada bagian *body text*. Kemudian server memberikan response yang dapat dilihat pada **Gambar 3.12**. Gambar tersebut menunjukkan bahwa server berhasil memberikan response dengan waktu yang singkat sekitar 1415 ms Untuk menampilkan seluruh data RPJMD periode.



Gambar 3.12 *Response* membaca data seluruh user berhasil

b. *Response* membaca data berdasarkan *id* Rencana Pembangunan Jangka Menengah Daerah (RPJMD) periode

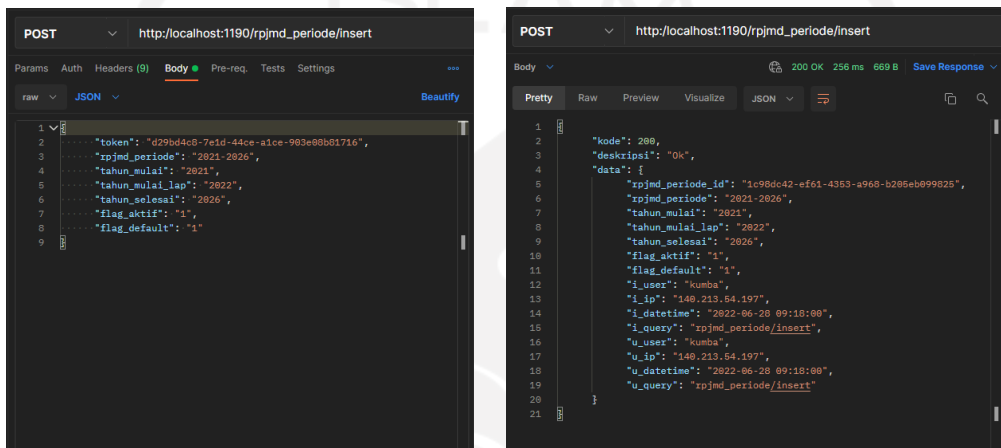
Endpoint *rpjmd_periode/get* digunakan untuk mengambil satu data RPJMD Periode berdasarkan *id* periode yang terdapat pada server. Pengujian ini dilakukan dengan mengirimkan request data ke *endpoint* tersebut dengan menggunakan *method* GET dan memasukkan *url* `http://localhost:1191/rpjmd_periode/get`. Untuk membaca data periode berdasarkan *id*, perlu memilih menu pada bagian 'Body' untuk membuat request ke *body*, lalu pilih *header* yang digunakan yaitu 'raw'. Selanjutnya mengisi *id* pada bagian *body text*. Kemudian server memberikan response yang dapat dilihat pada **Gambar 3.13**. Gambar tersebut menunjukkan bahwa server berhasil memberikan *response* berupa satu *id* pada database RPJMD periode.



Gambar 3.13 *Response* membaca data berdasarkan *id* periode berhasil

c. *Response* tambah data Rencana Pembangunan Jangka Menengah Daerah (RPJMD) periode

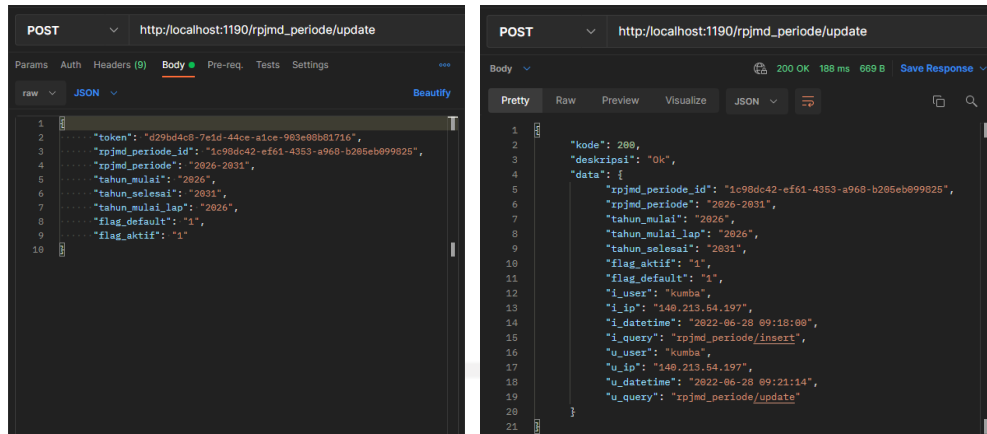
Endpoint *rpjmd_periode/insert* digunakan untuk menambahkan data RPJMD Periode. Pengujian ini dilakukan dengan mengirimkan *request* data ke *endpoint* tersebut dengan menggunakan *method* POST dan memasukkan *url* `http://localhost:1191/rpjmd_periode/insert`. Untuk menambahkan data periode, perlu memilih menu pada bagian ‘Body’ untuk membuat request ke *body*, lalu pilih *header* yang digunakan yaitu ‘raw’. Selanjutnya mengisi *rpjmd_periode*, *tahun_mulai*, *tahun_mulai_lap*, *tahun_selesai*, *flag_aktif*, dan *flag_default* pada bagian *body text* seperti pada **Gambar 3.14**. Kemudian developer perlu mengirim request dengan menekan tombol ‘Send’. Apabila proses berhasil, server akan memberikan *response*.



Gambar 3.14 Langkah membuat RPJMD periode baru (Kiri) dan Response menambah data RPJMD periode berhasil (Kanan)

d. **Response ubah data Rencana Pembangunan Jangka Menengah Daerah (RPJMD) periode**

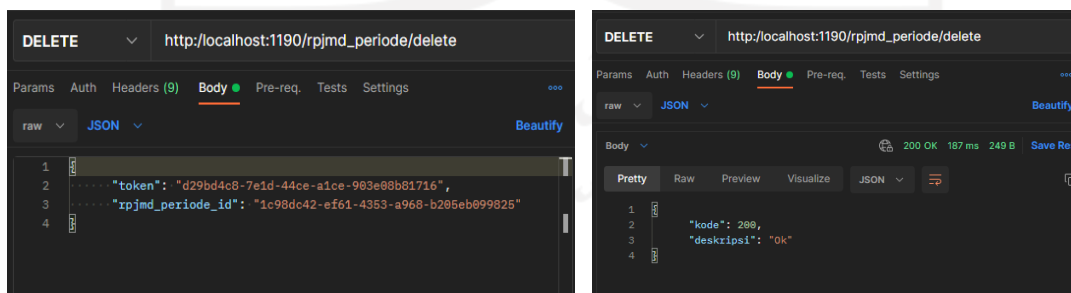
Endpoint *rpjmd_periode/update* digunakan untuk mengubah data RPJMD Periode. Pengujian ini dilakukan dengan mengirimkan *request* data ke *endpoint* tersebut dengan menggunakan *method* POST dan memasukkan *url* `http://localhost:1191/rpjmd_periode/update`. Untuk mengubah data periode, perlu memilih menu pada bagian ‘Body’ untuk membuat request ke *body*, lalu pilih *header* yang digunakan yaitu ‘raw’. Selanjutnya isi *rpjmd_periode_id* pada bagian *body text* untuk mengubah data sesuai dengan *id* data RPJMD periode. Pada **Gambar 3.15** dilakukan update pada *tahun_mulai*, dan *tahun_selesai*. Selanjutnya, Developer perlu mengirim *request* dengan menekan tombol ‘Send’. Apabila proses berhasil, server akan memberikan *response*.



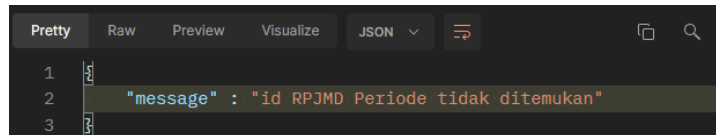
Gambar 3.15 Langkah mengubah data RPJMD periode (Kiri) dan *Response* edit data RPJMD periode berhasil (Kanan)

e. *Response* hapus data Rencana Pembangunan Jangka Menengah Daerah (RPJMD) periode

Endpoint `Url/rpjmd_periode/delete` digunakan untuk menghapus data RPJMD Periode. Pengujian ini dilakukan dengan mengirimkan *request* data ke *endpoint* tersebut dengan menggunakan *method* DELETE dan memasukkan *url* `http://localhost:1191/rpjmd_periode/delete`. Untuk menghapus data periode, perlu memilih menu pada bagian ‘*Body*’ untuk membuat request ke *body*, lalu pilih *header* yang digunakan yaitu ‘*raw*’. Selanjutnya isi *rpjmd_periode_id* pada bagian *body text* untuk menghapus data sesuai dengan *id* data RPJMD periode seperti pada **Gambar 3.16**. Selanjutnya, Developer perlu mengirim *request* dengan menekan tombol ‘*Send*’. Apabila proses berhasil, *server* akan memberikan *response* dan **Gambar 3.17** menampilkan *response* ketika *id* RPJMD periode tidak ditemukan.



Gambar 3.16 Langkah menghapus data sesuai *id* RPJMD periode (Kiri), *Response* hapus data sesuai *id* RPJMD berhasil (Kanan)



Gambar 3.17 *Response* ketika data hapus data RPJMD periode tidak ditemukan

Berdasarkan pada pengujian yang telah dilakukan, REST API sudah berjalan sesuai yang telah diharapkan. Untuk lebih jelasnya, dapat dilihat pada **Tabel 3.6** yang membuktikan seluruh daftar REST API berhasil diuji. Pengujian dilakukan terhadap seluruh endpoint RPJMD periode dengan *blackbox testing*.

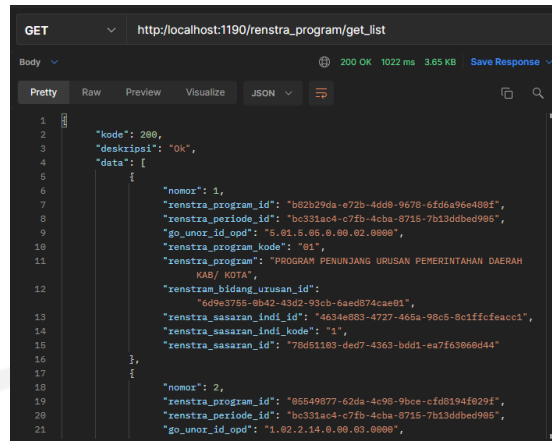
Tabel 3.6 Daftar *endpoint* REST API RPJMD Periode

Endpoint	Fungsionalitas yang diuji	Hasil yang diharapkan	Hasil pengujian
/rpjmd_periode/get_list	Menampilkan seluruh data RPJMD periode	Seluruh data RPJMD periode ditampilkan	Sukses
/rpjmd_periode/get	Menampilkan data RPJMD periode berdasarkan <i>id</i>	Data RPJMD berdasarkan <i>id</i> ditampilkan	Sukses
/rpjmd_periode/insert	Menambah data RPJMD periode	Data RPJMD periode ditambahkan	Sukses
/rpjmd_periode/update	Mengubah data RPJMD periode	Data RPJMD periode diubah	Sukses
/rpjmd_periode/delete	Menghapus data RPJMD periode	Data RPJMD periode dihapus	Sukses

2. Pengujian Rencana Strategis (Renstra) program

a. Response menampilkan seluruh data Rencana Strategis (Renstra) program

Endpoint `/renstra_program/get_list` digunakan untuk mengambil seluruh data Renstra program yang terdapat pada server. Pengujian ini dilakukan dengan mengirimkan *request* data ke *endpoint* tersebut dengan menggunakan *method* GET dan memasukkan *url* http://localhost:1191/renstra_program/get_list. Sebelum melakukan pengujian, developer perlu melakukan *authorize* pada *auth/login* untuk mendapatkan kode *token* yang akan terus digunakan selama pengujian dilakukan. Selanjutnya, untuk membaca seluruh data Renstra program, perlu mengisi *token* pada bagian *body text*. Kemudian server memberikan *response* yang dapat dilihat pada **Gambar 3.18**.

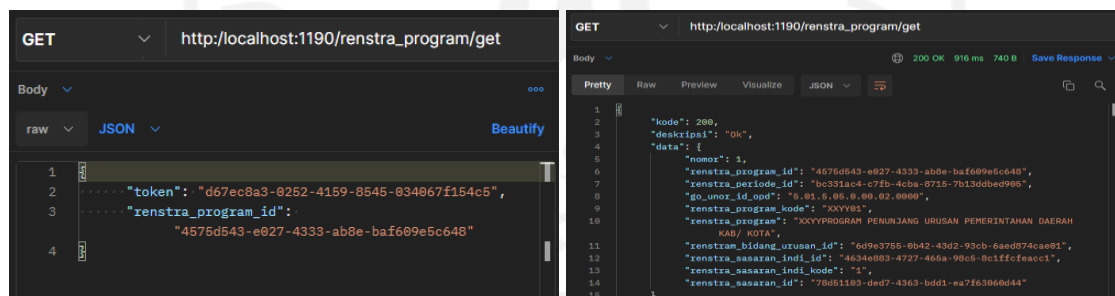


Gambar 3.18 Response menampilkan seluruh data Renstra program

Gambar tersebut menunjukkan bahwa server berhasil memberikan *response* dengan waktu yang singkat sekitar 1022 ms untuk menampilkan seluruh data Renstra program.

b. Response menampilkan data berdasarkan *id* Rencana Strategis (Renstra) program

Endpoint `/renstra_program/get` digunakan untuk mengambil satu data Renstra program berdasarkan *id* Renstra program yang terdapat pada server. Pengujian ini dilakukan dengan mengirimkan *request* data ke *endpoint* tersebut dengan menggunakan *method* GET dan memasukkan *url* `http://localhost:1191/renstra_program/get`. Untuk membaca data Renstra program berdasarkan *id*, perlu memilih menu pada bagian 'Body' untuk membuat *request* ke *body*, lalu pilih *header* yang digunakan yaitu 'raw'. Selanjutnya mengisi *id* Renstra program pada bagian *body text* seperti pada **Gambar 3.19**. Kemudian bila berhasil, server memberikan *response* berupa satu *id* Renstra program pada *database* Renstra program.

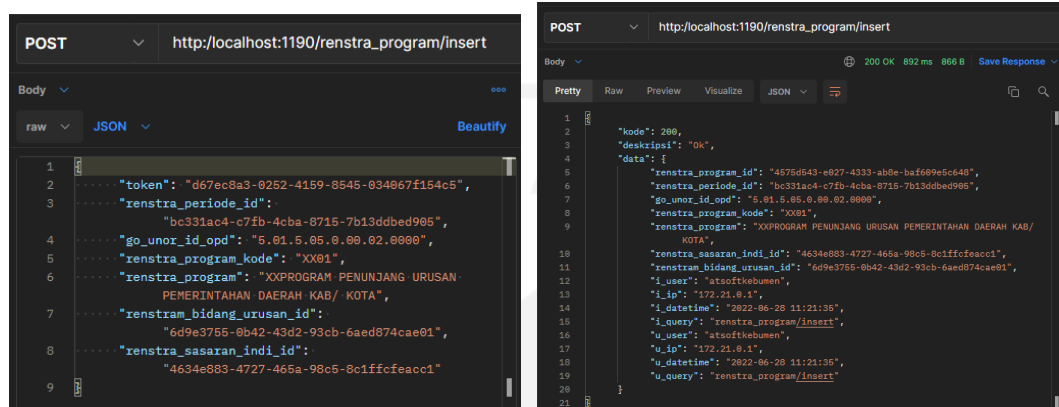


Gambar 3.19 Langkah mengambil data berdasarkan *id* Renstra program (Kiri) dan Response mengambil data berdasarkan *id* Renstra program (Kanan)

c. Response tambah data Rencana Strategis (Renstra) program

Endpoint `/renstra_program/insert` digunakan untuk menambahkan data Renstra program. Pengujian ini dilakukan dengan mengirimkan *request* data ke *endpoint* tersebut dengan menggunakan *method* POST dan memasukkan *url* `http://localhost:1191/renstra_program/insert`. Untuk menambahkan data Renstra program,

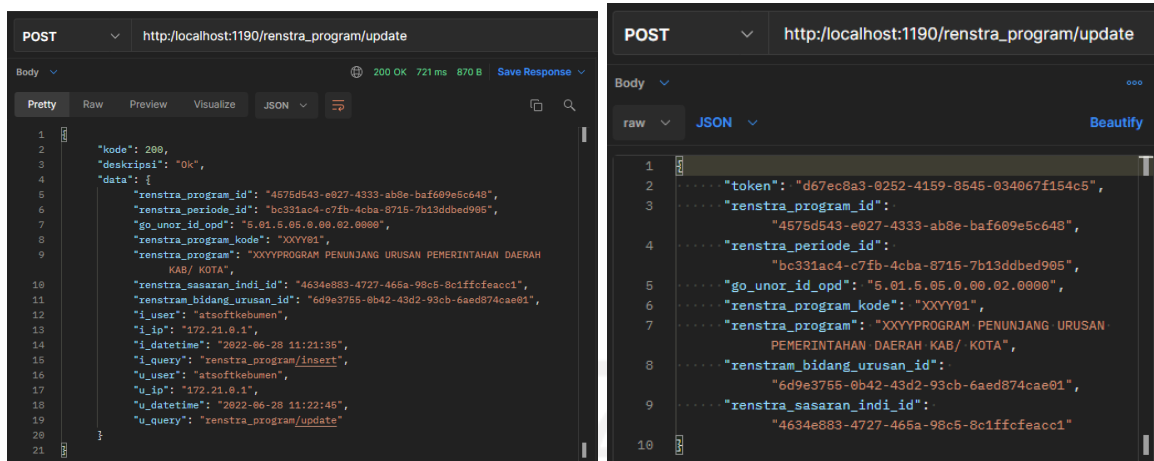
perlu memilih menu pada bagian ‘Body’ untuk membuat request ke *body*, lalu pilih *header* yang digunakan yaitu ‘raw’. Selanjutnya mengisi *renstra_periode_id*, *go_unor_id_opd*, *renstra_program_kode*, *renstra_program* pada bagian *body text* seperti pada **Gambar 3.20**. Kemudian developer perlu mengirim request dengan menekan tombol ‘Send’. Apabila proses berhasil, server akan memberikan *response*.



Gambar 3.20 Langkah menambah data Renstra program (Kiri) dan *Response* menambah data Renstra program berhasil (Kanan)

d. *Response* ubah data Rencana Strategis (Renstra) program

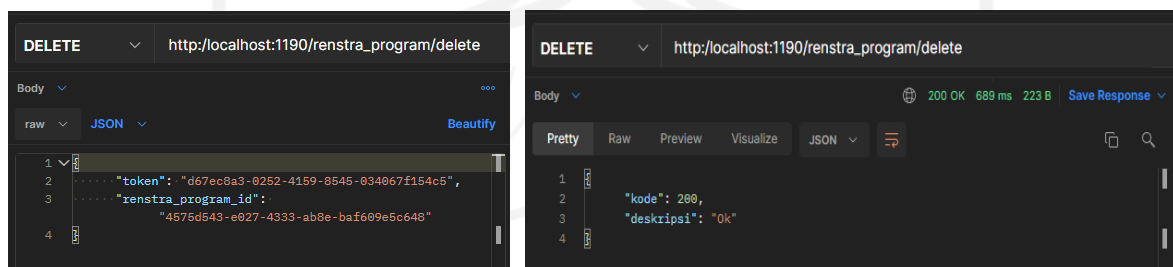
Endpoint `/renstra_program/update` digunakan untuk mengubah data Renstra program. Pengujian ini dilakukan dengan mengirimkan *request* data ke *endpoint* tersebut dengan menggunakan *method* POST dan memasukkan *url* `http://localhost:1191/renstra_program/update`. Untuk mengubah data Renstra program, perlu memilih menu pada bagian ‘Body’ untuk membuat *request* ke *body*, lalu pilih *header* yang digunakan yaitu ‘raw’. Selanjutnya isi *renstra_program_id* pada bagian *body text* untuk mengubah data sesuai dengan *id* data Renstra program seperti pada **Gambar 3.21**. Pada gambar tersebut dilakukan update pada *renstra_program_kode*, dan *renstra_program*. Selanjutnya, Developer perlu mengirim *request* dengan menekan tombol ‘Send’. Apabila proses berhasil, server akan memberikan *response*.



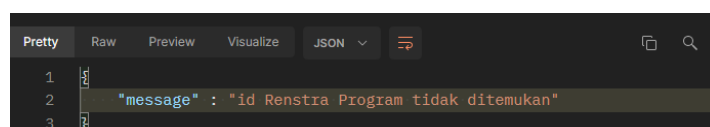
Gambar 3.21 Langkah mengubah data berdasarkan *id* Renstra program (Kiri) dan *Response* mengubah data berdasarkan *id* Renstra program berhasil (Kanan)

e. Response menghapus data Rencana Strategis (Renstra) program

Endpoint `Url/renstra_program/delete` digunakan untuk menghapus data Renstra program. Pengujian ini dilakukan dengan mengirimkan *request* data ke *endpoint* tersebut dengan menggunakan *method* DELETE dan memasukkan *url* `http://localhost:1191/renstra_program/delete`. Untuk menghapus data Renstra program, perlu memilih menu pada bagian ‘Body’ untuk membuat *request* ke *body*, lalu pilih *header* yang digunakan yaitu ‘raw’. Selanjutnya isi *renstra_program_id* pada bagian *body text* untuk menghapus data sesuai dengan *id* data Renstra program seperti pada **Gambar 3.22**. Selanjutnya, Developer perlu mengirim *request* dengan menekan tombol ‘Send’. Apabila proses berhasil, *server* akan memberikan *response* dan **Gambar 3.23** menampilkan *response* ketika *id* Renstra program tidak ditemukan.



Gambar 3.22 Langkah menghapus data berdasarkan *id* Renstra program (Kiri) dan *Response* menghapus data berdasarkan *id* Renstra program berhasil (Kanan)



Gambar 3.23 *Response* menghapus data berdasarkan *id* Renstra program tidak ditemukan

Adapun pengujian yang dilakukan terhadap seluruh endpoint Renstra program dengan *blackbox testing* seperti **Tabel 3.7**. *Response* yang diberikan saat pengujian API sudah sesuai dengan apa yang diharapkan, sehingga proses pertukaran data dapat dipastikan sudah sesuai dengan yang diinginkan.

Tabel 3.7 Daftar *endpoint* API Renstra Program

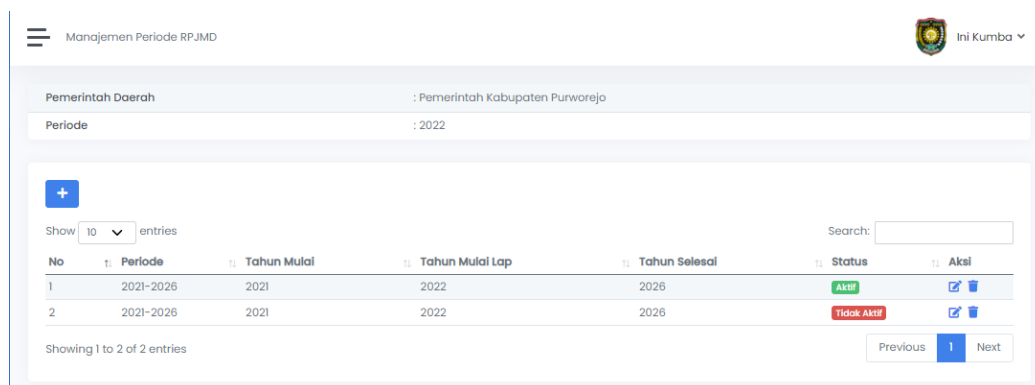
Endpoint	Fungsionalitas yang diuji	Hasil yang diharapkan	Hasil pengujian
/renstra_program/get_list	Menampilkan seluruh data Renstra program	Seluruh data Renstra program ditampilkan	Sukses
/renstra_program/get	Menampilkan data Renstra program berdasarkan <i>id</i>	Data Renstra program berdasarkan <i>id</i> ditampilkan	Sukses
/renstra_program/insert	Menambah data Renstra program	Data Renstra program ditambahkan	Sukses
/renstra_program/update	Mengubah data Renstra program	Data Renstra program diubah	Sukses
/renstra_program/delete	Menghapus data Renstra program	Data Renstra program dihapus	Sukses

3.3 Tampilan antarmuka Aplikasi berbasis *website* Atsoft E-Government (ATGOV)

Adapun tampilan yang sudah dibuat pada saat pelaksanaan magang dapat dilihat pada sub bab berikut ini.

3.3.1 Halaman Manajemen Rencana Pembangunan Jangka Menengah Daerah (RPJMD) Periode

Tampilan ini berfungsi untuk mengelola seluruh data RPJMD periode melalui fitur *CRUD* yang tersedia seperti: menambahkan periode baru, edit dan update data RPJMD periode, serta menghapus data periode. Pada halaman ini, juga terdapat fitur *pagination*, pencarian, dan *limit show*. Halaman manajemen RPJMD periode dapat dilihat pada **Gambar 3.24**.



Gambar 3.24 Tampilan halaman manajemen RPJMD periode

3.3.2 Halaman tambah data Rencana Pembangunan Jangka Menengah Daerah (RPJMD) Periode

Selanjutnya, **Gambar 3.25** merupakan tampilan Halaman tambah data RPJMD Periode. Halaman ini berfungsi untuk menambah data pada RPJMD periode dan disimpan pada database proyek ATGOV. Untuk melakukan proses tambah data, pengguna wajib mengisi nama periode, tahun mulai, tahun mulai lap, tahun selesai dan mengisi *checkbox* jika periode tersebut aktif maka pilih centang, jika tidak aktif maka lewati kolom tersebut. Untuk menuju halaman ini, dengan cara memilih ikon “tambah” pada halaman manajemen RPJMD Periode

Gambar 3.25 Tampilan halaman tambah data RPJMD periode

3.3.3 Halaman edit data Rencana Pembangunan Jangka Menengah Daerah (RPJMD) Periode

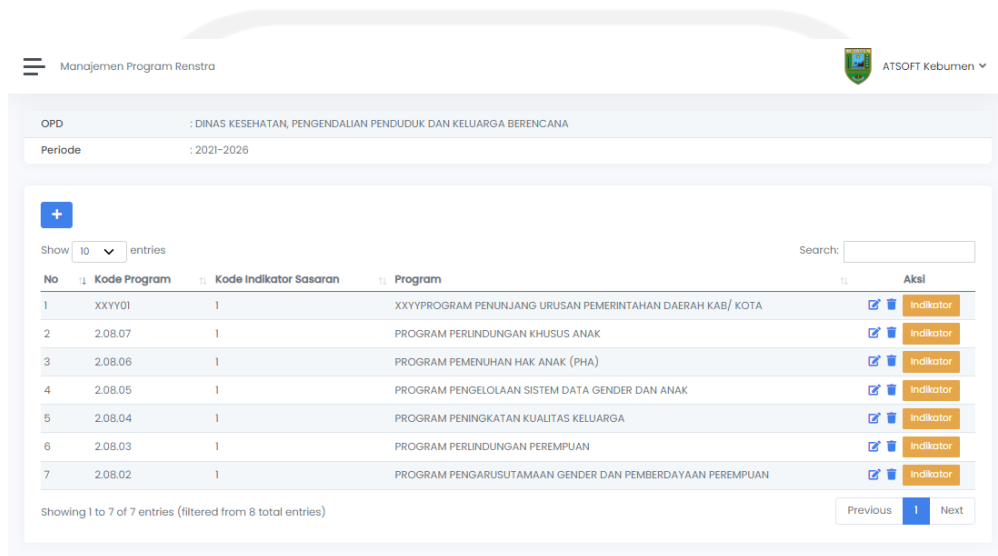
Selanjutnya, tampilan Halaman edit data RPJMD Periode dapat dilihat pada **Gambar 3.26**. Halaman ini merupakan halaman untuk melakukan edit data RPJMD periode yang sudah ada pada database ATGOV. Untuk melakukan edit data, pengguna dapat mengubah data yang tersedia pada form di halaman tersebut. Untuk menuju halaman ini, dengan cara memilih ikon pada bagian “Aksi” pada halaman manajemen RPJMD Periode.

Gambar 3.26 Tampilan halaman edit data RPJMD periode

3.3.4 Halaman manajemen Rencana Strategis (Renstra) Program

Sub fitur Renstra program adalah sub fitur yang mempermudah SKPD dalam dokumentasi rumusan program untuk keberhasilan perencanaan strategis pembangunan

daerah. Adapun tampilan pada Renstra program yaitu: Manajemen program Renstra, Halaman tambah data Renstra program, dan Halaman edit data Renstra program. Tampilan manajemen Renstra program digunakan untuk memanajemen seluruh data Renstra program melalui fitur *CRUD* yang tersedia seperti: menambahkan program baru, edit dan update data Renstra program, serta menghapus data program. Tampilan manajemen Renstra program dapat dilihat pada **Gambar 3.27**. Pada halaman ini, juga terdapat fitur *pagination*, pencarian, dan *limit show*.



Gambar 3.27 Halaman manajemen program Renstra

3.3.5 Halaman tambah data Rencana Strategis (Renstra) program

Selanjutnya, **Gambar 3.28** merupakan tampilan Halaman tambah data Renstra program. Halaman ini berfungsi untuk menambah data pada Renstra program dan disimpan pada *database* ATGOV. Untuk melakukan proses tambah data, pengguna wajib memilih indikator sasaran, mengisi kode program, program, dan memilih bidang urusan pemerintah pada form yang tersedia. Untuk menuju halaman ini, dengan cara memilih ikon “+” pada halaman manajemen Renstra program.

Gambar 3.28 Halaman tambah data Renstra program

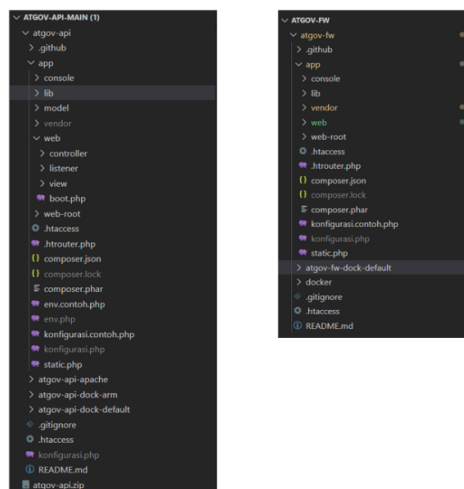
3.3.6 Halaman edit data Rencana Strategis (Renstra) program

Selanjutnya, tampilan Halaman edit data Renstra program dapat dilihat pada **Gambar 3.29**. Halaman ini merupakan halaman untuk melakukan edit data Renstra program yang sudah ada pada *database* ATGOV. Untuk melakukan edit data, pengguna dapat mengubah data yang tersedia pada form di halaman tersebut. Untuk menuju halaman ini, dengan cara memilih ikon pada bagian “Aksi” pada halaman manajemen Renstra program.

Gambar 3.29 Halaman edit data Renstra program

3.4 Mekanisme komunikasi antara atgov-fw dan atgov-api

Seperti apa yang telah dijelaskan bahwa pengembangan *backend* berbasis REST API pada E-Government digunakan untuk melakukan pertukaran data antara *client* dan *server*. Penjelasan mekanisme komunikasi akan diambil dari salah satu kasus pada saat menambahkan program pada Renstra program. Dalam pengembangan aplikasi, terdapat struktur folder pada atgov-fw (*client*) dan atgov-api (*server*) yang dapat dilihat pada **Gambar 3.30**



Gambar 3.30 Struktur folder pada atgov-api (Kiri) dan atgov-fw (Kanan)

Untuk menjalankan *client* dan *server* perlu mengeksekusi *docker-compose up -d*. Client akan berjalan pada *port* dan server akan berjalan pada *port* 1190. Selanjutnya pada **Gambar 3.31** merupakan halaman manajemen program Renstra.

No	Kode Program	Kode Indikator Sasaran	Program	Aksi
1	XXYY01	1	XXYYPROGRAM PENUNJANG URUSAN PEMERINTAHAN DAERAH KAB/ KOTA	☒ Indikator
2	2.08.07	1	PROGRAM PERLINDUNGAN KHUSUS ANAK	☒ Indikator
3	2.08.06	1	PROGRAM PEMENUHAN HAK ANAK (PHA)	☒ Indikator
4	2.08.05	1	PROGRAM PENGELOLAAN SISTEM DATA GENDER DAN ANAK	☒ Indikator
5	2.08.04	1	PROGRAM PENINGKATAN KUALITAS KELUARGA	☒ Indikator
6	2.08.03	1	PROGRAM PERLINDUNGAN PEREMPUAN	☒ Indikator
7	2.08.02	1	PROGRAM PENGARUSUTAMAAN GENDER DAN PEMBERDAYAAN PEREMPUAN	☒ Indikator

Gambar 3.31 Halaman manajemen program Renstra

Pada sisi *client*, ketika Atsoft E-Government (ATGOV) telah berjalan pada *browser*, pengguna dapat menekan tombol ‘+’ yang ada pada kiri atas halaman manajemen program Renstra untuk menambah data baru program seperti pada **Gambar 3.31**. Saat tombol tersebut ditekan, maka file *form.phtml* pada *atgov-fw* akan dipanggil dan kemudian menuju ke halaman tambah data program Renstra seperti pada **Gambar 3.32** untuk menambah program Renstra baru. Untuk melakukan tambah data, pengguna perlu mengisi form yang tersedia dan jika sudah dapat mengirim form dengan menekan tombol ‘Submit’

Gambar 3.32 Form tambah data program Renstra

Pada saat ‘Submit’ ditekan, fungsi `insertAction()` yang berada file *RenstraProgramController* akan dipanggil seperti pada **Gambar 3.33**. Setelah fungsi tersebut berhasil dipanggil, pertama client mengirim *request* melalui `$this->request->isPost()`. Selanjutnya, dilakukan validasi yang berguna untuk memberikan peringatan berupa *result error* apabila ada kekeliruan pada saat proses menambah data dan pengguna dapat mengulang

kembali hingga input data sudah benar. Jika validasi berhasil, maka nilai yang sudah dimasukkan kedalam form tadi akan ditampilkan di masing-masing *field*.

```
function insertAction()
{
$this->view->pick($this->router->getControllerName() . '/form');
$this->pt = "Tambah " . $this->x;
if ($this->request->isPost()) {
    $data['renstra_periode_id'] = $this->session->get('renstra_periode_id');
    $data['go_unor_id_opd'] = $this->session->get('go_unor_id');
    $data['renstra_program_kode'] = $this->request->get('renstra_program_kode');
    $data['renstra_program'] = $this->request->get('renstra_program');
    $data['renstra_sasaran_indi_id'] = $this->request->get('renstra_sasaran_indi_id')
    ?: null;
    $result = $this->req('renstra_program/insert', $data);
    if ($result->kode != 200) return $this->flashSession->error($result->deskripsi);
    $this->flashSession->success($result->deskripsi);
    return $this->response->redirect($this->router->getControllerName() . '/index');
}
}
```

Gambar 3.33 Fungsi insertAction() pada *RenstraProgramController*

Pada hasil request data, URL yang digunakan adalah `result = $this->req('renstra_program/insert', $data);` dimana variabel `req` merupakan deklarasi fungsi `req` pada file *AController* yang digunakan untuk memanggil *konfigurasi.php* pada *atgov-fw*. file *konfigurasi.php* berisi alamat server <http://localhost:1190/> dan *renstra_program/insert* merupakan endpoint yang digunakan. Artinya *request* memanggil method POST dengan endpoint *renstra_program/insert*.

Selanjutnya, pada sisi server, terdapat fungsi `insertAction()` untuk membuat program Renstra baru pada file *RenstraProgramController.php* seperti pada **Gambar 3.34**.

```
public function insertAction()
{
    $p = $this->request->getJsonRawBody();
    if (!isset($p->renstra_periode_id) ||
        !isset($p->go_unor_id_opd) ||
        !isset($p->renstra_program_kode) ||
        !isset($p->renstra_program) ||
        !isset($p->renstra_program_indi_id))
    {
        return $this->setJsonRespon('renstra_periode_id, go_unor_id_opd, renstra_program_kode, renstra_program wajib', 1);
    }
    $x = new RenstraProgram();
    $kol_arra = [
        'renstra_periode_id',
        'go_unor_id_opd',
        'renstra_program_kode',
        'renstra_program',
        'renstra_sasaran_indi_id'];
    foreach ($kol_arra as $kol) {
        if (property_exists($p, $kol)) $x->{$kol} = $p->{$kol};
    }
    if (!$x->create()) {
        $this->dm['error'] = $x->getMessages();
        return $this->setJsonRespon('terjadi kesalahan', 1);
    }
    else {
        return $this->setJsonRespon(null, null, $x);
    }
}
```

Gambar 3.34 Fungsi insertAction() pada *RenstraProgramController atgov-api*

Fungsi tersebut akan dipanggil ketika *endpoint* yang terhubung dipanggil. *RenstraProgramController* tersebut terhubung dengan koneksi *database* pada file *konfigurasi.php*, dan tabel pada model *RenstraProgram.php*. sehingga, saat *insertAction()* dipanggil, data yang diisi melalui *form* akan ditambahkan pada tabel Renstra program dan data akan disimpan dalam *database* MySQL.

Hasil dan pembahasan REST API sesuai dengan yang telah dijelaskan. Layanan REST API mampu memisahkan dan menyederhanakan interaksi antara *client* dan *server*. Perubahan yang terjadi pada *server* tidak mempengaruhi *client* sehingga kinerja *client* dan *server* menjadi lebih efisien karena kedua sisi dapat mengembangkan modul tanpa mempengaruhi proses pertukaran data



BAB IV

REFLEKSI PELAKSANAAN MAGANG

4.1 Teknis

Selama kegiatan magang, penulis terlibat dalam pengembangan Sistem Informasi Atsoft *E-Government*. Dalam Sistem informasi Atsoft *E-Government*, penulis fokus pada pengembangan REST API digunakan sebagai sarana untuk bertukar data antara client-server untuk Fitur Rencana Pembangunan Jangka Menengah Daerah (RPJMD) dan Rencana Strategis (Renstra). Pada awalnya mengalami kesulitan karena harus beradaptasi terlebih dahulu menggunakan Phalcon framework pada pengembangan Atsoft *E-Government*. Selain itu juga harus melakukan pendalaman materi terkait pembuatan REST API sebagai sarana bertukar data. Namun, sedikit demi sedikit dapat mengimplementasikan REST API mulai dari fitur CRUD pada RPJMD dan Renstra. Pada proses pengembangan, penulis dapat belajar dari kesalahan, misalnya kesalahan dalam penulisan alamat endpoint atau API. Adapun, dalam pembuatan API, endpoint wajib ditulis sesuai dengan aturan penulisan untuk memudahkan developer lain menulis ulang endpoint tersebut. Dalam Fitur RPJMD dan Renstra, semua operasi dilakukan terhadap entitas yang berbeda, sehingga semua operasi berada dalam endpoint yang berbeda. Contohnya, apabila ingin mengambil data semua RPJMD periode dalam database memakai URL `http://localhost:1190/rpjmd_periode/get_list`. Penggunaan *get_list* pada endpoint digunakan untuk memanggil fungsi *get_list* pada *controller* sehingga dapat memperlihatkan seluruh data RPJMD periode. Contoh lainnya untuk menghapus data RPJMD periode dengan *id* maka URL yang digunakan adalah `http://localhost:1190/rpjmd_periode/delete` dengan metode DELETE. Penggunaan *delete* pada endpoint digunakan untuk memanggil fungsi *delete* pada *controller*. Untuk membuktikan bahwa implementasi REST API yang telah dibuat dapat berfungsi dengan baik, penulis melakukan pengujian API yang telah dibuat. Pengujian yang dilakukan menggunakan *tools* Postman sehingga mudah dilakukan, karena *tools* ini dapat digunakan sebagai pengganti REST Client apabila client belum selesai dikembangkan. Antarmuka yang digunakan dalam Postman mudah dipahami dan dipelajari karena bersifat *user friendly*.

4.2 Non Teknis

Selama pelaksanaan program magang di CV. Atsoft Teknologi, banyak hal non-teknis yang penulis dapatkan. Misalnya, dalam hal komunikasi. Dalam sebuah pekerjaan pasti tidak terlepas dari namanya komunikasi. Adakalanya penulis harus bekerja secara individu maupun bekerja bersama dengan tim. Komunikasi merupakan hal yang penting untuk menjaga keharmonisan dan menghindari misunderstanding dalam sebuah tim. Misalnya saat penulis merasa kesulitan atau merasa kurang paham dengan tugas yang dikerjakan, penulis harus berkomunikasi dengan bertanya kepada anggota tim yang lainnya maupun dengan Project Manager atau supervisor. Lalu saat penulis ingin meminta saran atau memberi saran pengerjaan kepada tim, maka penulis harus bisa membicarakan dengan baik agar diterima dengan jelas.

Adanya tuntutan untuk aktif berkomunikasi, membuat kepercayaan diri penulis semakin naik. Pada awalnya penulis masih merasa malu dan sungkan untuk bertanya karena masih kesulitan menyesuaikan diri lingkungan kerja dan menepis rasa takut, perlahan mulai hilang dan membiasakan diri untuk kritis.

Berada di dunia kerja memang berbeda dengan dunia perkuliahan. Penulis tidak dapat seenaknya mengerjakan tugas yang diberikan karena hal ini menyangkut kepentingan perusahaan dan harus mengerjakan sesuai dengan spesifikasi yang diberikan. Karena itu penulis harus mampu mengatur waktu dengan baik agar dapat menyelesaikan pekerjaan kantor maupun tugas kuliah dengan tepat waktu, seimbang dan sebaik mungkin. Selain itu, pelaksanaan magang membuat penulis mempelajari hal baru. Penulis dapat mengeksplor hal baru melalui tugas yang diberikan dan mengasah kemampuan

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan pelaksanaan proyek magang yang telah dilakukan didapatkan kesimpulan yaitu:

- Kegiatan magang menghasilkan REST API pada sistem informasi Atsoft *E-Government* (ATGOV) yang mampu memisahkan dan menyederhanakan interaksi antara *client* dan *server*. Perubahan yang terjadi pada *server* tidak mempengaruhi *client* sehingga kinerja *client* dan *server* menjadi lebih efisien karena kedua sisi dapat mengembangkan modul tanpa mempengaruhi proses pertukaran data.
- Kegiatan magang juga menghasilkan fitur yang telah dibuat meliputi Rencana Pembangunan Jangka Menengah Daerah (RPJMD) dan fitur Rencana strategis (Renstra) yang memudahkan Pemerintah daerah dalam menjabarkan kebijakan pembangunan yang merespon kebutuhan masyarakat serta visi dan misi kepala daerah terpilih dalam tujuan, sasaran, dan program serta menentukan strategi atau arahan serta pengambilan keputusan alokasi sumber daya.
- Seluruh tugas yang diberikan oleh *Project Manager*(PM) meliputi pembuatan REST API pada fitur RPJMD dan fitur Renstra serta melakukan pengujian pada REST API yang telah dibuat sudah dikerjakan dengan baik sehingga tujuan dan manfaat pada laporan telah terpenuhi.

5.2 Saran

Adapun saran yang ingin penulis sampaikan terkait pelaksanaan magang akan ditujukan kepada:

- Perusahaan Magang

Saran untuk perusahaan magang, diharapkan kedepannya perlu terus melibatkan mahasiswa/i magang ketika terdapat *meeting* perusahaan. Sehingga, mahasiswa/i magang dapat ikut serta menganalisa masalah yang terdapat pada *meeting* tersebut. Selain itu juga diperlukan training terkait *task* apa yang akan dikerjakan beserta standar tools maupun teknologi yang digunakan supaya proyek yang dikerjakan jelas secara manajemennya.

- Prodi Informatika UII

Meningkatkan pemantauan terhadap mahasiswa/i yang melaksanakan magang dengan cara memberikan kuisioner yang berisi minat posisi magang agar mahasiswa dapat menempatkan posisi magang sesuai dengan jobdesknya. Terkait masalah komunikasi, sebaiknya jika ada pengumuman atau perubahan hal-hal penting disampaikan secara jelas dan tidak mendadak.

- Diri Sendiri

Saran untuk diri sendiri adalah lebih ditingkatkan belajar mengenai hal-hal teknis yang sudah pernah dicicipi selama magang dan juga untuk hal-hal non teknis dapat diterapkan di kehidupan sebagai bekal untuk memasuki dunia kerja. Selain itu juga penulis berharap dapat melanjutkan pengembangan proyek yang belum terselesaikan



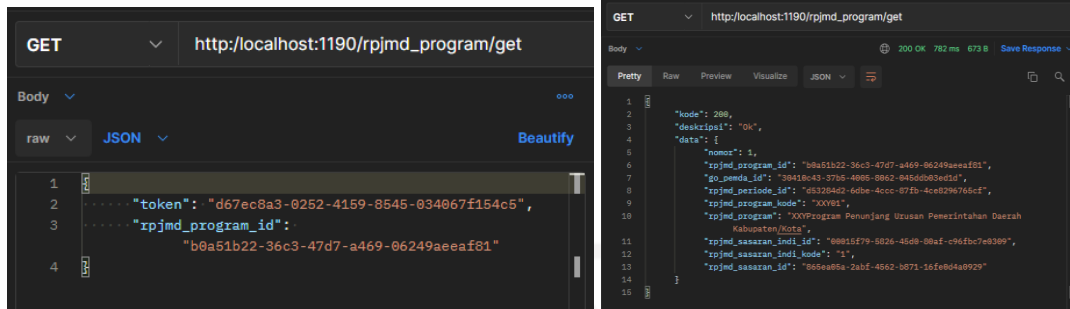
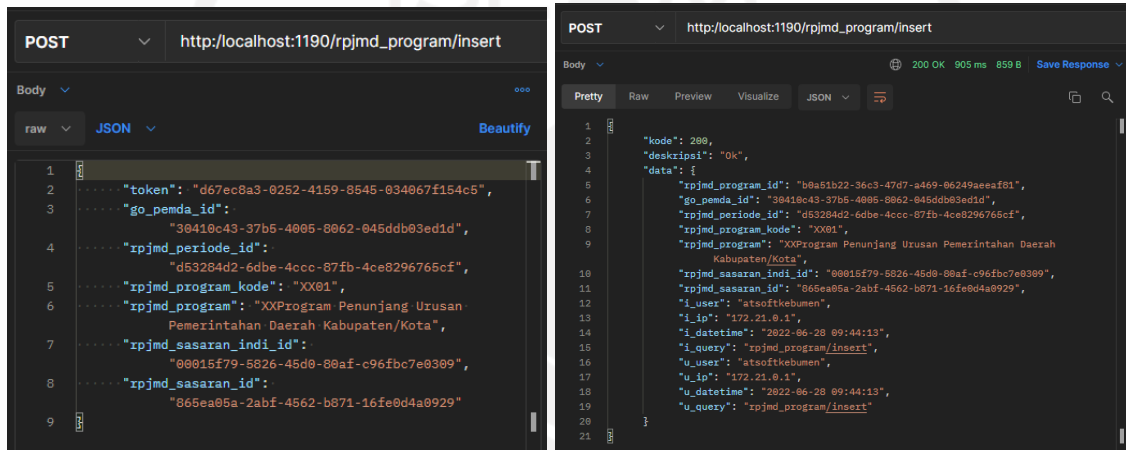
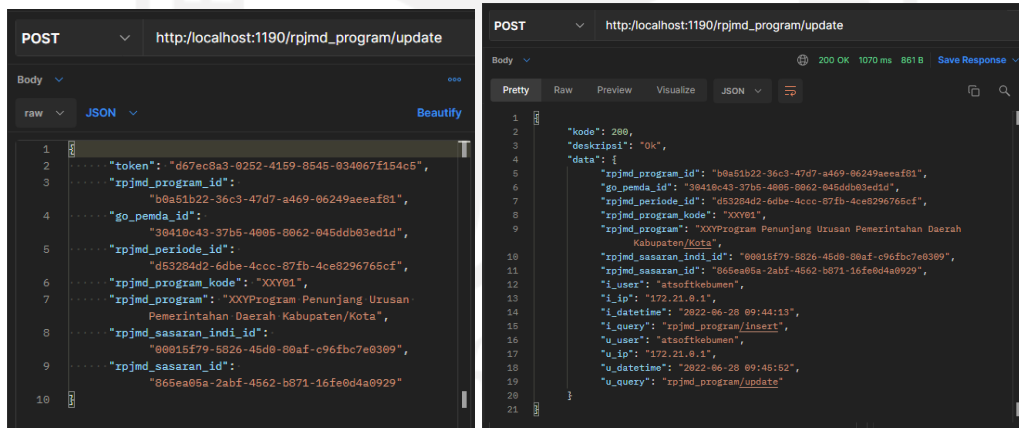
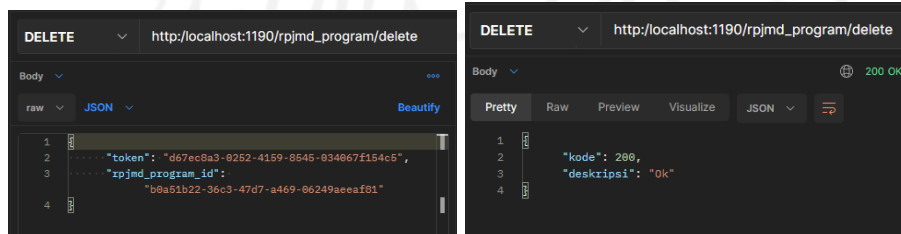
DAFTAR PUSTAKA

- Agnes, D., S. (September 2020). *Memahami API, REST API, RESTful API*. Dipetik Juli 14, 2022, dari <https://medium.com/wripolinema/memahami-api-rest-api-dan-restful-api-5fd2327edd3c>
- Ariata, C. (Maret, 2018) *Tutorial Phalcon Lengkap untuk Pemula*. Dipetik Juli 14, 2022, dari <https://www.hostinger.co.id/tutorial/tutorial-phalcon>.
- Babakan, A. (Desember 2019). *A comparison between Laravel and Phalcon*. Dipetik Juli 14, 2022, dari <https://dev.to/adnanbabakan/a-comparison-between-laravel-and-phalcon-2471>
- Bapedda, A. (September, 2017). *Pengertian RENSTRA*. Dipetik Juli 14, 2022, dari <https://bappeda.bulelengkab.go.id/informasi/detail/artikel/pengertian-renstra-68>
- Budi W., A., Ananda, A., & Triyanjaya, D. (2017). Penerapan Data JSON Untuk Mendukung Pengembangan Aplikasi Pada Perguruan Tinggi Dengan Teknik Restfull Dan Web Service. *Technomedia Journal (TMJ)* (Vol. 2, Issue 1).
- Feradis, N. (Februari, 2021). *Penyusunan Rencana Pembangunan Jangka Menengah Daerah*. Dipetik Juli 14, 2022, dari <https://kumparan.com/feradis-nurdin/penyusunan-rencana-pembangunan-jangka-menengah-daerah-1vCKcsg4ZOW>
- Fielding, R., T. (2000). Architectural Styles and the Design of Network-based Software Architectures. *Doctoral Disertation*.
- Hasanuddin, Asgar, H., & Hartono, B (2022). Rancang Bangun REST API Aplikasi Weshare Sebagai Upaya Mempermudah Pelayanan Donasi Kemanusiaan, *Jurnal Informatika, Teknologi Dan Sains*, vol. 4(1), Dipetik Juli 14, 2022 dari <http://jurnal.uts.ac.id/index.php/JINTEKS/article/view/1474>
- Kuswara, H., & Kusmana, D., (2017). Sistem Informasi Absensi Siswa Berbasis Web Dengan SMS Gateway Pada Sekolah Menengah Kejuruan Al – Munir Bekasi. *Indonesian Journal on Networking and Security*, 6(2), 17–22. Dipetik Juli 15, 2022, dari <http://ijns.org/journal/index.php/ijns/article/view/22>
- Pahlevy, B., A. (2017). Rancang Bangun Application Programming Interface (API) Ongkos Kirim Dan Tracking Ekspedisi Indonesia.
- Pangestika, R. (2021). Pengembangan Back-End Sistem Informasi Pendataan Sekolah Desa Komunitas Pendar Foundation Yogyakarta.

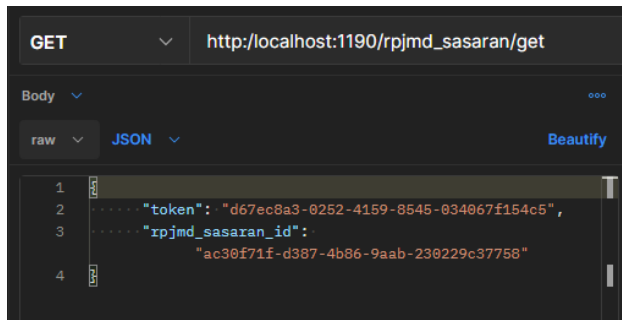
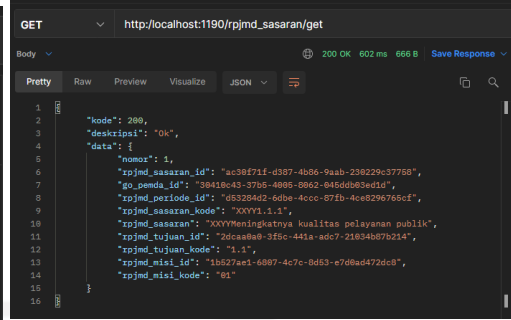
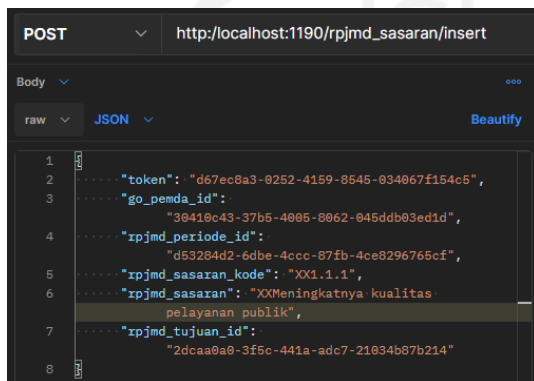
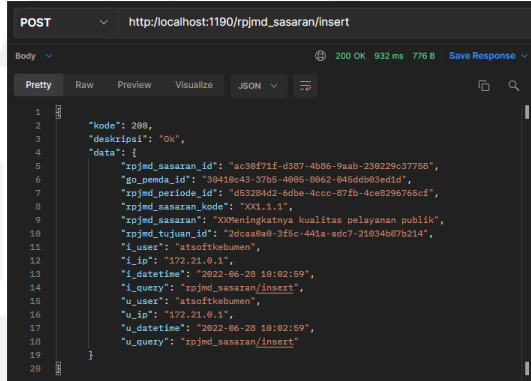
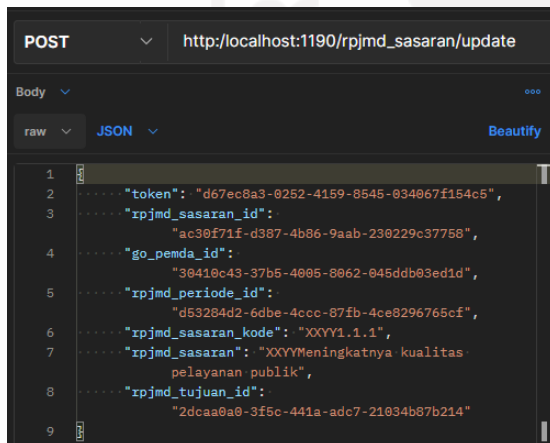
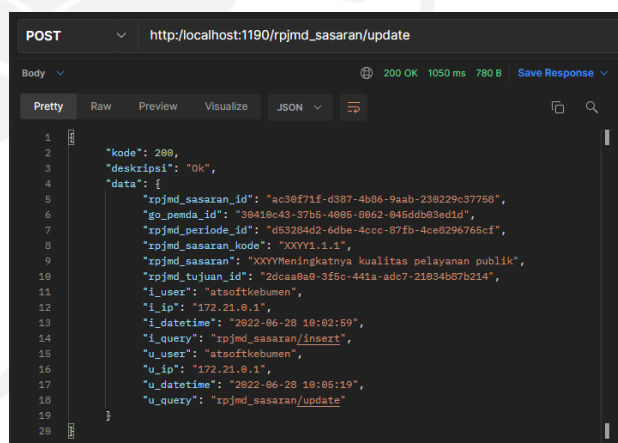
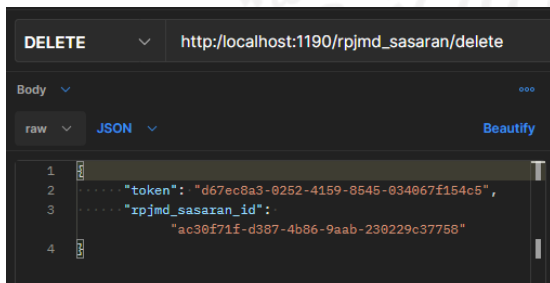
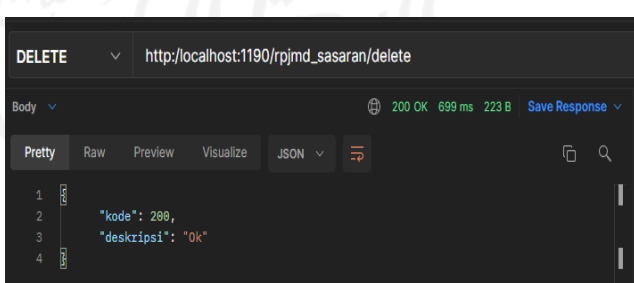
- Petrosyan, V. (Juni 2021) *A Guide to Popular PHP Frameworks for Beginners*. Dipetik Juli 14, 2022, dari <https://www.searchenginejournal.com/guide-popular-php-frameworks-beginners/180922>
- Phalcon PHP Framework Documentation Release 3.1.1 Phalcon Team (2017). Dipetik Juli 14, 2022, dari <https://docs.phalcon.io/4.0/en/tutorial-rest>.
- Prokofyeva, N., & Boltunova, V. (2016). Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems. *Procedia Computer Science*, 104, 51–56. <https://doi.org/10.1016/j.procs.2017.01.059>.
- Suharno, H., Gunantara, N., & Sudarma, M. (2020). Analisis Penerapan Metode Scrum Pada Sistem Informasi Manajemen Proyek Dalam Industri & Organisasi Digital. *Majalah Ilmiah Teknologi Elektro*, 19(2), 203-210. doi:10.24843/MITE.2020.v19i02.P12.
- Irawan, B., & Hidayat, M., N. (2020). E-government: konsep, esensi dan studi kasus. *Mulawarman University Press*

LAMPIRAN

1. Pembuatan API RPJMD Program



2. Pembuatan API RPJMD Sasaran



3. Pembuatan API RPJMD tujuan

```
GET http://localhost:1190/rpjmd_tujuan/get

Body
raw JSON Beautify

1
2
3
4
{
  "token": "d67ec8a3-0252-4159-8545-034067f154c5",
  "rpjmd_tujuan_id": "35f185a2-98c3-482a-84f4-f9ffdeabc291"
}
```

```
GET http://localhost:1190/rpjmd_tujuan/get

Body
Pretty Raw Preview Visualize JSON Save Response

1
2
3
4
5
6
7
8
9
10
11
12
13
14
{
  "kode": 200,
  "deskripsi": "Ok",
  "data": {
    "token": "d67ec8a3-0252-4159-8545-034067f154c5",
    "rpjmd_tujuan_id": "35f185a2-98c3-482a-84f4-f9ffdeabc291",
    "go_pemda_id": "30410c43-37b5-4095-8062-045ddb03ed1d",
    "rpjmd_periode_id": "d53284d2-6dbe-4ccc-87fb-4ce8296765cf",
    "rpjmd_tujuan_kode": "XXYY1.1",
    "rpjmd_tujuan": "XXYYMewujudkan reformasi tata kelola pemerintahan yang bersih dan baik",
    "rpjmd_misi_id": "1b527ae1-6807-4c7c-8d53-e7d0ad472dc8",
    "rpjmd_misi_kode": "01"
  }
}
```

```
POST http://localhost:1190/rpjmd_tujuan/insert

Body
raw JSON Beautify

1
2
3
4
5
6
7
8
{
  "token": "d67ec8a3-0252-4159-8545-034067f154c5",
  "go_pemda_id": "30410c43-37b5-4095-8062-045ddb03ed1d",
  "rpjmd_periode_id": "d53284d2-6dbe-4ccc-87fb-4ce8296765cf",
  "rpjmd_tujuan_kode": "XX1.1",
  "rpjmd_tujuan": "XXMewujudkan reformasi tata kelola pemerintahan yang bersih dan baik",
  "rpjmd_misi_id": "1b527ae1-6807-4c7c-8d53-e7d0ad472dc8"
}
```

```
POST http://localhost:1190/rpjmd_tujuan/insert

Body
Pretty Raw Preview Visualize JSON Save Response

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
{
  "kode": 200,
  "deskripsi": "Ok",
  "data": {
    "rpjmd_tujuan_id": "35f185a2-98c3-482a-84f4-f9ffdeabc291",
    "go_pemda_id": "30410c43-37b5-4095-8062-045ddb03ed1d",
    "rpjmd_periode_id": "d53284d2-6dbe-4ccc-87fb-4ce8296765cf",
    "rpjmd_tujuan_kode": "XX1.1",
    "rpjmd_tujuan": "XXMewujudkan reformasi tata kelola pemerintahan yang bersih dan baik",
    "rpjmd_misi_id": "1b527ae1-6807-4c7c-8d53-e7d0ad472dc8",
    "i_user": "atsoftkebumen",
    "i_ip": "172.21.0.1",
    "i_datetime": "2022-06-28 10:16:35",
    "i_query": "rpjmd_tujuan/insert",
    "u_user": "atsoftkebumen",
    "u_ip": "172.21.0.1",
    "u_datetime": "2022-06-28 10:16:35",
    "u_query": "rpjmd_tujuan/insert"
  }
}
```

```
POST http://localhost:1190/rpjmd_tujuan/update

Body
raw JSON Beautify

1
2
3
4
5
6
7
8
9
{
  "token": "d67ec8a3-0252-4159-8545-034067f154c5",
  "rpjmd_tujuan_id": "35f185a2-98c3-482a-84f4-f9ffdeabc291",
  "go_pemda_id": "30410c43-37b5-4095-8062-045ddb03ed1d",
  "rpjmd_periode_id": "d53284d2-6dbe-4ccc-87fb-4ce8296765cf",
  "rpjmd_tujuan_kode": "XXYY1.1",
  "rpjmd_tujuan": "XXYYMewujudkan reformasi tata kelola pemerintahan yang bersih dan baik",
  "rpjmd_misi_id": "1b527ae1-6807-4c7c-8d53-e7d0ad472dc8"
}
```

```
POST http://localhost:1190/rpjmd_tujuan/update

Body
Pretty Raw Preview Visualize JSON Save Response

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
{
  "kode": 200,
  "deskripsi": "Ok",
  "data": {
    "rpjmd_tujuan_id": "35f185a2-98c3-482a-84f4-f9ffdeabc291",
    "go_pemda_id": "30410c43-37b5-4095-8062-045ddb03ed1d",
    "rpjmd_periode_id": "d53284d2-6dbe-4ccc-87fb-4ce8296765cf",
    "rpjmd_tujuan_kode": "XXYY1.1",
    "rpjmd_tujuan": "XXYYMewujudkan reformasi tata kelola pemerintahan yang bersih dan baik",
    "rpjmd_misi_id": "1b527ae1-6807-4c7c-8d53-e7d0ad472dc8",
    "i_user": "atsoftkebumen",
    "i_ip": "172.21.0.1",
    "i_datetime": "2022-06-28 10:16:35",
    "i_query": "rpjmd_tujuan/insert",
    "u_user": "atsoftkebumen",
    "u_ip": "172.21.0.1",
    "u_datetime": "2022-06-28 10:17:45",
    "u_query": "rpjmd_tujuan/update"
  }
}
```

```
DELETE http://localhost:1190/rpjmd_tujuan/delete

Body
raw JSON Beautify

1
2
3
4
{
  "token": "d67ec8a3-0252-4159-8545-034067f154c5",
  "rpjmd_tujuan_id": "35f185a2-98c3-482a-84f4-f9ffdeabc291"
}
```

```
DELETE http://localhost:1190/rpjmd_tujuan/delete

Body
Pretty Raw Preview Visualize JSON Save Response

1
2
3
4
{
  "kode": 200,
  "deskripsi": "Ok"
}
```

4. Pembuatan API Renstra Periode

```
GET http://localhost:1190/renstra_periode/get

Body
raw JSON Beautify

1 [{"token": "d67ec8a3-0252-4159-8545-034067f154c5",
2  "renstra_periode_id":
3  "2a0d0fec-e1b6-4e60-a2ea-8bff26e308ca"}
4 ]
```

```
GET http://localhost:1190/renstra_periode/get

Body
Pretty Raw Preview Visualize JSON Save Response

1 [{"kode": 200,
2  "deskripsi": "Ok",
3  "data": [
4  {
5    "token": "d67ec8a3-0252-4159-8545-034067f154c5",
6    "renstra_periode_id": "2a0d0fec-e1b6-4e60-a2ea-8bff26e308ca",
7    "renstra_periode": "2026-2031",
8    "tahun_mulai": "2026",
9    "tahun_selesai": "2031",
10   "flag_aktif": "0",
11   "flag_default": "0",
12   "rpjmd_periode_id": "d53284d2-6dbe-4ccc-87fb-4ce8296765cf"
13 }
14 ]}]
```

```
POST http://localhost:1190/renstra_periode/insert

Body
raw JSON Beautify

1 [{"token": "d67ec8a3-0252-4159-8545-034067f154c5",
2  "renstra_periode": "2026-2031",
3  "tahun_mulai": "2026",
4  "tahun_selesai": "2031",
5  "flag_aktif": "0",
6  "flag_default": "0",
7  "rpjmd_periode_id":
8  "d53284d2-6dbe-4ccc-87fb-4ce8296765cf"}
9 ]
```

```
POST http://localhost:1190/renstra_periode/insert

Body
Pretty Raw Preview Visualize JSON Save Response

1 [{"kode": 200,
2  "deskripsi": "Ok",
3  "data": [
4  {
5    "renstra_periode_id": "2a0d0fec-e1b6-4e60-a2ea-8bff26e308ca",
6    "renstra_periode": "2026-2031",
7    "tahun_mulai": "2026",
8    "tahun_selesai": "2031",
9    "flag_aktif": "0",
10   "flag_default": "0",
11   "rpjmd_periode_id": "d53284d2-6dbe-4ccc-87fb-4ce8296765cf",
12   "i_user": "atsofkebumen",
13   "i_ip": "172.21.0.1",
14   "i_datetime": "2022-06-20 18:47:07",
15   "i_query": "renstra_periode/insert",
16   "u_user": "atsofkebumen",
17   "u_ip": "172.21.0.1",
18   "u_datetime": "2022-06-20 18:47:07",
19   "u_query": "renstra_periode/insert"
20 }
21 ]}]
```

```
POST http://localhost:1190/renstra_periode/update

Body
raw JSON Beautify

1 [{"token": "d67ec8a3-0252-4159-8545-034067f154c5",
2  "renstra_periode_id":
3  "2a0d0fec-e1b6-4e60-a2ea-8bff26e308ca",
4  "renstra_periode": "2026-2031",
5  "tahun_mulai": "2026",
6  "tahun_selesai": "2031",
7  "flag_aktif": "0",
8  "flag_default": "0",
9  "rpjmd_periode_id":
10 "d53284d2-6dbe-4ccc-87fb-4ce8296765cf"}
11 ]
```

```
POST http://localhost:1190/renstra_periode/update

Body
Pretty Raw Preview Visualize JSON Save Response

1 [{"kode": 200,
2  "deskripsi": "Ok",
3  "data": [
4  {
5    "renstra_periode_id": "2a0d0fec-e1b6-4e60-a2ea-8bff26e308ca",
6    "renstra_periode": "2026-2031",
7    "tahun_mulai": "2026",
8    "tahun_selesai": "2031",
9    "flag_aktif": "0",
10   "flag_default": "0",
11   "rpjmd_periode_id": "d53284d2-6dbe-4ccc-87fb-4ce8296765cf",
12   "i_user": "atsofkebumen",
13   "i_ip": "172.21.0.1",
14   "i_datetime": "2022-06-20 18:47:07",
15   "i_query": "renstra_periode/insert",
16   "u_user": "atsofkebumen",
17   "u_ip": "172.21.0.1",
18   "u_datetime": "2022-06-20 18:48:23",
19   "u_query": "renstra_periode/update"
20 }
21 ]}]
```

```
DELETE http://localhost:1190/renstra_periode/delete

Body
raw JSON Beautify

1 [{"token": "d67ec8a3-0252-4159-8545-034067f154c5",
2  "renstra_periode_id":
3  "2a0d0fec-e1b6-4e60-a2ea-8bff26e308ca"}
4 ]
```

```
DELETE http://localhost:1190/renstra_periode/delete

Body
Pretty Raw Preview Visualize JSON Save Response

1 [{"kode": 200,
2  "deskripsi": "Ok"}
3 ]
```