Pengembangan Aplikasi T-Feeder Untuk Pelaporan Pddikti Dengan Menggunakan Bahasa Laravel (Back-End)

Development Of The T-Feeder Application For Pddikti Reporting Using Laravel Language (Back-End)

PROYEK AKHIR

Vienna Khansa Khalida Ranutri Widjaja
6703213006

PROGRAM STUDI D3 KOMPUTERISASI AKUNTANSI
FAKULTAS ILMU TERAPAN
UNIVERSITAS TELKOM
BANDUNG, 2024

Untuk Orang Tua Tercinta

Terima kasih atas cinta, dukungan, dan doa yang tiada henti. Tanpa bimbingan dan pengorbanan kalian, karya ini tidak akan pernah terwujud.

Untuk segala ilmu, bimbingan, dan motivasi yang telah diberikan oleh pembimbing, saya sangat berterima kasih. Karena bimbingan Bapak/Ibu akan menjadi cahaya penuntun dalam setiap langkah saya.

Untuk dukungan, kebersamaan, dan semangat yang telah diberikan oleh sahabat saya. Kehadiran kalian memberikan warna dan makna dalam perjalanan saya.

Saya mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah membantu dan mendukung proses pengerjaan karya ini, baik secara langsung maupun tidak . Kontribusi dan kerjasama kalian sangat berarti bagi saya.

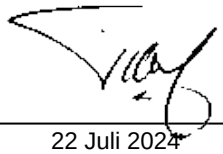
LEMBAR PENGESAHAN PROYEK AKHIR

Pengembangan Aplikasi T-Feeder Untuk Pelaporan Pddikti Dengan Menggunakan Bahasa Laravel (Back-End)

Development Of The T-Feeder Application For Pddikti Reporting Using Laravel Language (Back-End)

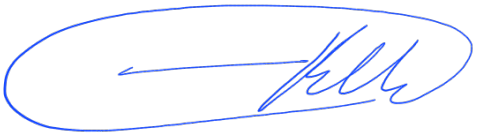
Penulis
Vienna Khansa Khalida Ranutri W
NIM 6703213006

Pembimbing I
Dr. Tora Fahrudin, S.T., M.T.
NIP 11850076



22 Juli 2024

Pembimbing II
Raswyshnoe Boing Kotjoprayudi, S.E., M.M.
NIP 10790005



Ketua Program Studi
Dr. Asniar, S.T., M.T.
NIP 14810007

Tanggal Pengesahan: 2024

PERNYATAAN

Dengan ini saya menyatakan bahwa:

1. Proyek Akhir ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik (Ahli Madya, Sarjana, Magister dan Doktor), baik di Fakultas Ilmu Terapan Universitas Telkom maupun di perguruan tinggi lainnya;
2. karya tulis ini murni gagasan, rumusan, dan penelitian saya sendiri, tanpa bantuan pihak lain, kecuali arahan tim pembimbing atau tim promotor atau penguji;
3. dalam karya tulis ini tidak terdapat cuplikan karya atau pendapat yang telah ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan menyebutkan nama pengarang dan dicantumkan dalam daftar pustaka;
4. saya mengijinkan karya tulis ini dipublikasikan oleh Fakultas Ilmu Terapan Universitas Telkom, dengan tetap mencantumkan saya sebagai penulis; dan

Pernyataan ini saya buat dengan sesungguhnya dan apabila pada kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh karena karya tulis ini, serta sanksi lainnya sesuai norma yang berlaku di Fakultas Ilmu Terapan Universitas Telkom.

Bandung, 19 Juli 2024

Pembuat pernyataan,

Vienna Khansa Khalida Ranutri W

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT atas rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan proyek akhir ini dengan tepat waktu. Proyek akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar pada Program Studi D3 Sistem Informasi Akuntansi, Fakultas Ilmu Terapan, Universitas Telkom.

Penulis menyadari bahwa penyusunan tugas akhir ini tidak lepas dari bantuan, dukungan, dan bimbingan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada :

1. Bapak Dr. Tora Fahrudin, S.T., M.T. selaku dosen pembimbing yang telah memberikan bimbingan, arahan, serta motivasi selama proses penyusunan proyek akhir ini.
2. Ibu Dr. Asniar, S.T., M.T. selaku Ketua Program Studi D3 Sistem Informasi Akuntansi, yang telah memberikan dukungan dan fasilitas selama penulis menempuh pendidikan.
3. Seluruh dosen dan staff di Fakultas Ilmu Terapan Universitas Telkom yang telah memberikan ilmu pengetahuan dan bantuan selama masa studi.
4. Kedua orang tua dan keluarga besar penulis yang selalu memberikan do'a, dukungan, dan semangat yang tiada henti.
5. Teman-teman seperjuangan yang telah memberikan bantuan dan kerjasama selama proses perkuliahan dan penyusunan proyek akhir ini.
6. Terima kasih kepada diriku, terima kasih karena telah mempercayaiku, terima kasih karena telah bekerja keras,dan terima kasih karena tidak pernah menyerah.

Penulis menyadari bahwa tugas akhir ini masih jauh dari sempurna. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun demi

kesempurnaan proyek akhir ini di masa mendatang. Semoga proyek akhir ini dapat bermanfaat bagi penulis khususnya, dan bagi para pembaca umumnya.

Akhir kata, penulis mengucapkan terima kasih.

Bandung, 19 Juli 2024

Penulis

ABSTRAK

Pengembangan aplikasi *T-Feeder* untuk pelaporan Pangkalan Data Pendidikan Tinggi (PDDikti) bertujuan untuk meningkatkan efisiensi dan akurasi dalam pengelolaan data pendidikan tinggi di Indonesia. Dalam penelitian ini, Aplikasi *T-Feeder* dirancang untuk memudahkan proses pelaporan data ke *Neo Feeder* PDDikti, yang meliputi data mahasiswa, dosen, kurikulum, dan aktivitas akademik lainnya. Penggunaan *Laravel* sebagai *backend* memberikan beberapa keuntungan, seperti struktur kode yang rapi, keamanan yang lebih baik, serta dukungan terhadap berbagai fitur modern yang dibutuhkan dalam pengembangan aplikasi *web*. Penelitian yang dilakukan akan berfokus pada implementasi REST API dalam mengembangkan aplikasi backend. Sistem tersebut berbentuk aplikasi *website* yang dibuat menggunakan metode *SDLC* dengan model *waterfall*, bahasa pemprograman PHP ^8.1, *framework Laravel Lumen* 10^10, dan basis data *MySQL*. Hasil dari pengembangan aplikasi ini menunjukkan bahwa *T-Feeder* mampu meningkatkan efisiensi proses pelaporan data PDDikti, mengurangi kesalahan input data, dan mempercepat waktu yang dibutuhkan untuk pengiriman data. Dengan demikian, aplikasi ini diharapkan dapat menjadi solusi yang efektif bagi perguruan tinggi dalam memenuhi kewajiban pelaporan data ke PDDikti.

Kata Kunci: *Backend, Neo Feeder* PDDikti*, Rest* Api*, Laravel Lumen, MySQL*

ABSTRACT

The development of the T-Feeder application for Higher Education Database (PDDikti) reporting aims to increase efficiency and accuracy in managing higher education data in Indonesia. In this research, the T-Feeder application is designed to facilitate the process of reporting data to PDDikti's Neo Feeder, which includes data on students, lecturers, curriculum and other academic activities. Using Laravel as a backend provides several benefits, such as a neat code structure, better security, and support for various modern features needed in web application development. The research carried out will focus on implementing REST APIs in developing backend applications. The system is in the form of a website application created using the SDLC method with a waterfall model, PHP ^8.1 programming language, Laravel Lumen 10^10 framework, and MySQL database. The results of developing this application show that T-Feeder is able to increase the efficiency of the PDDikti data reporting process, reduce data input errors, and speed up the time required for data delivery. Thus, it is hoped that this application can be an effective solution for universities in fulfilling their data reporting obligations to PDDikti.

Keywords: Backend, Neo Feeder PDDikti, Rest Api, Laravel Lumen, MySQL

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Dalam rangka menunjang aspek keahlian profesional Fakultas ilmu Terapan Universitas Telkom telah menyediakan sarana dan prasarana penunjang pendidikan dengan lengkap, namun sarana dan prasarana tersebut hanya menunjang aspek keahlian profesional secara teori saja. Dalam dunia kerja nantinya dibutuhkan keterpaduan antara pengetahuan akan teori yang telah didapatkan dari bangku perkuliahan dan pelatihan praktik di lapang guna memberikan gambaran tentang dunia kerja yang sebenarnya.

Magang Kerja merupakan bentuk perkuliahan melalui kegiatan bekerja secara langsung di dunia kerja. Magang Kerja ini merupakan suatu kegiatan praktik bagi mahasiswa dengan tujuan mendapatkan pengalaman dari kegiatan tersebut, yang nantinya dapat digunakan untuk pengembangan profesi. Kegiatan magang kerja ini dilaksanakan di Bagian Standar dan Layanan Akademik atau yang dikenal sebagai BSLA, merupakan sebuah bagian di dalam Direktorat Akademik di bawah Koordinasi dari Wakil Rektor I Bidang Akademik Universitas Telkom yang memiliki tugas pokok Mengelola pelaporan PDDikti.

Pangkalan Data Pendidikan Tinggi (PDDikti) merupakan sebuah sistem penyimpan data yang dikelola Pusat Data dan Informasi (Pusdatin) Kementrian Ristek dan Pendidikan Tinggi (Kemristekdikti). Perguruan tinggi dapat mengolah data yang menunjukkan profil perguruan tinggi tersebut melalui sebuah antarmuka aplikasi PDDikti *FEEDER*. Kemudian data tersebut disinkronisasi dengan data yang ada di Pusdatin Kemristekdikti. Data yang tersedia di PDDikti merupakan data yang akurat, karena proses pelaporan data akademik secara berkala dilakukan dua kali setiap semester dan perkembangan akademik setiap mahasiswa dapat ditampilkan pada aplikasi Forlap yang bisa diketahui oleh masyarakat luas [1].

Peran *Backend* sangat dibutuhkan dalam pembuatan atau pengembangan aplikasi *T-Feeder*. *Backend developer* bertugas untuk membuat model dan *controller* pada aplikasi *T-Feeder*. *Controller* mengandung proses *input* dari pengguna, kemudian mengirim perintah kepada *mode* dan *view* agar melakukan proses sesuai dengan masukan. Singkatnya, *controller* merupakan pemetaan aksi pengguna terhadap respons *system*. Dan Model mengandung fungsi-fungsi yang berhubungan dengan *database*, seperti *create*, *read*, *update*, dan *delete*.

Backend adalah tempat di mana proses pada suatu *system* informasi atau aplikasi berjalan, data dapat ditambahkan, diubah maupun dihapus. *Backend* biasanya mengurus segala jenis proses yang tidak berhubungan langsung dengan pengguna, seperti *server* dan basis data. *Backend* dibutuhkan dalam pengembangan sistem dan manejemen data pada *system*.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah yang dapat diambil yaitu :

1. Bagaimana mengimplementasikan API *backend* yang bersih dan terdokumentasi dengan baik untuk mendukung integrasi dengan aplikasi klien dan layanan eksternal?
2. Bagaimana mengintegrasikan *backend* aplikasi dengan sistem eksternal seperti basis data, layanan pihak ketiga, atau sistem autentikasi?
3. Bagaimana mengelola dan menyimpan data pengguna dengan efisien untuk memastikan konsistensi data?

1.3 Tujuan

Adapun tujuan dalam melaksanakan proyek tersebut :

a. Untuk membantu terhubung nya aplikasi T-Feeder dengan portal PDDikti melaui API.
b. Untuk memudahkan perguruan tinggi dalam menyusun dan mengirimkan laporan dalam jumlah besar ke aplikasi *Neo Feeder* PDDikti.

c. Untuk memastikan bahwa data yang disampaikan oleh perguruan tinggi selalu akurat dan *up-to-date*.

1.4 Batasan Masalah

Batasan masalah dapat berisi:

1. Aplikasi *web* yang dibangun menggunakan *Framework Laravel*,
2. Laporan ini berfokus pada pengembangan sebuah aplikasi *T-Feeder* berbasis *website*,
3. Aplikasi ini berfokus pada *backend* aplikasi *T-Feeder*

1.5 Definisi Operasional

a. *Backend* merupakan suatu program yang berjalan pada sisi *server* (*server-side*) yang melakukan tugas untuk berinteraksi langsung dengan basis data dalam melakukan manipulasi data ke basis data, sehingga *backend* tidak melakukan interaksi secara langsung kepada pengguna [2].

b. Dalam melakukan pelaporan data, dibuatlah aplikasi *Feeder* PDDikti. Aplikasi tersebut merupakan aplikasi yang berfungsi menjamin kualitas, relevansi, keterjangkauan dan pemetaan dari data dan informasi Perguruan Tinggi. Penerapan aplikasi ini digunakan untuk melakukan sinkronisasi data perguruan tinggi yang memanfaatkan *web service*. *Web Service* 18 merupakan informasi-informasi yang bisa diakses oleh siapa saja dan oleh perangkat apa saja [3].

c. API merupakan antarmuka yang digunakan untuk mengakses aplikasi atau layanan dari sebuah program [4]. API memungkinkan *developer* untuk memakai fungsi yang sudah ada dari aplikasi lain sehingga tidak perlu membuat ulang dari awal. Salah satu arsitektur *backend* adalah *Representational State Transfer* (*REST*). *REST* merupakan seperangkat prinsip arsitektur yang melakukan transmisi data melalui antarmuka yang terstandardisasi seperti HTTP. *REST* API sendiri merupakan istilah yang

dipakai untuk layanan *web* yang mengimplementasikan arsitektur *REST* sebagai API [5].

d. *Laravel Lumen* atau yang biasa disebut *Lumen* merupakan sebuah *micro-framework* berbasis bahasa pemrograman PHP yang ditujukan untuk pembuatan *microservices website* dan API yang cepat [6].*Lumen* dikembangkan oleh pengembang *framework* PHP *Laravel*, yaitu *Taylor Otwell* sebagai sebuah proyek yang bertujuan untuk membuat Laravel versi ringan. *Lumen* dijadikan *framework* yang dapat membuat *REST API* dengan fiturnya yang kuat dan cepat seperti *Eloquent ORM, validation, middleware, routing*, dan lain sebagainya.

e. Pengertian *MySQL* Pada perkembangannya, *MySQL* disebut juga *SQL* yang merupakan singkatan dari *Structured Query Language*. *SQL* merupakan bahasa terstruktur yang khusus digunakan untuk mengolah *database*. *SQL* pertama kali didefinisikan oleh *American National Standards Institute (ANSI)* pada tahun 1986. *MySQL* adalah sebuah sistem manajemen database yang bersifat *open source*. *MySQL* merupakan sistem manajemen *database* yang bersifat *relational*. Artinya, data yang dikelola dalam *database* yang akan diletakkan pada beberapa tabel yang terpisah sehingga manipulasi data akan jauh lebih cepat. *MySQL* dapat digunakan untuk mengelola *database* mulai dari yang kecil sampai dengan yang sangat besar [7].

1.6 Metode Pengerjaan

Metode ini sangat tepat untuk pengembangan perangkat lunak ketika menentukan solusi optimal untuk menyelesaikan permasalahan di Bagian Standar & Layanan Akademik.

1. **Metode *Waterfall***

   Pengembangan *system* ini dilakukan dengan menggunakan metode *Waterfall*. Metode *Waterfall* adalah pendekatan dalam pengembangan perangkat lunak dengan tahapan analisis kebutuhan, perancangan,

implementasi, pengujian, dan pemeliharaan. *Waterfall* adalah salah satu metode pengembangan sistem perangkat lunak. Metode ini memiliki keunggulan yakni proses pengembangan yang terstruktur dan terorganisir dengan baik.Kemudahan dalam pemahaman struktur hingga menghasilkan perangkat lunak dapat terdokumentasi dengan baik.

Gambar 2. 1 Metode Waterfall

Metode ini memungkinkan kontrol yang ketat terhadap jadwal, dan biaya. Nantinya hal ini akan berpengaruh pada kualitas, serta memungkinkan penyelesaian satu tahap sebelum memulai lanjutannya. Penggunaan Metode *Waterfall* mengikuti pendekatan linear atau sekuensial. Di mana pengembangan perangkat lunak dilakukan dalam tahap-tahap yang terdefinisi dengan jelas dan saling terkait [8].

Berikut adalah urutan tahapan Metode Air terjun atau *Waterfall* :

1. Analisis Kebutuhan (*Requirement* Analysis)

   Tahap ini dimulai dengan memahami kebutuhan dan tujuan dari perangkat lunak yang akan dikembangkan. Tim pengembang akan mempelajari kebutuhan dan persyaratan pengguna, serta menentukan fitur-fitur dan fungsi yang diperlukan.

2. Perancangan (*Design*)

   Setelah memahami kebutuhan, tim yang menggunakan Metode *Waterfall* merancang arsitektur, desain, dan spesifikasi teknis *software*. Perancangan juga melibatkan pembuatan diagram alir dan desain antarmuka pengguna.

3. Implementasi (*Implementation*)

   Implementasi mengarah pada pembuatan kode program, dan pengujian untuk memastikan kualitas perangkat lunak yang dibangun.

4. Pengujian (*Testing*)

   Setelah kode program selesai dibuat, tahap pengujian dilakukan untuk memastikan *software* berfungsi dengan baik. Hasilnya ialah perangkat lunak yang mampu memenuhi persyaratan pengguna.

5. Pemeliharaan (*Maintenance*)

   Proses pemeliharaan baru dilaksanakan apabila produk sudah dikeluarkan oleh developer kepada konsumen. Tim pengembang akan terus memperbaiki, memperbarui, dan memperluas perangkat lunak sesuai dengan kebutuhan pengguna.Tahapan ini tidak hanya menjaga kondisi perangkat tetap berjalan baik, namun juga melakukan *upgrade* berkala. Dengan begitu tingkat kepuasan pengguna akan meningkat seiring dengan perawatan dan perbaikan yang dilakukan.

1.7 Jadwal Pengerjaan

Penjadwalan magang di Bagian Standar dan Layanan Akademik (BSLA) – Universitas Telkom dimulai dari tanggal 5 February 2024 – 5 Juni 2024 dengan jadwal 3 hari kerja secara offline di Ruang Layanan BSLA, Gedung Rektorat/Gedung Bangkit Lt.1 . Adapun rincian pelaksanaan praktik kerja magang yang dilaksanakan oleh penulis sebagai berikut:

Tabel 1. 1 Pelaksanaan Kerja

No	Deskripsi Kerja	Feb				Mar				Apr				Mei				Jun				Jul				Agu			
		1	2	1	1	2	3	4	2	3	4	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Preparation																												
2	Planing																												
3	Design																												
4	Develop																												
5	Testing																												

BAB 2
LATAR BELAKANG

2.1 Penelitian Terdahulu

Berikut penelitian terdahulu yang diambil sebagai referensi pengembangan proyek akhir ini :

Tabel 1. 2 Penelitian Terdahulu

No.	Judul Penilitian	Penulis	Tahun	Persamaan	Perbedaan
1.	IMPLEMENTASI REST API PADA PENGEMBANGAN APLIKASI BACKEND UNTUK PLATFORM KURSUS ONLINE (GROWUP)	Rizqy Eka Putra Rizaldy [9]	2022	Kedua nya menggunakan teknologi *REST API* untuk menghubungkan dan mengintegrasikan data antara aplikasi yang berbeda.	GrowUp menerapkan metode *FCP (Fundamental, Conceptual, dan Practical*) dalam pengembangan *platform* mereka, sementara *T-Feeder* menggunakan model *SDLC* dengan pendekatan *waterfall* dalam pengembangan aplikasi mereka.
2.	PENGEMBANGAN BACKEND APLIKASI LAYANAN INTEGRASI LOGISTIK BERBASIS WEB	Muhammad Raga Devara Kusumo Widiyanto [10]	2023	Kedua nya menggunakan *framework Laravel* sebagai *backend*.	*Website* logistik mengelola data terkait pengiriman barang dan operasi logistik, sementara *T-*

	MENGGUNAKAN LARAVEL				*Feeder* mengelola data pendidikan tinggi, termasuk data mahasiswa, dosen, kurikulum, dan aktivitas akademik.
3.	PENERAPAN PRINSIP-PRINSIP RESTFUL API DALAM PENGEMBANGAN WEB SERVICE PADA NEO FEEDER DENGAN FRAMEWORK LARAVEL	M. Izul Ula, Kusrini Kusrini, Alva Hendi Muhammad [11]	2024	Kedua sistem berhubungan dengan pengelolaan data yang terintegrasi dengan *Neo Feeder* PDDikti.	Pada pengembangan *web service* lebih umum dalam cakupan pengembangan dan penggunaan *RESTful Web Service* untuk pengembangan data di *Neo Feeder,* sedangkan pengembangan aplikasi *T-Feeder* lebih spesifik untuk pelaporan data pendidikan tinggi dan integrasi dengan *Neo Feeder* PDDikti.

Berdasarkan Tabel 1.2, ketiga penelitian terdahulu tersebut terdapat kesamaan yaitu teknologi yang digunakan, *framework* yang digunakan, dan berhubungan dengan pengelolaan data yang terintegrasi dengan *Neo Feeder* PDDikti. Namun memiliki perbedaan pada metode, jenis data yang dikelola, dan ruang lingkup yang digunakan.

2.2 Profil Perusahaan

BAA yang kemudian menjadi BSLA dalam hal ini dikenal dengan Bagian Standar & Layanan Akademik berwenang mengelola layanan administrasi akademik yang merupakan bagian dari Direktorat Akademik Universitas Telkom [12].

Gambar 2. 2 Logo Bagian Standar & Layanan Akademik

Sumber : baa.telkomuniversity.ac.id

Bagian Standar & Layanan Akademik merupakan sebuah bagian di dalam Direktorat Akademik di bawah Koordinasi dari Wakil Rektor I Bidang Akademik Universitas Telkom yang memiliki tugas pokok :

a. Mengelola layanan administrasi akademik level Universitas seperti penerbitan Dokumen kelulusan (ijazah digital : *https://basila.telkomuniversity.ac.id/*),
b. Mengelola fasilitas yang menunjang kegiatan operasional akademik untuk perkuliahan bersama,
c. Berkoordinasi dengan Direktorat Sekretariat dan Perencanaan Strategis (SPS) dalam hal pelaksanaan wisuda.
d. Mengelola pelaporan PDDIKTI,
e. Mengelola rencana implernentasi, *monitoring*, dan evaluasi program kerja dengan pihak *internal* (mahasiswa dan Wakil Dekan Bidang Akademik dan Dukungan Penelitian) serta ekstemal (L2Dikti dan alumni).

Gambar 2. 3 Struktur Organisasi BSLA

Sumber : baa.telkomuniversity.ac.id

2.3 *Backend Developer*

Backend adalah tempat untuk memproses suatu sistem yang berjalan dimana, pada *backend* ini data dapat diproses, ditambahkan, diubah atau dihapus [13]. *Backend* sering juga disebut sebagai *server side*. Proses yang tidak dilihat atau yang tidak berinteraksi langsung dengan *user* berada pada *backend* seperti *database*, server, keamanan, dan sebagainya. *Backend* adalah bagian dibalik layar dari suatu *software*. Bahasa pemrograman yang digunakan oleh seorang *backend developer* adalah PHP, *ruby*, *python*, dan bahasa lainnya [14].

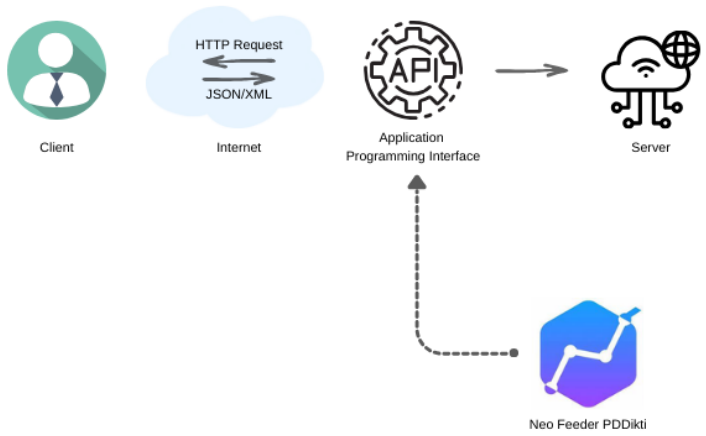
Backend bertujuan untuk mendukung *frontend* agar dapat bekerja sebagaimana mestinya. Seorang *backend developer* dibutuhkan dalam pengembangan suatu sistem atau aplikasi yang mempunyai data yang selalu berubah-ubah. Terdapat beberapa kriteria atau ciri-ciri pada *backend software* yang baik.

2.4 *Neo Feeder* PDDikti

PDDikti merupakan sistem yang digunakan untuk menyimpan data yang dikelola oleh Pusat Data dan Informasi Kementrian Ristek dan Pendidikan Tinggi. Data-data pada perguruan tinggi sudah termasuk data yang berasal dari proses pelaporan berkala dari Universitas yang dilakukan setiap dua kali di tiap semesternya. Dalam melakukan pelaporan data tersebut, dibuatlah aplikasi *Feeder* PDDikti. Aplikasi tersebut merupakan aplikasi yang berfungsi menjamin kualitas, relevansi,

keterjangkauan dan pemetaan dari data dan informasi Perguruan Tinggi. Penerapan aplikasi ini digunakan untuk melakukan sinkronisasi data perguruan tinggi yang memanfaatkan *web service*. *Web Service* 18 merupakan informasi-informasi yang bisa diakses oleh siapa saja dan oleh perangkat apa saja [3].

Teknologi *web service* ini bertujuan supaya *website* yang telah dibuat mampu mengakses layanan yang ada di *web service* yang terdiri dari *REST* dan API dimana *REST* artinya arsitektur pada *web service* yang sifatnya *client* server dengan klien akan melakukan HTTP *request* kepada server yang nantinya server akan melakukan pemrosesan *request* dan dikembalikan melalui responsenya. Ketika menggunakan teknologi *REST* terdapat pula *tools* yang bernama API berbasis *website*. API digunakan untuk menyediakan data dan klien yang dapat melakukan *request* data berupa JSON/XML [15] seperti pada Gambar 2.4



Gambar 2. 4 Alur Neo Feeder PDDikti

2.5 *REST* API

API merupakan antarmuka yang digunakan untuk mengakses aplikasi atau layanan dari sebuah program [4]. API memungkinkan *developer* untuk memakai fungsi yang sudah ada dari aplikasi lain sehingga tidak perlu membuat ulang dari awal. Salah satu arsitektur *backend* adalah *Representational State Transfer (REST). REST* merupakan seperangkat prinsip arsitektur yang melakukan transmisi data melalui antarmuka yang terstandardisasi seperti HTTP. *REST* API sendiri merupakan istilah yang dipakai untuk layanan *web* yang mengimplementasikan arsitektur *REST* sebagai API [5].

Terdapat berbagai macam method HTTP yang bisa digunakan pada *REST* API, *method* yang paling umum digunakan adalah sebagai berikut:

- *GET, method* ini digunakan untuk mengambil data dari *REST* server.
- *POST, method* ini digunakan untuk membuat data baru pada *REST* server.
- *PUT, method* ini digunakan untuk memperbarui data di dalam *REST* server.
- *DELETE, method* ini digunakan untuk menghapus data yang ada pada *REST* server.

2.6 Laravel Lumen

Laravel Lumen atau yang biasa disebut *Lumen* merupakan sebuah *micro-framework* berbasis bahasa pemrograman PHP yang ditujukan untuk pembuatan *microservices website* dan API yang cepat [6].*Lumen* dikembangkan oleh pengembang *framework* PHP *Laravel*, yaitu Taylor Otwell sebagai sebuah proyek yang bertujuan untuk membuat *Laravel* versi ringan. *Lumen* dijadikan *framework* yang dapat membuat *REST* API dengan fiturnya yang kuat dan cepat seperti *Eloquent ORM, validation, middleware, routing*, dan lain sebagainya.

Dengan memangkas beberapa *library* dari *Laravel*, *micro-framework Lumen* yang bersifat *open-source* ini memiliki ukuran yang lebih kecil dibandingkan dengan *framework Laravel*, sehingga *Lumen* diklaim menjadi salah satu *micro-framework* tercepat yang pernah ada. Berdasarkan dari *benchmarking test* oleh *Taylor Otwell* sendiri, *Lumen* bisa menangani lebih dari 1.700 *requests* per *second*-nya, lebih cepat dibandingkan dengan *micro-framework* PHP lainnya seperti *Slim* dan *Silex* [16].

Lumen memiliki keunggulan lain yaitu mempunyai *code base* yang memiliki banyak kesamaan dengan *Laravel*, sehingga *Lumen* memiliki lebih banyak dokumentasi dibandingkan dengan *micro-framewrok* PHP lainnya. Dengan berbagai fitur dan keunggulan tersebut, *Lumen* memenuhi kebutuhan pengembangan *serviceTeknologi.id Event* dan dijadikan sebagai *framework* utamanya.

2.7 MySQL

Pengertian *MySQL* Pada perkembangannya, *MySQL* disebut juga *SQL* yang merupakan singkatan dari *Structured Query Language*. *SQL* merupakan bahasa terstruktur yang khusus digunakan untuk mengolah *database*. *SQL* pertama kali didefinisikan oleh *American National Standards Institute (ANSI)* pada tahun 1986. *MySQL* adalah sebuah sistem manajemen *database* yang bersifat *open source*. *MySQL* merupakan sistem manajemen *database* yang bersifat relational. Artinya, data yang dikelola dalam *database* yang akan diletakkan pada beberapa tabel yang terpisah sehingga manipulasi data akan jauh lebih cepat. *MySQL* dapat digunakan untuk mengelola *database* mulai dari yang kecil sampai dengan yang sangat besar [7].

2.8 Postman

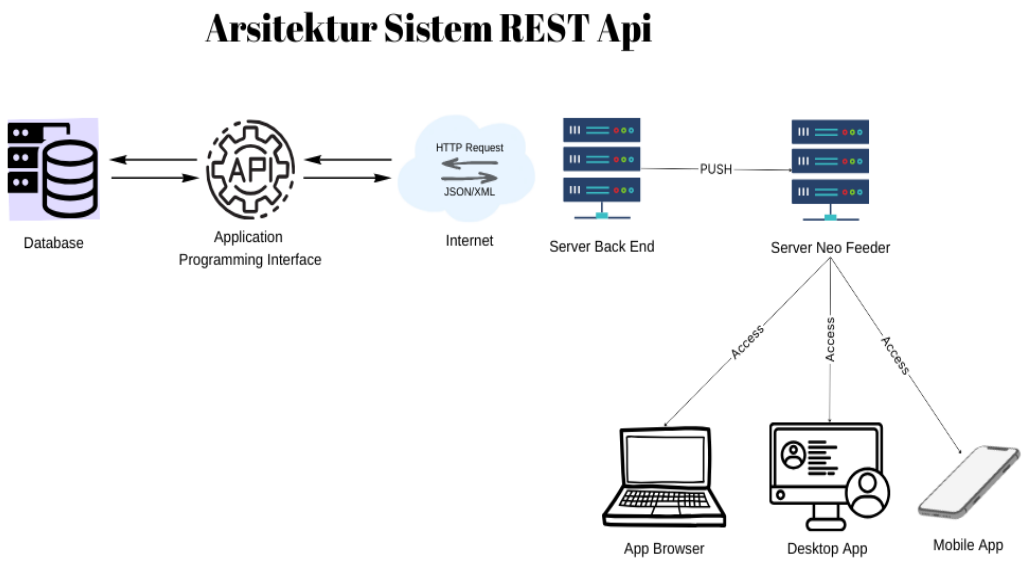
Postman adalah aplikasi (berupa *plugin* untuk *browser chrome*, yang berfungsi sebagai *REST Client* atau dengan kata lain adalah aplikasi yang berfungsi untuk melakukan uji coba *REST* API yang telah dibuat (Pranawa) [17]. Definisi lainnya *Postman* adalah alat yang disediakan oleh *Google Chrome* dengan fungsi utama sebagai *GUI* API *caller* namun sekarang juga menyediakan fitur lain yaitu *Sharing Collection API for Documentation*, *Testing API, Realtime Collaboration Team*, dan lain-lain [18]. *Postman* juga adalah aplikasi/*tools* yang disediakan oleh *Chrome* dan hanya bisa berjalan di *browser Chrome* dimana berfungsi untuk melakukan uji coba *REST* API yang telah dibuat.

BAB 3
ANALISIS DAN PERANCANGAN

3.1 Gambaran Sistem Saat Ini

Platform *website T-FEEDER* ini memiliki potensi besar untuk menjadi sarana yang efektif bagi para pengguna yang memerlukan efesiensi untuk melaporkan data. Namun, dalam proses pembuatan dan pengembangan ini masih ada beberapa aspek yang memerlukan perbaikan dan peningkatan untuk memastikan pengguna mendapatkan pengalaman yang memuaskan saat menggunakan *website T-FEEDER* ini.

Tugas magang yang diselesaikan sebagai *Backend Developer* dalam mengembangkan perangkat lunak pada bagain *backend* yang memastikan bahwa fungsionalitas sudah sesuai dengan desain yang dibuat oleh *staff Frontend*.



Gambar 2. 5 Arsitektur sistem REST Api

Berdasarkan gambar, dapat dilihat bahwa untuk memperoleh data dari *database*, *backend* akan mengirimkan *request* data ke *database* dengan menggunakan metode *CURL*. *Request* kemudian diterima oleh *REST* Api yang melakukan tindak lanjut sesuai dengan *request* dari *backend*. *REST* Api kemudian mengambil data

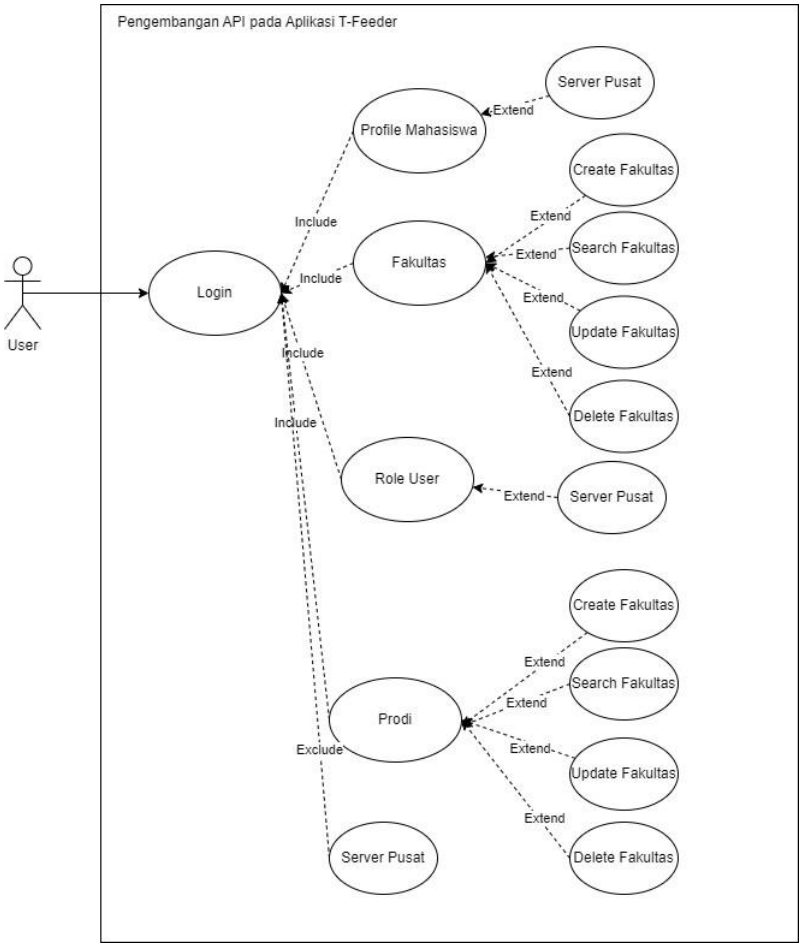
existing dari *database*. Data yang telah diterima kemudian diproses oleh subprogram yang telah dibuat sesuai *request*, yang kemudian akan di *PUSH* dari *server backend* ke *server Neo Feeder*. Dan terakhir bisa diakses dengan *App browser*. Jika permintaan *backend* hanya untuk melihat data yang ada, maka *REST* API mengirimkan *response* ke *backend* tanpa mengubah data pada *database*. Jika layanan yang diminta oleh *backend* kemudian menghasilkan perubahan terhadap data sebelumnya pada *database*, hasil *query* data pada *REST* API dikembalikan ke *database* untuk memperbarui data yang sebelumnya. Kemudian *response* dari *request backend* dikembalikan dalam format *JSON*.

Penerapan *Lumen* untuk pembuatan fungsi-fungsi *logic*. Pada proses ini fungsi-fungsi *logic* dibuat agar *system* dapat bekerja secara fungsional. Dan penerapan *Lumen* untuk pembuatan *REST* Api. Pada proses ini, fungsi-fungsi *logic* dan data-data akan dikemas ke dalam API yang berisikan *response*.

3.2 Analisis Kebutuhan Sistem

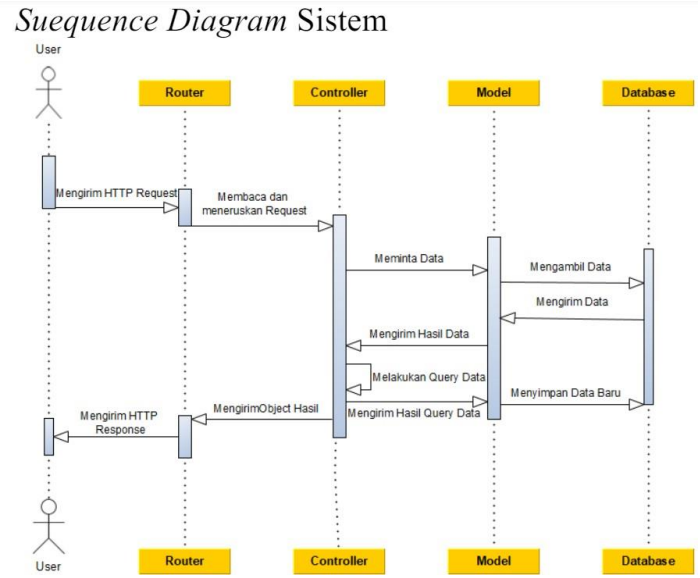
Berdasarkan pengembangan Aplikasi, maka pada analisis kebutuhan *system* usulan dibutuhkan *use case* diagram, *class* diagram, *sequence* diagram.

3.2.1 Diagram Sistem



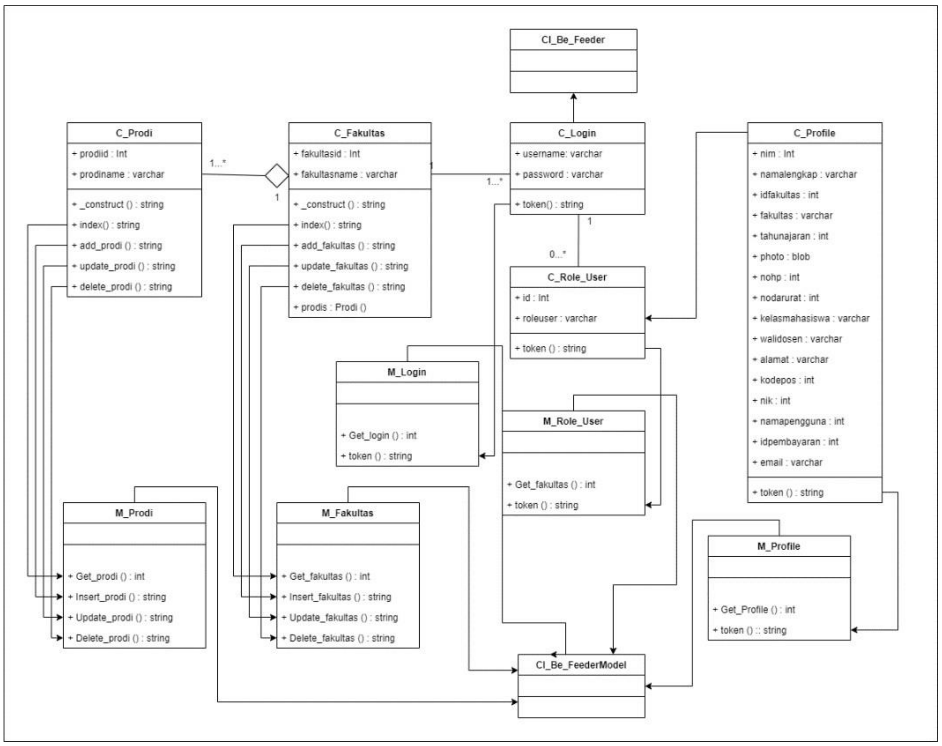
Gambar 2. 6 Use Case Diagram

Berdasarkan hasil identifikasi *actor* dan identifikasi kebutuhan sistem, dibuat *Use Case* diagram yang menggambarkan hubungan antara actor dengan sistem. *Use Case* diagram sistem dapat dilihat pada Gambar 2.7. Berdasarkan *Use Case* diagram yang telah dibuat, maka dibuat *Use Case* scenario yang menerangkan urutan sederhana ketika aktor berinteraksi dengan sistem. Di tahap ini terdapat 16 *Use Case* scenario dari masing – masing kebutuhan yang dijelaskan berdasarkan objektif, aktor, prasyarat, alur utama, dan alur alternatif.



Gambar 2. 7 Sequence Diagram

Terdapat 11 *Sequence* diagram pada pengembangan API aplikasi T-*FEEDER*. Pada Gambar 2.8 menunjukkan perancangan *Sequence* diagram yang menggambarkan urutan dari *actor User* ketika mengirimkan *Request* pada API.



Gambar 2. 8 Class Diagram

Selanjutnya dibuat *Class* diagram berdasarkan *Sequence* diagram yang telah dibuat.

Menggambarkan struktur sistem dengan menunjukkan kelas-kelas, atribut, operasi (atau metode), dan hubungan antar kelas.

3.3 Perancangan Standar Operasional Prosedur Kerja

Berikut adalah Standar Operasional Prosedur sebagai *Backend Development* di Bagian Standar & Layanan Akademik Universitas Telkom

1. Perencanaan
   a. Analisis Kebutuhan:
      - Bekerjasama dengan *stakeholder* untuk memahami kebutuhan proyek.
      - Membuat spesifikasi teknis dan fungsional.
   b. Desain Arsitektur:
      - Mendesain arsitektur aplikasi *backend* yang *scalable* dan *maintainable*.
      - Memilih teknologi dan alat yang sesuai untuk proyek seperti *Node.Js, Express, NextJS, PHP, Laravel.*
2. Pengembangan
   a. *Setup* Lingkungan Pengembangan:
      - Mengatur lingkungan pengembangan sesuai dengan *server* lokal.
      - Mengatur *system version control* (misalnya *Git*) untuk kolaborasi dan manajemen kode.
   b. Penulisan Kode:
      - Mengembangkan fitur sesuai dengan spesifikasi yang telah ditetapkan.
      - Menulis unit *test* untuk memastikan setiap komponen berfungsi dengan benar.
      - Membuat desain dan implementasi API yang efisien dan aman.

- Membuat desain dan implementasi skema *database*.
- Melakukan *code review* secara berkala untuk memastikan kualitas kode.

3. *Testing*

   a. Unit *Testing*:

   - Menjalankan unit *test* untuk memastikan fungsi individual bekerja dengan benar.

   b. Integration *Testing*:

   - Menguji integrasi antara berbagai komponen *backend* dan dengan *frontend*.

   c. *User Acceptance Testing (UAT*):

   - Mengkoordinasikan *UAT* dengan pengguna akhir untuk memastikan aplikasi memenuhi kebutuhan mereka.

4. *Deployment*

   a. *Staging Environment*:

   - Mendeploy aplikasi ke *staging environment* untuk *testing* akhir sebelum ke *production*.

   b. *Production Deployment*:

   - Mendeploy aplikasi ke *production environment* sesuai dengan prosedur yang telah ditetapkan.
   - Memastikan semua layanan berjalan dengan baik pasca *deployment*.

5. *Maintenance*

   a. *Monitoring*:

   - Memantau performa aplikasi dan *server* secara *real-time*.

- Menggunakan alat monitoring untuk mendeteksi dan mengatasi isu dengan cepat.

b. *Bug Fixing*:

- Menangani *bug* yang dilaporkan oleh tim QA atau pengguna.
- Melakukan *patching* dan *update* secara berkala untuk meningkatkan keamanan dan performa.

c. *Documentation*:

- Menulis dokumentasi teknis yang lengkap untuk setiap fitur dan modul yang dikembangkan.
- Menggunakan *tools* seperti *Postman* untuk mendokumentasikan API.
- Mengupdate dokumentasi seiring dengan perubahan pada aplikasi.

3.4 Kebutuhan Perangkat Keras dan Perangkat Lunak

Berikut adalah kebutuhan perangkat keras dan perangkat lunak selama menjadi *Back-end* di Bagian Standar & Layanan Akademik.

Tabel 1. 3 Kebutuhan Perangkat Keras dan Perangkat Lunak

Jenis Kebutuhan	Kebutuhan	Spesifikasi Server	Spesifikasi User
Perangkat Lunak	Sistem Operasi	*Windows 10*	*Windows 10*
	Web Server	*Google Chrome*	*Google Chrome*
	Framework	*Laravel ^10.10*	-
	Database	*MySQL*	-
	Language	PHP ^8.1	-
Perangkat Keras	*RAM*	4 GB	-
	Input Device	*Keyboard, Mouse*	*Keyboard, Mouse*
	Jaringan	Wifi	Wifi

BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Terdapat dua implementasi yang akan dijelaskan, yaitu implementasi data dan implementasi dalam bentuk *Source code*.

4.1.1 Implementasi Data

Dalam pengembangan aplikasi ini menggunakan *database MySQL*. Pada implementasinya, *database* yang dibuat yaitu dengan nama api_bsla, terdiri dari beberapa *table*.

id	faculties_id	studyprogram_id	studyprogramname
1	3	51	D3 Manajemen Pemasaran
2	3	32	D3 Rekayasa Perangkat Lunak
3	3	54	D3 Perhotelan
4	3	73	D3 Sistem Informasi Akuntansi
5	3	104	D3 Teknik Telekomukasi - Kampus Jakarta
6	3	72	D3 Sistem Informasi
7	3	14	D3 Teknologi Telekomunikasi
8	3	71	D3 Teknologi Komputer
9	3	33	D4 Teknologi Rekayasa Multimedia
10	4	94	S1 Desain Produk

Gambar 2. 9 Database Prodi

id	faculties_id	facultiesname
3	3	ILMU TERAPAN
4	4	INDUSTRI KREATIF
5	5	TEKNIK ELEKTRO
6	6	REKAYASA INDUSTRI
7	7	INFORMATIKA
8	8	EKONOMI DAN BISNIS
9	9	KOMUNIKASI DAN BISNIS

id	role
2	mahasiswa
3	pegawai

Gambar 2. 10 Database Fakultas & Role User

Database ini berisi tentang *table* fakultas, *role user*, dan Prodi yang ada di Universitas Telkom. *Database* adalah kumpulan data yang terorganisir dan disimpan secara sistematis, biasanya dalam bentuk *table*, untuk memudahkan pengelolaan, pencarian, dan manipulasi data. *Database* digunakan dalam berbagai aplikasi untuk menyimpan informasi seperti pengguna, produk, transaksi, dan banyak lagi.

4.1.2 Implementasi pengembangan aplikasi

routes > api.php

```php
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Api\ProfileController;
use App\Http\Controllers\Api\RoleUserController;
use App\Http\Controllers\Api\FacultiesController;
use App\Http\Controllers\Api\ProdiController;
use App\Http\Controllers\Api\ProdibyIdController;


Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});

Route::get('profile', [ProfileController::class, 'index']);
Route::post('profile', [ProfileController::class, 'store']);
Route::get('profile/{id}', [ProfileController::class, 'show']);
Route::get('profile/{id}/edit', [ProfileController::class, 'edit']);
Route::put('profile/{id}/edit', [ProfileController::class, 'update']);
Route::delete('profile/{id}/delete', [ProfileController::class, 'destroy']);

Route::get('roleuser', [RoleUserController::class, 'index']);
Route::post('roleuser', [RoleUserController::class, 'store']);
Route::get('roleuser/{id}', [RoleUserController::class, 'show']);
Route::get('roleuser/{id}/edit', [RoleUserController::class, 'edit']);
Route::put('roleuser/{id}/edit', [RoleUserController::class, 'update']);
Route::delete('roleuser/{id}/delete', [RoleUserController::class, 'destroy']);

Route::get('faculties', [FacultiesController::class, 'index']);
Route::post('faculties', [FacultiesController::class, 'store']);
Route::get('faculties/{id}', [FacultiesController::class, 'show']);
Route::get('faculties/{id}/edit', [FacultiesController::class, 'edit']);
Route::put('faculties/{id}/edit', [FacultiesController::class, 'update']);
Route::delete('faculties/{id}/delete', [FacultiesController::class, 'destroy']);

```

```php
Route::get('prodi', [ProdiController::class, 'index']);
Route::post('prodi', [ProdiController::class, 'store']);
Route::get('prodi/{id}', [ProdiController::class, 'show']);
Route::get('prodi/{id}/edit', [ProdiController::class, 'edit']);
Route::put('prodi/{id}/edit', [ProdiController::class, 'update']);
Route::delete('prodi/{id}/delete', [ProdiController::class, 'destroy']);

Route::get('studentprofile', [StudentProfileController::class, 'index']);
Route::post('studentprofile', [StudentProfileController::class, 'store']);
Route::get('studentprofile/{id}', [StudentProfileController::class, 'show']);
Route::get('studentprofile/{id}/edit', [StudentProfileController::class, 'edit']);
Route::put('studentprofile/{id}/edit', [StudentProfileController::class, 'update']);
Route::delete('studentprofile/{id}/delete', [StudentProfileController::class, 'destroy']);
```

Gambar 2. 11 Api.php routes

Source code ini berisi tentang file api.php tempat mendefinisikan *route* untuk API. *Route* yang didefinisikan di dalam file ini secara otomatis akan diprefiksikan dengan /api, yang membantu dalam pengorganisasian *route* khusus API dan membedakannya dari *route web* biasa.

```php
<?php

namespace App\Http\Controllers\Api;

use App\Models\roleuser;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;

class RoleUserController extends Controller
{
    public function index()
    {
        $roleuser = roleuser::all();
        if ($roleuser->count() > 0){
            return response()->json([
                'status' => 200,
                'roleuser' => $roleuser
            ], 200);
        }else{
            return response() ->json([
                'status' => 404,
                'message' => 'No Records Found'
            ], 404);
        }
    }

    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'role' => 'required|string|max:191'
        ]);

        if($validator->fails()){

            return response()->json([
```

```
11  {
29      {
34          if($validator->fails()){
36              return response()->json([
37                  'status' => 422,
38                  'errors' => $validator->messages()
39              ], 422);
40          }else{
41
42              $roleuser = roleuser::create([
43              'role' => $request->role
44              ]);
45
46              if($roleuser){
47
48                  return response()->json([
49                      'status' => 200,
50                      'message' => 'Role User Created Successfully'
51                  ],200);
52              }else{
53
54                  return response()->json([
55                      'status' => 500,
56                      'message' => 'Something Went Wrong!'
57                  ],500);
58              }
59          }
60
61      }
62
63      public function show($id)
64      {
65          $roleuser = roleuser::find($id);
66          if($roleuser){
67
68              return response()->json([
```

```
11  {
64      {
66          if($roleuser){
68              return response()->json([
69                  'status' => 200,
70                  'message' => $roleuser
71              ],200);
72          }else{
73
74              return response()->json([
75                  'status' => 404,
76                  'message' => 'No Such Role Found!'
77              ],404);
78          }
79      }
80
81      public function edit($id)
82      {
83          $roleuser = roleuser::find($id);
84          if($roleuser){
85
86              return response()->json([
87                  'status' => 200,
88                  'message' => $roleuser
89              ],200);
90          }else{
91
92              return response()->json([
93                  'status' => 404,
94                  'message' => 'No Such Role Found!'
95              ],404);
96          }
97      }
98
99      public function update(Request $request, int $id)
100     {
```

```
100     {
101         $validator = Validator::make($request->all(), [
102             'role' => 'required|string|max:191'
103         ]);
104 
105         if($validator->fails()){
106 
107             return response()->json([
108                 'status' => 422,
109                 'errors' => $validator->messages()
110             ], 422);
111         }else{
112 
113             $roleuser = roleuser::find($id);
114             if($roleuser){
115 
116                 $roleuser->update([
117                     'role' => $request->role
118                     ]);
119 
120                 return response()->json([
121                     'status' => 200,
122                     'message' => 'Role Updated Successfully'
123                 ],200);
124             }else{
125 
126                 return response()->json([
127                     'status' => 404,
128                     'message' => 'No Such Role Found!'
129                 ],404);
130             }
131         }
132     }
133 
134     public function destroy($id)
```

```
134     public function destroy($id)
135     {
136         $roleuser = roleuser::find($id);
137         if($roleuser){
138 
139             $roleuser->delete();
140             return response()->json([
141                 'status' => 200,
142                 'message' => 'Role Deleted Successfully'
143             ],200);
144         }else{
145 
146             return response()->json([
147                 'status' => 404,
148                 'message' => 'No Such Role Found!'
149             ],404);
150         }
151     }
152 }
153 
```

Gambar 2. 12 RoleUserController.php Api

Source code ini berisi tentang *Controller* yang digunakan untuk mengelompokkan logika yang menangani permintaan HTTP. Untuk API, *Controller* ini biasanya digunakan untuk menangani permintaan yang datang ke *route* yang didefinisikan di api.php.

app > Models > roleuser.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class roleuser extends Model
{
    use HasFactory;

    protected $fillable = [
        'role'
    ];

}
```

Gambar 2. 13 roleuser.php models

Source code ini berisi tentang direktori *app\Models* biasanya digunakan untuk menyimpan semua model *Eloquent*. Model adalah bagian penting dari arsitektur *MVC (Model-View-Controller)*, yang bertanggung jawab untuk berinteraksi dengan database. Model *Eloquent* di *Laravel* dan *Lumen* menyediakan cara yang mudah dan elegan untuk bekerja dengan *database*, termasuk melakukan *query* dan manipulasi data.

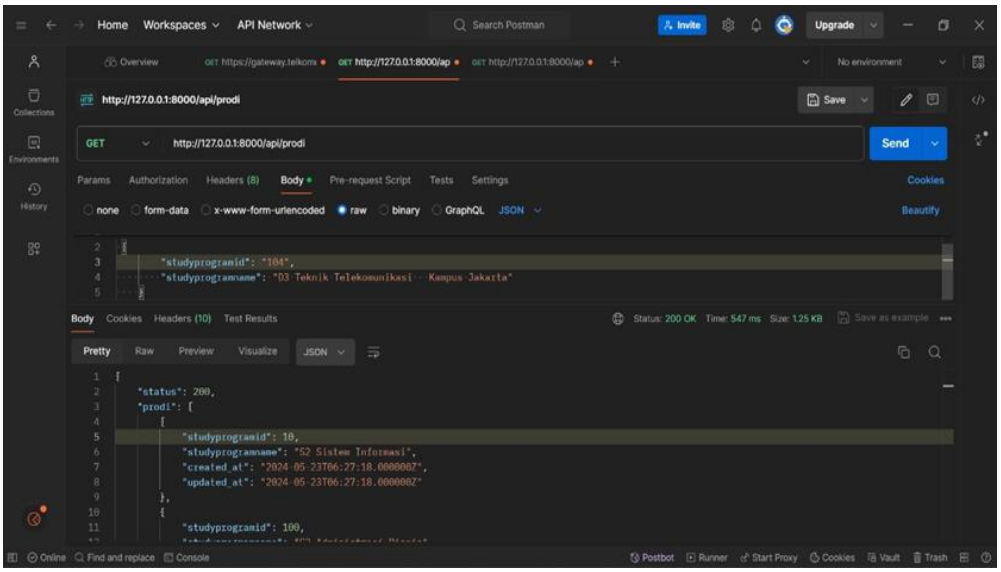
database > migrations > 2024_05_23_031800_create_roleuser_table.php

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('roleuser', function (Blueprint $table) {
            $table->id();
            $table->string('role');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('roleuser');
    }
};

```

Gambar 2. 14 Create_roleuser_table.php

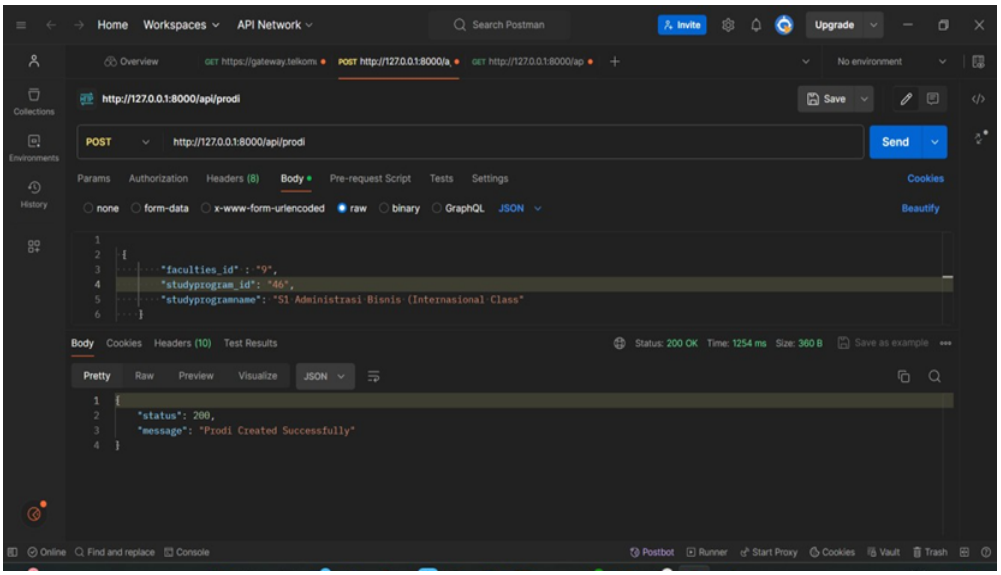
Source code ini berisi tentang Migrasi *database* di *Laravel* dan *Lumen* adalah cara untuk mengelola skema *database* Anda dengan cara yang terorganisir dan

versioned. Dengan menggunakan migrasi dapat membuat, mengubah, dan menghapus tabel *database* dengan mudah melalui perintah Artisan.



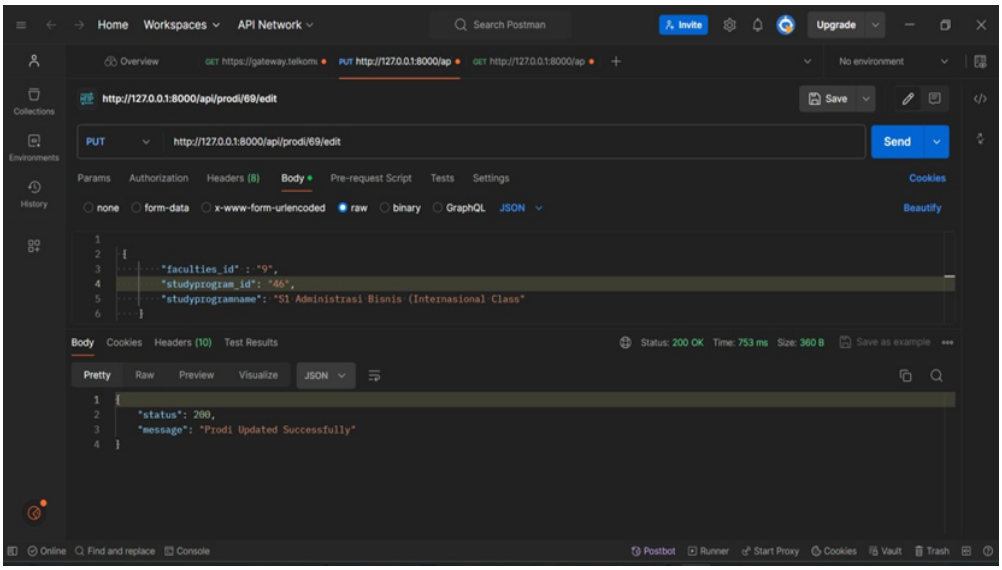
Gambar 2. 15 GET Prodi

Method Get adalah *endpoint* yang digunakan untuk mengakses data Prodi dari *database* yang disediakan oleh BSLA, dalam aplikasi *T-Feeder*.Ini memungkinkan universitas untuk mengambil data Prodi dengan mudah dan mengintegrasikannya ke dalam sistem akademik mereka.



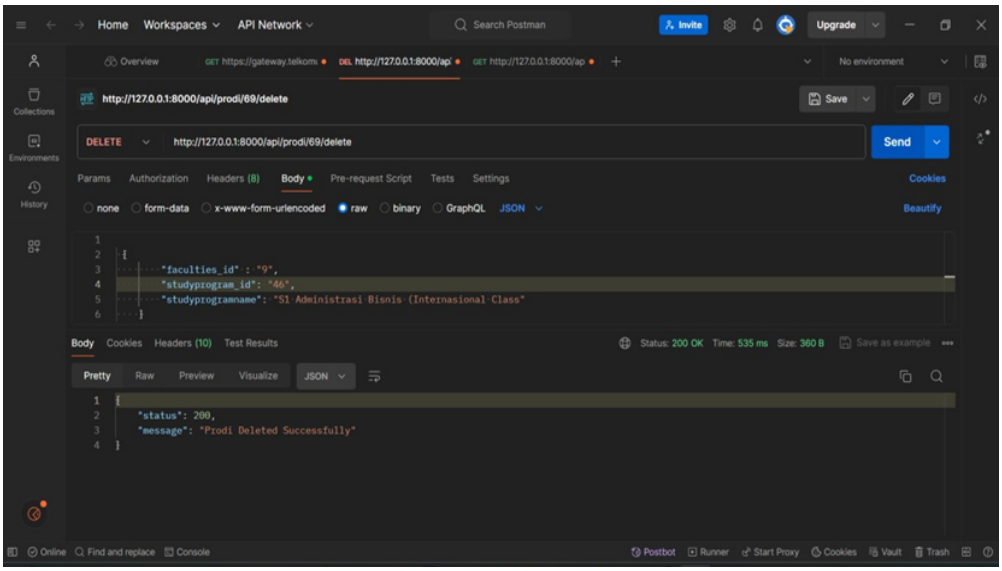
Gambar 2. 16 POST Prodi

Method POST adalah *endpoint* yang digunakan untuk mengirim data Prodi ke *server* untuk membuat atau memperbarui *database*. Data Prodi yang dikirimkan dengan *POST* biasanya ditempatkan dalam *body* dari permintaan HTTP, tidak seperti *GET* yang mengirim data sebagai bagian dari *URL*.



Gambar 2. 17 PUT Prodi

Method PUT adalah *endpoint* yang digunakan untuk memperbarui *database* yang ada atau membuat *database* baru jika tidak ada. Biasanya digunakan dalam API *RESTful* untuk memperbarui data.



Gambar 2. 18 DELETE Prodi

Method DELETE adalah *endpoint* yang digunakan untuk menghapus *database* yang ada. Biasanya digunakan dalam API *RESTful* untuk menghapus data.

4.2 Pengujian

Pengujian aplikasi ini menggunakan pengujian dengan metode *black box testing*. Metode Pengujian ini dilakukan untuk menguji fungsi yang ada pada sistem apakah *output* dari *endpoint* API sesuai dengan yang diharapkan atau tidak.

1. *Profile* Mahasiswa

Tabel 1. 4 Tabel Profile Mahasiswa

No.	*Method*	*Condition*	*Action*	*Response*
1.	*GET*	Mengambil data *Profile* dari *REST* server	*Index*	{ "status": 200, "profile": [{ "id": 1, "numberid": "670321****", "fullname": "VIENNA KHANSA *******", "studyprogramid": “7*", "studyprogram": "D3 Sistem Informasi Akuntansi", "facultyid": "3", "faculty": "ILMU TERAPAN", "schoolyear": "2122", "photo": "https://images.telkomuniversity.ac” "phone": "0851565****", "emergencyphone": "08515659****", "studentclass": "D3KA-45-03", "lectureguardian": "RENNY ****", "address": "KOMP.ESTERN HILLS REGENCY ****** “ "zipcode": "406**", "idcardnumber": "327325580*****", "user": "vienna*****", "paymentid": "670321****", "email": "viennakhansa@*****",

				"created_at": "2024-05-20T07:22:27.000000Z", "updated_at": "2024-05-20T11:54:33.000000Z" }] }

2.	*POST*	Membuat data *Profile* baru pada *REST* server	*Store*	{
				"status": 200,
				"message": "Profile Created Successfully"
				}

3.	*GET*	Mengambil data *Profile* dari *REST* server untuk ditampilkan	*Show*	{
				"status": 200,
				"profile": [
				{
				"id": 1,
				"numberid": "670321****",
				"fullname": "VIENNA KHANSA *******",
				"studyprogramid": “7*",
				"studyprogram": "D3 Sistem Informasi Akuntansi",
				"facultyid": "3",
				"faculty": "ILMU TERAPAN",
				"schoolyear": "2122",
				"photo": "https://images.telkomuniversity.ac”
				"phone": "0851565****",
				"emergencyphone": "08515659****",
				"studentclass": "D3KA-45-03",
				"lectureguardian": "RENNY ****",
				"address": "KOMP.ESTERN HILLS REGENCY ****** “
				"zipcode": "406**",
				"idcardnumber": "327325580***",
				"user": "vienna*****",
				"paymentid": "670321****",
				"email": "viennakhansa@*****",
				"created_at": "2024-05-20T07:22:27.000000Z",
				"updated_at": "2024-05-20T11:54:33.000000Z"
				}
				]
				}

4.	*GET*	Mengambil data *Profile* dari *REST* server untuk di edit	*Edit*	{
				"status": 200,
				"profile": [
				{
				"id": 1,
				"numberid": "670321****",
				"fullname": "VIENNA KHANSA *******",
				"studyprogramid": "7*",
				"studyprogram": "D3 Sistem Informasi Akuntansi",
				"facultyid": "3",
				"faculty": "ILMU TERAPAN",
				"schoolyear": "2122",
				"photo": "https://images.telkomuniversity.ac"
				"phone": "0851565****",
				"emergencyphone": "08515659****",
				"studentclass": "D3KA-45-03",
				"lectureguardian": "RENNY ****",
				"address": "KOMP.ESTERN HILLS REGENCY ****** "
				"zipcode": "406**",
				"idcardnumber": "327325580*****",
				"user": "vienna*****",
				"paymentid": "670321****",
				"email": "viennakhansa@*****",
				"created_at": "2024-05-20T07:22:27.000000Z",
				"updated_at": "2024-05-20T11:54:33.000000Z"
				}
				]
				}

5.	*PUT*	Memperbarui data *Profile* di dalam *REST* server	*Update*	{
				"status": 200,
				"message": "Profile Updated Successfully"
				}

6.	*DELETE*	Menghapus data *Profile* yang ada pada *REST* server	*Destroy*	{
				"status": 200,
				"message": "Profile Deleted Successfully"
				}

2. *Role User*

Tabel 1. 5 Tabel Role User

No.	*Method*	*Condition*	*Action*	*Response*
1.	*GET*	Mengambil data *Role User* dari *REST* server	*Index*	{ "status": 200, "roleuser": [{ "id": 2, "role": "mahasiswa", "created_at": "2024-05-23T03:59:27.000000Z", "updated_at": "2024-05-23T03:59:27.000000Z" }] }
2.	*POST*	Membuat data *Role User* baru pada *REST* server	*Store*	{ "status": 200, "message": "Role User Created Successfully" }
3.	*GET*	Mengambil data *Role User* dari *REST* server untuk ditampilkan	*Show*	{ "status": 200, "roleuser": [{ "id": 2, "role": "mahasiswa", "created_at": "2024-05-23T03:59:27.000000Z", "updated_at": "2024-05-23T03:59:27.000000Z" }] }

4.	*GET*	Mengambil data *Role User* dari *REST* server untuk di edit	*Edit*	{
				"status": 200,
				"roleuser": [
				{
				"id": 2,
				"role": "mahasiswa",
				"created_at": "2024-05-23T03:59:27.000000Z",
				"updated_at": "2024-05-23T03:59:27.000000Z"
				}
				]
				}

5.	*PUT*	Memperbarui data *Role User* di dalam *REST* server	*Update*	{
				"status": 200,
				"message": "Role User Updated Successfully"
				}

6.	*DELETE*	Menghapus data *Role User* yang ada pada *REST* server	*Destroy*	{
				"status": 200,
				"message": "Role User Deleted Successfully"
				}

3. Fakultas

Tabel 1. 6 Tabel Fakultas

No.	*Method*	*Condition*	*Action*	*Result*
1.	*GET*	Mengambil data Fakultas dari *REST* server	*Index*	{ "status": 200, "faculties": [{ "facultiesid": 3, "facultiesname": "Ilmu Terapan", "created_at": "2024-05-23T05:38:33.000000Z", "updated_at": "2024-05-23T05:38:33.000000Z" }, { "facultiesid": 4, "facultiesname": "Industri Kreatif", "created_at": "2024-05-23T05:39:46.000000Z", "updated_at": "2024-05-23T05:39:46.000000Z" }, { "facultiesid": 5, "facultiesname": "Teknik Electro", "created_at": "2024-05-23T05:40:09.000000Z", "updated_at": "2024-05-23T05:40:09.000000Z" }, { "facultiesid": 6, "facultiesname": "Rekayasa Industri", "created_at": "2024-05-23T05:41:46.000000Z", "updated_at": "2024-05-23T05:41:46.000000Z" }, { "facultiesid": 7, "facultiesname": "Informatika", "created_at": "2024-05-

				23T05:42:04.000000Z",
				"updated_at": "2024-05-23T05:42:04.000000Z"
				},
				{
				"facultiesid": 8,
				"facultiesname": "Ekonomi dan Bisnis",
				"created_at": "2024-05-23T05:42:25.000000Z",
				"updated_at": "2024-05-23T05:42:25.000000Z"
				},
				{
				"facultiesid": 9,
				"facultiesname": "Komunikasi dan Bisnis",
				"created_at": "2024-05-23T05:42:50.000000Z",
				"updated_at": "2024-05-23T05:42:50.000000Z"
				}
				]
				}

2.	*POST*	Membuat data Fakultas baru pada *REST* server	*Store*	{
				"status": 200,
				"message": "Faculties Created Successfully"
				}

3.	*GET*	Mengambil data Fakultas dari *REST* server untuk ditampilkan	*Show*	{ "status": 200, "faculties": [{ "facultiesid": 3, "facultiesname": "Ilmu Terapan", "created_at": "2024-05-23T05:38:33.000000Z", "updated_at": "2024-05-23T05:38:33.000000Z" }, { "facultiesid": 4, "facultiesname": "Industri Kreatif", "created_at": "2024-05-23T05:39:46.000000Z",

				"updated_at": "2024-05-23T05:39:46.000000Z"
				},
				{
				"facultiesid": 5, "facultiesname": "Teknik Electro",
				"created_at": "2024-05-23T05:40:09.000000Z",
				"updated_at": "2024-05-23T05:40:09.000000Z"
				},
				{
				"facultiesid": 6, "facultiesname": "Rekayasa Industri",
				"created_at": "2024-05-23T05:41:46.000000Z",
				"updated_at": "2024-05-23T05:41:46.000000Z"
				},
				{
				"facultiesid": 7, "facultiesname": "Informatika","created_at": "2024-05-23T05:42:04.000000Z",
				"updated_at": "2024-05-23T05:42:04.000000Z"
				},
				{
				"facultiesid": 8, "facultiesname": "Ekonomi dan Bisnis",
				"created_at": "2024-05-23T05:42:25.000000Z",
				"updated_at": "2024-05-23T05:42:25.000000Z"
				},
				{
				"facultiesid": 9,
				"facultiesname": "Komunikasi dan Bisnis",
				"created_at": "2024-05-23T05:42:50.000000Z",
				"updated_at": "2024-05-23T05:42:50.000000Z"
				}
				]
				}

4.	*GET*	Mengambil data Fakultas dari *REST* server untuk di edit	*Edit*	{ "status": 200, "faculties": [{ "facultiesid": 3, "facultiesname": "Ilmu Terapan", "created_at": "2024-05-23T05:38:33.000000Z", "updated_at": "2024-05-23T05:38:33.000000Z" }, { "facultiesid": 4, "facultiesname": "Industri Kreatif", "created_at": "2024-05-23T05:39:46.000000Z", "updated_at": "2024-05-23T05:39:46.000000Z" }, { "facultiesid": 5, "facultiesname": "Teknik Electro", "created_at": "2024-05-23T05:40:09.000000Z", "updated_at": "2024-05-23T05:40:09.000000Z" }, { "facultiesid": 6, "facultiesname": "Rekayasa Industri", "created_at": "2024-05-23T05:41:46.000000Z", "updated_at": "2024-05-23T05:41:46.000000Z" }, { "facultiesid": 7, "facultiesname": "Informatika", "created_at": "2024-05-23T05:42:04.000000Z", "updated_at": "2024-05-

				23T05:42:04.000000Z"
				},
				{
				"facultiesid": 8,
				"facultiesname": "Ekonomi dan Bisnis",
				"created_at": "2024-05-23T05:42:25.000000Z",
				"updated_at": "2024-05-23T05:42:25.000000Z"
				},
				{
				"facultiesid": 9,
				"facultiesname": "Komunikasi dan Bisnis",
				"created_at": "2024-05-23T05:42:50.000000Z",
				"updated_at": "2024-05-23T05:42:50.000000Z"
				}
				]
				}

5.	*PUT*	Memperbarui data Fakultas di dalam REST server	*Update*	{
				"status": 200,
				"message": "Faculties Updated Successfully"
				}

6.	*DELETE*	Menghapus data Fakultas yang ada pada *REST* server	*Destroy*	{
				"status": 200,
				"message": "Faculties Deleted Successfully"
				}

4. Prodi

Tabel 1. 7 Tabel Prodi

No.	*Method*	*Condition*	*Action*	*Result*
1.	*GET*	Mengambil data Prodi dari *REST* server	*Index*	{ "status": 200, "prodi": [{ "studyprogramid": 10, "studyprogramname": "S2 Sistem Informasi", "created_at": "2024-05-23T06:27:18.000000Z", "updated_at": "2024-05-23T06:27:18.000000Z" }, { "studyprogramid": 100, "studyprogramname": "S2 Administrasi Bisnis", "created_at": "2024-06-24T15:49:33.000000Z", "updated_at": "2024-06-24T15:49:33.000000Z" }, { "studyprogramid": 101, "studyprogramname": "S1 Teknik Sistem Energi", "created_at": "2024-06-24T15:50:01.000000Z", "updated_at": "2024-06-24T15:50:01.000000Z" }, { "studyprogramid": 102, "studyprogramname": "S1 Penyiaran Konten Digital", "created_at": "2024-06-24T15:50:27.000000Z", "updated_at": "2024-06-24T15:50:27.000000Z" }, { "studyprogramid": 103, "studyprogramname": "S1 Manajemen Bisnis Rekreasi",

				"created_at": "2024-06-24T15:50:53.000000Z",
				"updated_at": "2024-06-24T15:50:53.000000Z"
				},
				{
				"studyprogramid": 104,
				"studyprogramname": "D3 Teknik Telekomunikasi - Kampus Jakarta",
				"created_at": "2024-06-24T15:51:29.000000Z",
				"updated_at": "2024-06-24T15:51:29.000000Z"
				}
				]
				}

2.	*POST*	Membuat data Prodi baru pada *REST* server	*Store*	{
				"status": 200,
				"message": "Prodi Created Successfully"
				}

3.	*GET*	Mengambil data Prodi dari *REST* server untuk ditampilkan	*Show*	{ "status": 200, "prodi": [{ "studyprogramid": 10, "studyprogramname": "S2 Sistem Informasi", "created_at": "2024-05-23T06:27:18.000000Z", "updated_at": "2024-05-23T06:27:18.000000Z" }, { "studyprogramid": 100, "studyprogramname": "S2 Administrasi Bisnis", "created_at": "2024-06-24T15:49:33.000000Z", "updated_at": "2024-06-24T15:49:33.000000Z" }, { "studyprogramid": 101,

				"studyprogramname": "S1 Teknik Sistem Energi",
4.	*GET*	Mengambil data Prodi dari *REST* server untukdi edit	*Edit*	{ "status": 200, "prodi": [{ "studyprogramid": 10, "studyprogramname": "S2 Sistem Informasi", "created_at": "2024-05-23T06:27:18.000000Z", "updated_at": "2024-05-23T06:27:18.000000Z" }, { "studyprogramid": 100, "studyprogramname": "S2 Administrasi Bisnis", "created_at": "2024-06-24T15:49:33.000000Z", "updated_at": "2024-06-24T15:49:33.000000Z" }, { "created_at": "2024-06-24T15:50:01.000000Z", "updated_at": "2024-06-24T15:50:01.000000Z" }, { "studyprogramid": 102, "studyprogramname": "S1 Penyiaran Konten Digital", "created_at": "2024-06-24T15:50:27.000000Z", "updated_at": "2024-06-24T15:50:27.000000Z" }, { "studyprogramid": 103, "studyprogramname": "S1 Manajemen Bisnis Rekreasi", "created_at": "2024-06-24T15:50:53.000000Z", "updated_at": "2024-06-

				24T15:50:53.000000Z"
				},
				{
				"studyprogramid": 104,
				"studyprogramname": "D3 Teknik Telekomunikasi - KampusJakarta",
				"created_at": "2024-06-24T15:51:29.000000Z",
				"updated_at": "2024-06-24T15:51:29.000000Z"
				}
				]
				}
				"studyprogramid": 101,
				"studyprogramname": "S1 Teknik Sistem Energi",
				"created_at": "2024-06-24T15:50:01.000000Z",
				"updated_at": "2024-06-24T15:50:01.000000Z"
				},
				{
				"studyprogramid": 102,
				"studyprogramname": "S1 Penyiaran Konten Digital",
				"created_at": "2024-06-24T15:50:27.000000Z",
				"updated_at": "2024-06-24T15:50:27.000000Z"
				},
				{
				"studyprogramid": 103,
				"studyprogramname": "S1 Manajemen Bisnis Rekreasi",
				"created_at": "2024-06-24T15:50:53.000000Z",
				"updated_at": "2024-06-24T15:50:53.000000Z"
				},
				{
				"studyprogramid": 104,
				"studyprogramname": "D3 Teknik Telekomunikasi - Kampus Jakarta",

				"created_at": "2024-06-24T15:51:29.000000Z", "updated_at": "2024-06-24T15:51:29.000000Z" }]

5.	*PUT*	Memperbaruidata Prodi di dalam *REST* server	*Update*	{
				"status": 200,
				"message": "Prodi Updated Successfully"
				}

6.	*DELETE*	Menghapus data Prodi yang ada pada *REST* server	*Destroy*	{
				"status": 200,
				"message": "Prodi Deleted Successfully"
				}

BAB 5
KESIMPULAN

5.1 Kesimpulan

Dalam pengembangan *web service* pada *T-Feeder* dengan menggunakan *framework Laravel Lumen* versi *10^10*, penerapan prinsip-prinsip *RESTful* API telah terbukti menjadi pendekatan yang efektif dan konsisten. Dalam bab ini, telah dipelajari dan diimplementasikan prinsip-prinsip *RESTful* API yang meliputi penggunaan metode HTTP, pengaturan *URL*, representasi sumber daya, dan lainnya. Beberapa temuan utama dari penelitian ini adalah:

1. Kemudahan Penggunaan

   Penggunaan prinsip *RESTful* API memudahkan pengembang dalam mengelola permintaan dan respons HTTP. Struktur URL yang konsisten dan metode HTTP yang sesuai mempermudah komunikasi antara klien dan *server*.

2. Pengelolaan Sumber Daya

   Prinsip *RESTful* API membantu dalam pengelolaan sumber daya dengan baik. Setiap sumber daya direpresentasikan dengan baik dan dapat diakses melalui *URL* yang bersih.

3. Skalabilitas

   Pengembangan berbasis *RESTful* API memungkinkan skalabilitas yang baik. *T-Feeder* dapat dengan mudah mengakomodasi pertumbuhan dan perubahan kebutuhan dalam sistem.

4. Keamanan

   Prinsip-prinsip *RESTful* API dapat digunakan untuk mengimplementasikan tingkat keamanan yang sesuai. Autentikasi dan otorisasi dapat diatur dengan baik untuk melindungi data dan operasi yang sensitif.

5.2 Saran

Berdasarkan hasil penelitian ini, terdapat beberapa saran yang dapat diterapkan dalam pengembangan *web service* pada *T-Feeder*:

1. Pemantauan Kinerja

   Implementasikan pemantauan kinerja dan *logging* untuk memantau permintaan dan *respons RESTful* API. Ini akan membantu dalam pemecahan masalah dan peningkatan kinerja sistem.

2. Dokumentasi yang Lengkap

   Buat dokumentasi yang lengkap dan mudah dimengerti tentang *RESTful* API yang telah dikembangkan. Ini akan membantu pengguna atau pengembang yang ingin mengintegrasikan layanan ini.

3. Pengujian yang Mendalam

   Lakukan pengujian menyeluruh, termasuk pengujian unit, pengujian integrasi, dan pengujian beban untuk memastikan bahwa sistem berfungsi dengan baik dan mampu menangani jumlah pengguna yang tinggi.

4. Pemantauan Keamanan

   Terus pemantauan keamanan system untuk mengidentifikasi dan merespons potensi kerentanannya. Pastikan bahwa praktik keamanan terbaru diterapkan untuk melindungi data dan privasi pengguna.

5. Perluasan Fungsionalitas

   Pertimbangkan untuk mengembangkan lebih lanjut fungsionalitas *T-Feeder* dengan menambahkan fitur-fitur yang relevan dan bermanfaat bagi pengguna.

DAFTAR PUSTAKA

[1] T. d. P. T. R. I. K. RISET, "Sistem Informasi Manajemen Akademik Modul Pangkalan Data Perguruan Tinggi. Jakarta: Direktorat Jenderal Pendidikan Tinggi Republik Indonesia," 2017.

[2] S. N. T. I. d. K. S. Kosasi, "Perancangan Dan Pemanfaatan E-Commerce Untuk Memperluas Pasar Produk Furniture," vol. vol. 2015, pp. pp. 17-24, 2015.

[3] M. C. R. M. a. D. P. J. Simarmata, "Teknologi Informasi: Aplikasi dan Penerapannya," 2020. [Online]. Available: https://books.google.com/books?hl=id&lr=&id=dxH5DwAAQBAJ&oi=fn. [Accessed 15 Maret 2024].

[4] Q. A. a. N. P. L. S. U. Rahardja, "Pengintegrasian YII Framework Berbasis API pada Sistem Penilaian Absensi," *Sisfotenika,* vol. 8, pp. no.2 , p. 140, 2018.

[5] S. F. S. G. a. M. A. I. A. Faruqi, "Perancangan Back-End Aplikasi Rumantara Dengan Gaya Arsitektur Rest Menggunakan Metode Iterative Incremental," *eProceedings Eng,* vol. 5, pp. no. 1, pp. 1411–1417, 2018.

[6] R. Fauzi, "Pengenalan lumen framework, Micro Framework Berbasis PHP," 2017. [Online]. Available: https://www.codepolitan.com/pengenalan-lumen-framework- micro-framework-berbasis-php-59f19fe6ea010/. [Accessed 15 Maret 2024].

[7] M. S. S. A. &. F. C. E. Novendri, "Aplikasi Inventaris Barang Pada Mts Nurul Islam Dumai Menggunakan Php Dan Mysql. lentera dumai," p. 10(2), 2019.

[8] MEILINAEKA, "Metode Waterfall dalam Pengembangan Perangkat Lunak," PuTI, [Online]. Available: https://it.telkomuniversity.ac.id/metode-waterfall-dalam- pengembangan-perangkat-lunak/. [Accessed 26 Juni 2024].

[9] R. E. P. Rizaldy, "IMPLEMENTASI REST API PADA PENGEMBANGAN APLIKASI BACKEND UNTUK PLATFORM KURSUS ONLINE (GROWUP)," 2022.

[10] M. R. D. K. Widiyanto, "PENGEMBANGAN BACKEND APLIKASI LAYANAN INTEGRASI LOGISTIK BERBASIS WEB MENGGUNAKAN," 2023.

[11] K. K. A. H. M. M.Izul Ula, "PENERAPAN PRINSIP-PRINSIP RESTFUL API DALAM PENGEMBANGAN WEB SERVICE PADA NEO FEEDER DENGAN FRAMEWORK

LARAVEL," 2024.

[12] baa.telkomuniversity.ac.id, "Organisasi," [Online]. Available: https://baa.telkomuniversity.ac.id/organisasi/. [Accessed 26 Maret 2024].

[13] P. P. Arhandi, "Pengembangan Sistem Informasi Perijinan Tenaga Kesehatan," *Teknologi Informasi,* vol. 7(1), pp. 39-48, 2016.

[14] I. H. &. R. F. Kurniawan, "REST API Menggunakan NodeJS pada Aplikasi Transaksi Jasa Elektronik Berbasis Android. JITSI : Jurnal Ilmiah Teknologi Sistem Informasi," vol. 1(4), pp. 127-132, 2020.

[15] I. A. …. B. R.-T. I. d. I. W. Wardhana, "Implementasi Teknologi Restful Web Service Dalam Pengembangan Sistem Informasi Perekaman Prestasi Mahasiswa Berbasis Website," jptiik.ub.ac.id, 2020. [Online]. Available: http://j- ptiik.ub.ac.id/index.php/j-ptiik/article/view/7024. [Accessed 15 Maret 2024].

[16] R. Fauzi, "Pengenalan lumen framework, Micro Framework Berbasis PHP," 2017. [Online]. Available: https://www.codepolitan.com/pengenalan-lumen-framework- micro-framework-berbasis-php-59f19fe6ea010/. [Accessed 15 Maret 2024].

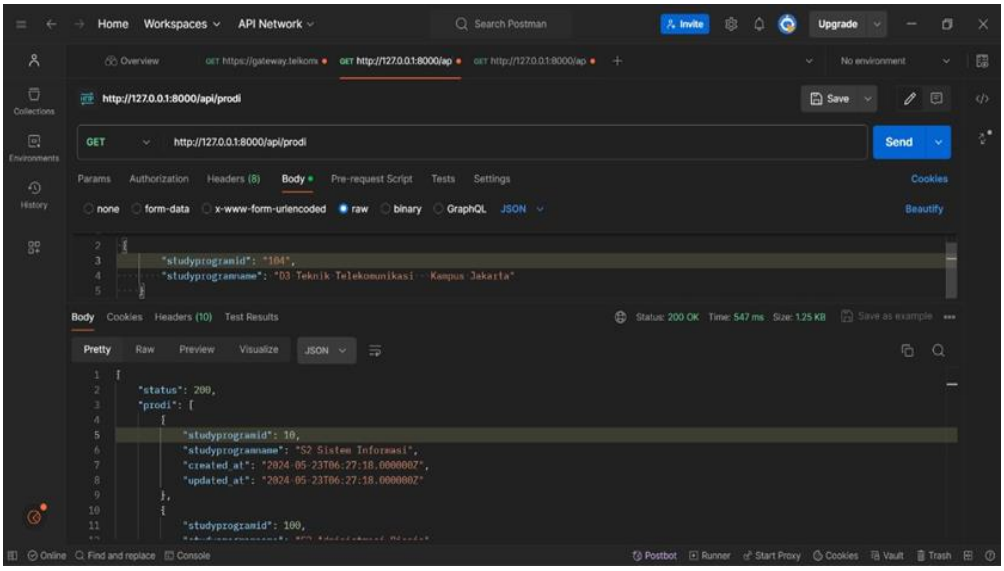
[17] M. F. P. H. &. A. H. Duskarnaen, "pascasarjana Universitas Negeri Jakarta tahun ajar 2017/2018. Jurnal PINTER," *Pengembangan web service penerimaan mahasiswa baru,* vol. 3(1), pp. 38-44, 2019.

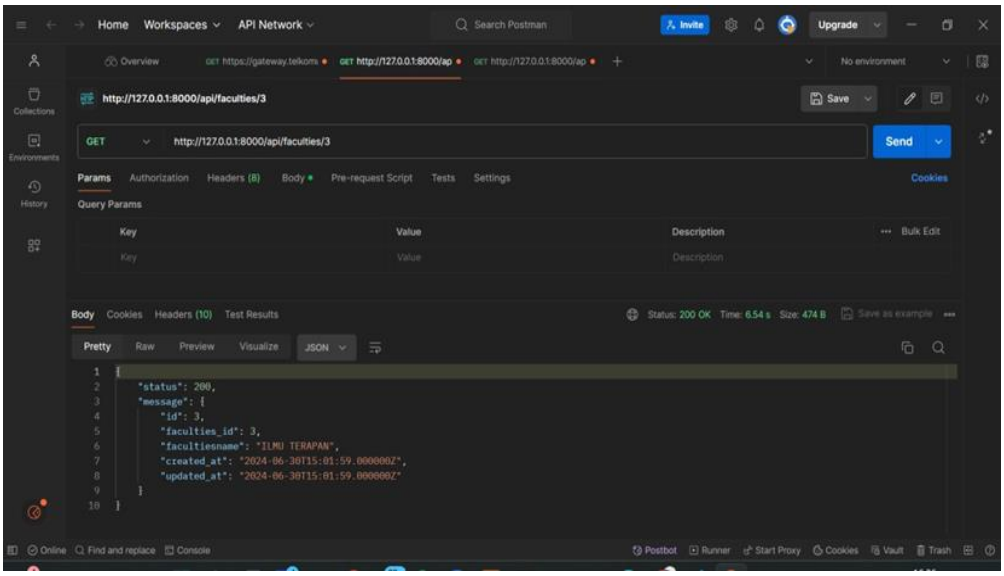
[18] F. Widyoutomo, "Universitas Negeri Jakarta [skripsi]. Jakarta: Fakultas Teknik, Universitas Negeri Jakarta.," *Pengembangan web service modul mahasiswa pada sistem informasi akademik,* 2019.

[19] K. R. D. T. KEMENTERIAN PENDIDIKAN, User Guide PDDikti Neo Feeder 2.0, 2023.

LAMPIRAN

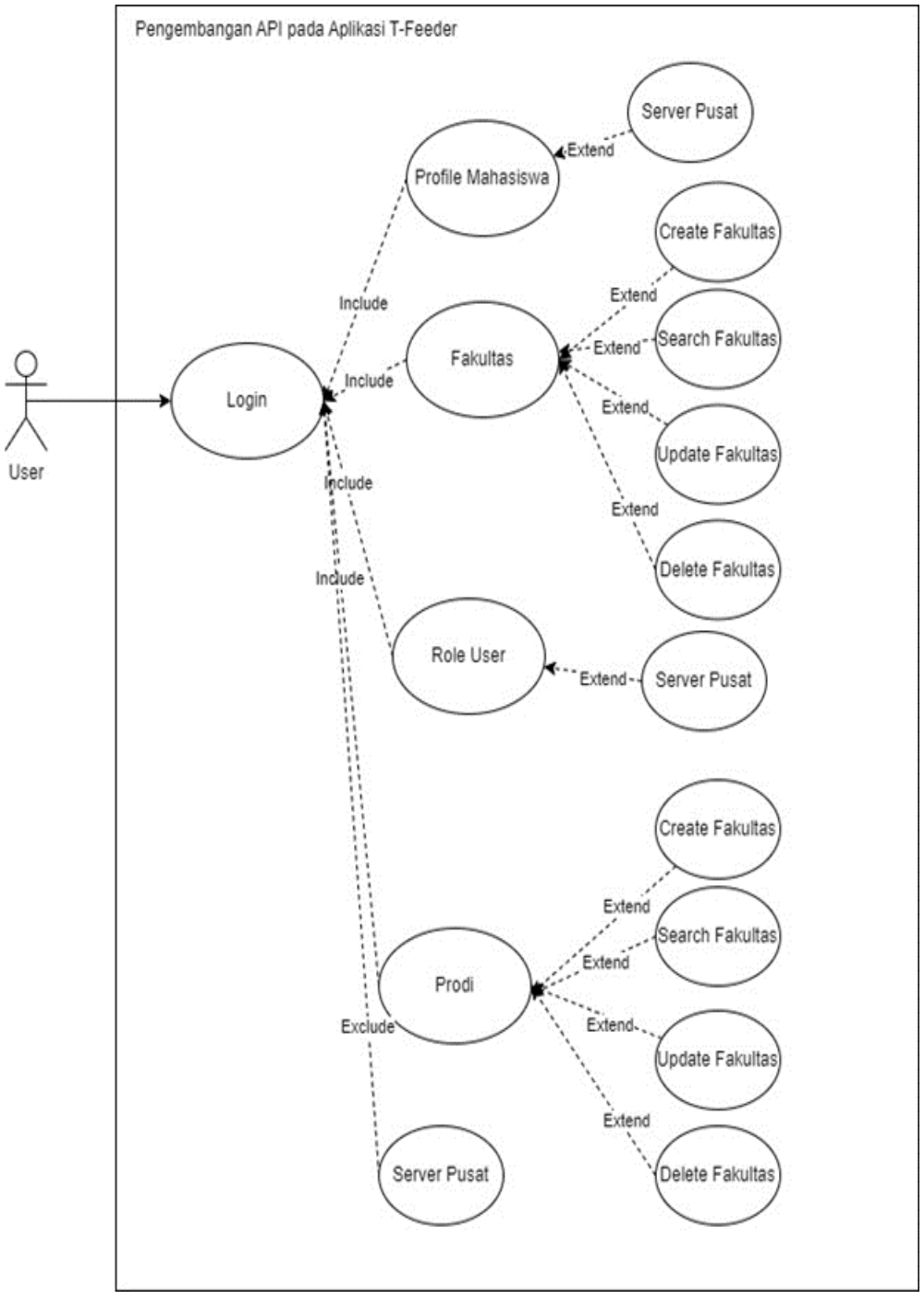
Lampiran 1
Dokumentasi Testing dengan Postman

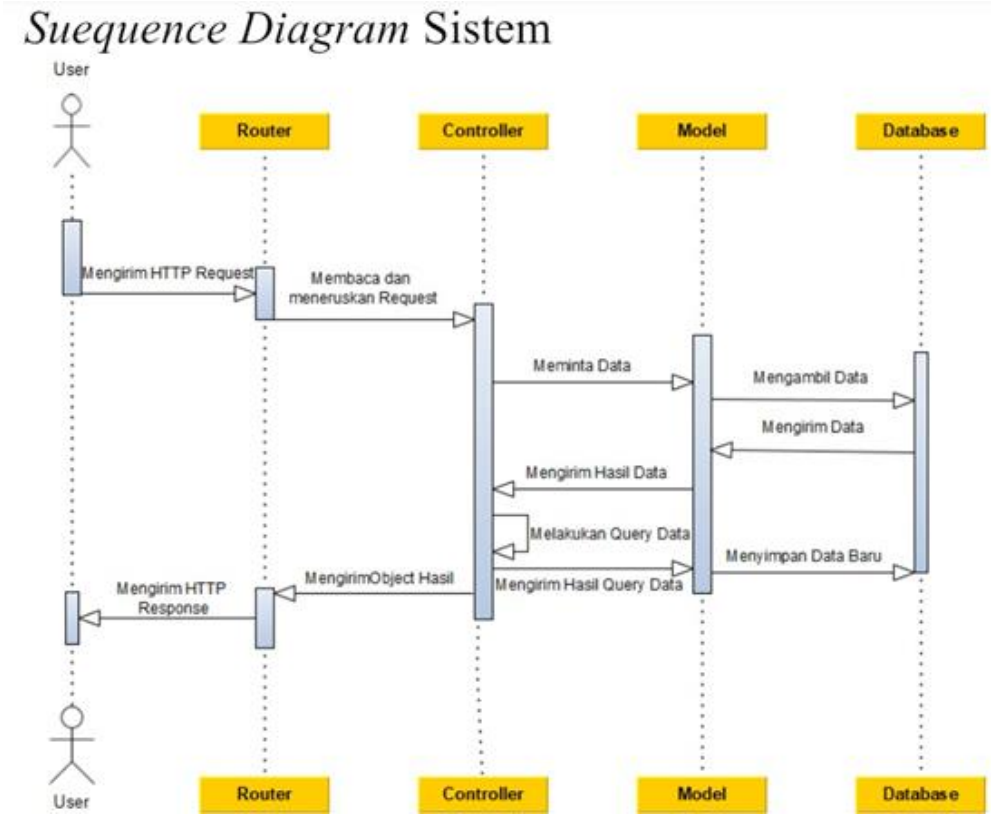


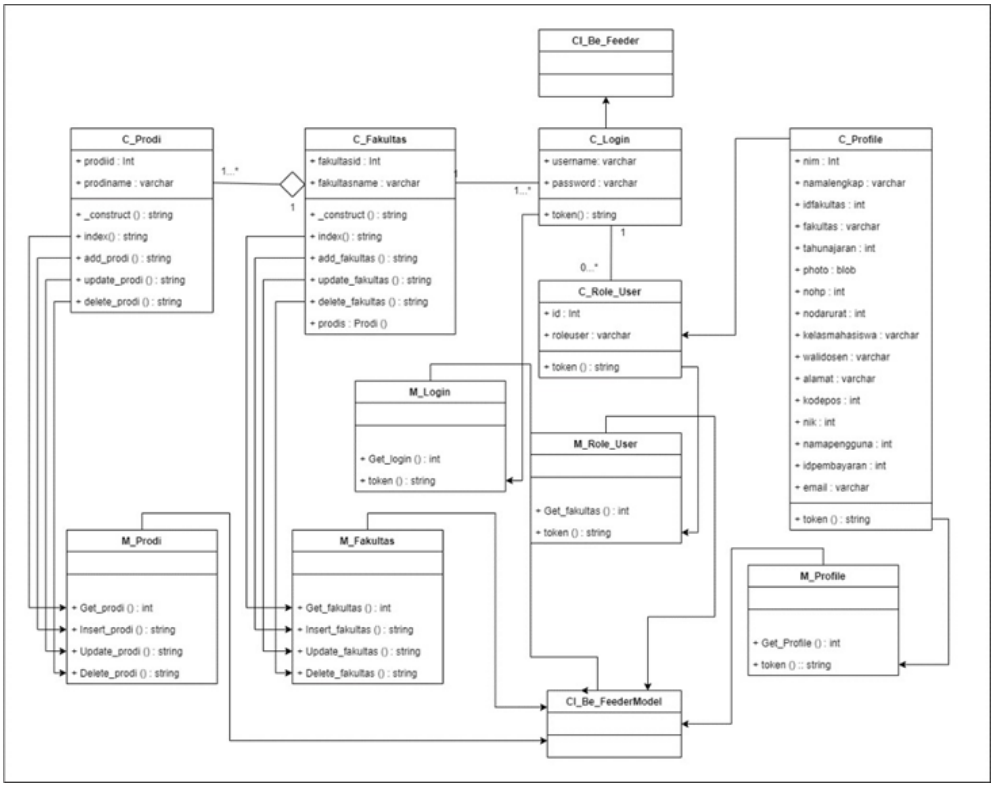


Lampiran 2 Dokumentasi Pemaparan untuk Pembuatan Aplikasi

Lampiran 3 Gambar yang Terlalu Besar




Suequence Diagram Sistem
User
Router
Controller
Model
Database
Mengirim HTTP Request
Membaca dan meneruskan Request
Meminta Data
Mengambil Data
Mengirim Data
Mengirim Hasil Data
Melakukan Query Data
Menyimpan Data Baru
Mengirim HTTP Response
MengirimObject Hasil
Mengirim Hasil Query Data
User
Router
Controller
Model
Database


CI_Be_Feeder
C_Prodi
+ prodiid : Int
+ prodiname : varchar
+ _construct () : string
+ index() : string
+ add_prodi () : string
+ update_prodi () : string
+ delete_prodi () : string
C_Fakultas
+ fakultasid : Int
+ fakultasname : varchar
+ _construct () : string
+ index() : string
+ add_fakultas () : string
+ update_fakultas () : string
+ delete_fakultas () : string
+ prodis : Prodi ()
C_Login
+ username: varchar
+ password : varchar
+ token() : string
C_Profile
+ nim : Int
+ namalengkap : varchar
+ idfakultas : int
+ fakultas : varchar
+ tahunajaran : int
+ photo : blob
+ nohp : int
+ nodarurat : int
+ kelasmahasiswa : varchar
+ walidosen : varchar
+ alamat : varchar
+ kodepos : int
+ nik : int
+ namapengguna : int
+ idpembayaran : int
+ email : varchar
+ token () : string
C_Role_User
+ id : Int
+ roleuser : varchar
+ token () : string
M_Login
+ Get_login () : int
+ token () : string
M_Role_User
+ Get_fakultas () : int
+ token () : string
M_Prodi
+ Get_prodi () : int
+ Insert_prodi () : string
+ Update_prodi () : string
+ Delete_prodi () : string
M_Fakultas
+ Get_fakultas () : int
+ Insert_fakultas () : string
+ Update_fakultas () : string
+ Delete_fakultas () : string
M_Profile
+ Get_Profile () : int
+ token () :: string
CI_Be_FeederModel