Jurnal Informatika Terpadu

https://journal.nurulfikri.ac.id/index.php/JIT
ISSN ONLINE : 2460-8998

ANALISIS DAN IMPLEMENTASI RESTFUL API GUNA PENGEMBANGAN SISTEM INFORMASI AKADEMIK PADA PERGURUAN TINGGI

Mohammad Akmaluddin Novianto[1], Sirojul Munir[2]

[1,2]Teknik Informatika, Sekolah Tinggi Teknologi Terpadu Nurul Fikri
Jakarta Selatan, DKI Jakarta, Indonesia 12640
akmalseferagic@student.nurulfikri.ac.id, rojulman@nurulfikri.co.id

Abstract

In an organization, one of the most important is the processing of data and information, with the larger and complexity of an information system, the need for data processing and integration is a big concern for many organizations. STT-NF now has started to utilize technology and develop applications to support the ease of academic community. One of the applications developed is a mobile-based academic information system application. Currently STT-NF already has an academic system called AIS with a web platform located in ais.nurulfikri.ac.id. However, there is a problem when the academic information system will be developed in STT Terpadu Nurul Fikri with a platform or with a different programming language, the problem is the unavailability of web services that are able to integrate the system with other systems to be developed, thus complicating development, especially in terms of exchange, integration and processing of data. Therefore, it is necessary to design and create the service (web service) to solve the difficulties experienced by the AIS system. This study will design and create a web service API study result card module for academic information system mobile applications in STT NF using RESTful – Spring Framework, which in addition to providing services for other systems in need, also offers the ease of bridging data exchange without any concerns about the differences in platforms and programming languages. Design web service module KHS using waterfall development method, waterfall aims to produce a software or system that has been determined with the best quality. Waterfall prioritizes interaction with stakeholders at the beginning of the project to get a clear picture, and when using the waterfall process, each team completes a stage, then the team must advance to the next stage and be expected without repeating the previous process.

Keywords: *Web Service, Restful-Spring Framework, Academic Information Systems, Service Oriented Architecture*

Abstrak

Dalam suatu organisasi salah satu yang terpenting adalah pengolahan data dan informasi, dengan semakin besar dan kompleksnya suatu sistem informasi, maka kebutuhan akan akan pengolahan dan integrasi data menjadi perhatian besar bagi banyak organisasi. STT-NF saat ini sudah mulai melakukan pemanfaatan teknologi dan mengembangkan aplikasi-aplikasi penunjang kemudahan bagi para civitas akademiknya. Salah satu aplikasi yang dikembangkan adalah aplikasi sistem informasi akademik berbasis mobile. Saat ini STT-NF sudah mempunyai sistem akademik bernama AIS dengan platform web yang beralamat di ais.nurulfikri.ac.id. Namun muncul masalah pada saat akan dikembangkan sistem informasi akademik yang ada pada STT Terpadu Nurul Fikri dengan *platform* ataupun dengan bahasa pemrograman yang berbeda, yakni belum tersedianya layanan (*web service*) yang mampu mengintegrasikan sistem tersebut dengan sistem lain yang akan dikembangkan sehingga hal ini mempersulit pengembangan terutama dalam hal pertukaran, integrasi dan pengolahan data. Oleh karena itu perlu dilakukan perancangan dan pembuatan layanan (*web service*) tersebut untuk menyelesaikan kesulitan yang dialami oleh sistem AIS. Penelitian ini akan merancang dan membuat API *web service* modul kartu hasil studi untuk aplikasi mobile sistem informasi akademik di STT NF menggunakan RESTful – Spring Framework, yang selain karena menyediakan layanan bagi sistem lain yang membutuhkan juga menawarkan kemudahan dalam menjembatani pertukaran data tanpa mempermasalahkan perbedaan platform dan bahasa pemrograman. Rancang bangun *web service* modul KHS ini menggunakan metode pengembangan *waterfall, waterfall* bertujuan untuk menghasilkan suatu *software* atau sistem yang telah ditentukan dengan kualitas terbaik. *Waterfall* mengutamakan interaksi dengan *stakeholders* di awal *project* dikerjakan untuk mendapat gambaran yang jelas, dan saat menggunakan proses *waterfall*, setiap tim menyelesaikan suatu tahap, maka tim harus lanjut ke tahap berikutnya dan diharapkan tanpa mengulang proses sebelumnya.

Kata kunci: *Web Service, RESTful-Spring Framework,* Sistem Informasi Akademik, *Service Oriented Architecture*

Diterima 19 Agustus 2021, Direvisi 13 Desember 2021, Diterima untuk publikasi 8 Maret 2022

1. PENDAHULUAN

Salah satu bagian terpenting dalam suatu organisasi adalah pengolahan data dan informasi, dengan semakin besar dan kompleksnya suatu sistem informasi, maka kebutuhan akan akan pengolahan dan integrasi data menjadi perhatian besar bagi banyak organisasi, proses-proses bisnis terus mengalami perubahan mengikuti kebutuhan organisasi, mengakibatkan diperlukan pengembangan sistem dan aplikasi yang ada [1]. Namun tantangan muncul dalam pengembangan sistem dan aplikasi yang telah tersedia sebelumnya, diantaranya adalah bagaimana data dan informasi yang lama dapat diintegrasikan dan dapat dipergunakan kembali pada pengembangan sistem dan aplikasi baru, bagaimana membangun sistem dan aplikasi yang dapat digunakan kembali di masa mendatang yang dapat berjalan baik lintas *platform*, bahasa pemrograman , maupun berbagai sistem operasi.

Hal tersebut yang mendasari konsep dan pemikiran lama akan sistem informasi khususnya mengenai arsitektur perangkat lunak yang terus berkembang. *Service Oriented Architecture* merupakan salah satu konsep arsitektur perangkat lunak yang menyediakan layanan bagi suatu sistem untuk bisa digunakan pada sistem lain sesuai kebutuhan. SOA bertujuan untuk memberikan layanan yang dapat diakses sistem lain, sehingga mendukung integrasi antar sistem [2].

Dalam mengimplementasikan SOA, *web service* dapat digunakan untuk membuat pertukaran data yang diakses melalui standar *internet protocol*. Dalam perkembangan *web service* telah dikambangan REST (*Representational State Transfer*) *web service*. Dengan mengimplementasikan REST *web service* dalam sistem maupun dengan bahasa pemrograman atau platform berbeda. *Web service* adalah standar yang digunakan untuk melakukan pertukaran data antar aplikasi atau sistem, karena aplikasi yang melakukan pertukaran data bisa ditulis dengan bahasa pemrograman yang berbeda atau berjalan pada platform yang berbeda [11]. Beberapa contoh implementasi *web service* antara lain adalah SOAP dan REST. *Web service* yang berbasis arsitektur REST kemudian dikenal sebagai RESTful *web services*, layanan ini menggunakan metode HTTP untuk menerapkan konsep arsitektur REST.

Sekolah Tinggi Teknologi Terpadu Nurul Fikri yang disingkat STT Terpadu Nurul Fikri merupakan perguruan tinggi teknologi yang memadukan antara keilmuan praktis di bidang teknologi informasi dengan pengembangan kepribadian islami. Teknologi informasi merupakan salah satu teknologi yang berkembang cepat pada saat ini. penggunaan alat bantu komputer sebagai salah satu bentuk teknologi informasi untuk menunjang sistem informasi agar dapat memberikan hasil lebih baik dan akurat untuk sebuah sistem.

STT NF saat ini sudah mulai melakukan pemanfaatan dan pendayagunaan teknologi informasi guna mewujudkan misi dari STT-NF. Oleh karena itu STT-NF mengembangkan aplikasi-aplikasi penunjang kemudahan bagi para civitas akademiknya. Salah satu aplikasi yang dikembangkan adalah aplikasi sistem informasi akademik berbasis mobile. Saat ini STT-NF sudah mempunyai sistem akademik bernama AIS dengan platform web yang beralamat di ais.nurulfikri.ac.id. tetapi kesulitan akan timbul pada saat akan dikembangkan sistem informasi akademik yang ada pada STT Terpadu Nurul Fikri tersebut dengan *platform* ataupun dengan bahasa pemrograman yang berbeda, yakni belum tersedianya layanan (*web service*) yang mampu mengintegrasikan sistem tersebut dengan sistem lain yang akan dikembangkan sehingga hal ini mempersulit pengembangan terutama dalam hal pertukaran, integrasi dan pengolahan data. Oleh karena itu perlu dilakukan perancangan dan pembuatan layanan (*web service*) tersebut untuk menyelesaikan kesulitan yang dialami oleh sistem AIS.

Penelitian ini akan merancang dan membuat API *web service* modul kartu hasil studi untuk aplikasi mobile sistem informasi akademik di STT NF menggunakan RESTful – *Spring Framework*, yang selain karena menyediakan layanan bagi sistem lain yang membutuhkan juga menawarkan kemudahan dalam menjembatani pertukaran data tanpa mempermasalahkan perbedaan platform dan bahasa pemrograman.

a. Perumusan masalah

Berdasarkan latar belakang di atas, maka rumusan masalah dalam penelitian ini antara lain:

1. Bagaimana merancang API *web service* modul KHS AIS STT Terpadu Nurul Fikri dengan RESTful – Spring Framework?
2. Apakah REST *web service* yang telah dirancang pada sistem informasi akademik berhasil diimplementasikan?

b. Tujuan dan Manfaat Penelitian

Tujuan Penelitian :

1. Mengimplementasikan REST *web service* pada *database* utama sistem informasi akademik STT Terpadu Nurul Fikri.
2. Merancang model kebutuhan dan standarisasi API pada sistem informasi akademik STT Terpadu Nurul Fikri modul KHS.

Manfaat Penelitian :

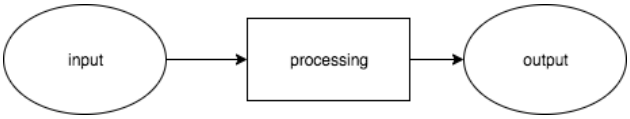
1. Memberikan kemudahan bagi *developer* dalam mengakses data sistem informasi akademik, baik pada platfrom yang sama ataupun berbeda.
2. Memberikan nilai tambah pada sistem informasi akademik STT Terpadu Nurul Fikri.

2. LANDASAN TEORI

2.1. Sistem Informasi

Sistem informasi memuat berbagai informasi penting mengenai orang, tempat, dan segala sesuatu yang ada dalam lingkungan sekitar organisasi tersebut.

Sistem informasi selalu menggambarkan, merancang, mengimplementasikan dengan menggunakan proses perkembangan sistematis dan merancang sistem informasi berdasarkan analisis kebutuhan. Jadi, bagian utama dari proses ini adalah mengetahui rancangan dan analisis sistsem. Seluruh aktivitas utama dilibatkan dalam siklus perkembangan yang lengkap [3].



Gambar 1. Konsep Dasar Sistem Informasi

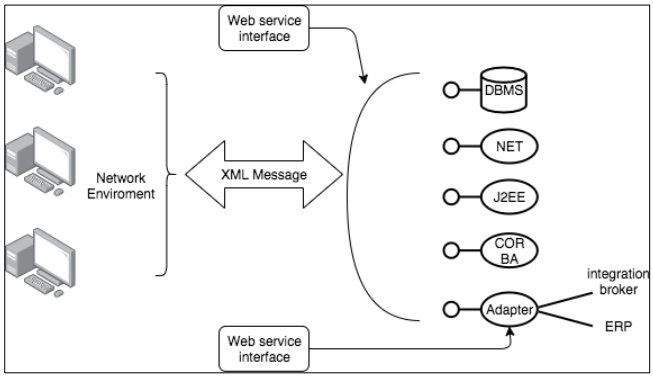
2.2. *Web Service*

2.2.1. *Web Service*

Sebuah *software* yang dirancang untuk mendukung interoperabalilitas interaksi antar mesin melalui sebuah jaringan. *Web service* secara teknis memiliki mekanisme interkasi antar sistem sebagai penunjang interoperabilitas, baik berupa agregasi (pengumpulan) maupun sindikasi dan data kolaborasi informasi yang bisa diakses melalui internet oleh berbagai pihak menggunakan teknologi yang dimiliki oleh masing-masing pengguna [4].

Alasan menggunakan *web service* adalah kemudahan dalam penggunaan kembali dan dapar *sharing* logika yang sama dengan klien yang beragam seperti *mobile*, *desktop*, dan aplikasi *web*. Jangkauan *web service* yang luas karena *web service* bergantung pada standar yang terbuka, dapat beroperasi pada platform yang berbeda, serta tidak bergantung pada teknologi eksekusi yang mendasarinya. Semua *web service* setidaknya menggunakan HTTP dan format penukaran data standar berupa XML, JSON, atau media lain.

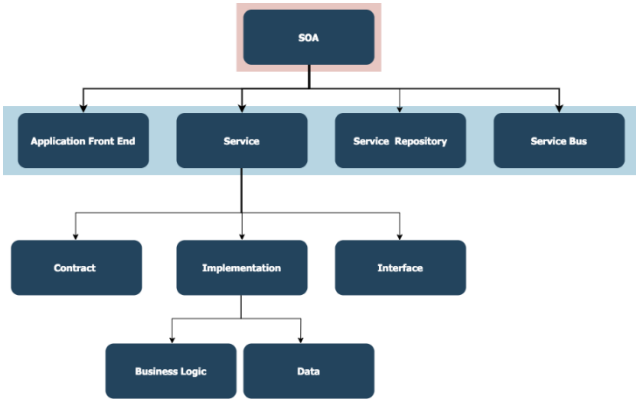
Selain itu, *web service* menggunakan HTTP dalam dua cara yang berbeda, yaitu sebagai protokol transportasi untuk menyampaikan data [5].



Gambar 2. Antarmuka *Web Service* dengan Sistem Lainnya.

2.2.2. SOA

Service Oriented Architecture atau SOA didefinisikan sebagai kebijakan, praktek, kerangka kerja yang memungkinkan fungsionalitas aplikasi disediakan dan dikonsumsi sebagai seperangkat *service* pada sebuah unit yang sesuai dengan kebutuhan *service customer*. *Service* dapat digunakan, dipublikasikan, ditemukan, dan diabstraksikan menggunakan standar antarmuka [6].



Gambar 3. Struktur Data Hirarki SOA

2.2.3. REST *Web Service*

REST merupakan singkatan dari *Representational State Transfer*. RESTful bukanlah sebuah standar protokol *web service*, melainkan hanya sebuah gaya arsitektur. ide dasar dari arsitektur REST adalah bagaimana menghubungkan jalur komunikasi antar mesin atau aplikasi melalui HTTP sederhana. Sebelum adanya REST, mekanisme atau protokol *middleware* yang cukup kompleks seperti DCE, CORBA, RPC, ataupun SOA.

REST mampu mengeksploitasi berbagai kelebihan dari HTTP yang digunakan untuk kebutuhan *web service*. Walaupun SOAP juga dapat menggunakan protokol HTTP, namun hanya terbatas untuk kebutuhan transport saja, dengan adanya REST, aplikasi *client* dapat berupa aplikasi apa saja hanya dengan memanfaatkan HTTP. Berikut ini beberapa prinsip arsitektur dari REST yang dikutip dari sebuah buku berjudul “RESTful java with JAX-RS” [7] :

1. *Addressability*

Addrressability merupakan sebuah ide dimana setiap objek *resource* pada suatu sistem dapat dicapai hanya dengan melalui sebuah *unique identifier*. Pada dunia REST, *addressability* dikelola dengan penggunaan *Uniform Resource identifier* (URI).

2. *Constrained & Uniform Interface*

Pada sistem COBRA ataupun SOAP. Pengembangan *client* harus mengetahui *method* apa saja yang disediakan oleh *web service server*. Pemanggilan *method* tersebut dikenal dengan istilah RPC (*Remote Procedure Call).*

Namun pada sistem REST, *method* atau *procedure* yang digunakan untuk layanan apapun hanyalah *method-method* yang disediakan pada HTTP. Istilah yang biasa digunakan untuk menyatakan prinsip *uniform interface* pada REST adalah CRUD (*Create, Read, Update, Delete*). Berikut adalah ulasan detail mengenai *method-method* tersebut :

Tabel 1. HTTP *Method* dan Penggunaanya dalam REST

Metode	Deskripsi
GET	Mendapatkan (read) sebuah sumber daya (*resource)* yang diidentifikasi dengan URI (*Uniform Resource Identifier*).
POST	Mengirimkan sumber daya (*resource*) ke *server*. Digunakan untuk membuat (*create*) sumber daya baru.
PUT	Mengirimkan sumber daya (*resource)* ke *server*, digunakan untuk memasukkan (*insert*) atau memperbaharui (*update*) sumber daya yang tersimpan.
DELETE	Menghapus sumber daya (*resource*) yang diidentifikasi dengan URI.
HEAD	Mendapatkan metadata (*response header) dari sumber daya* (*resource*) yang diidentifikasi dengan URI.

Dengan menggunakan protokol HTTP, URI dapat dijadikan sebagai media yang digunakan untuk mengakses *resources* dari *server*. Hal ini disebut dengan URI *tunneling.* URI *tunneling* mempergunakan URI untuk mentransfer informasi pada antar sistem yang dalam jaringan dengan melakukan *encode* pada URI itu sendiri. Dengan mengirim HTTP *method* yang telah disebutkan sebelumnya, server dapat melakukan eksekusi terhadap suatu program yang menghasilkan atau mengambil suatu *resource* dan mengirimnya kembali ke *client.* Dalam proses ini terjadi proses mapping dan URI menjadi *method call* pada *server* yang dituju [8].

3. *Stateless Communication*

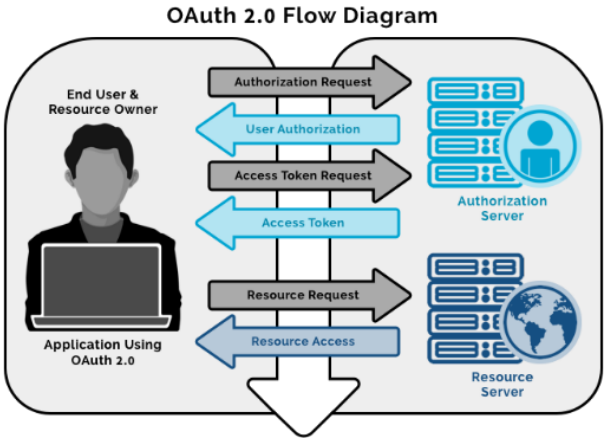
Pada dunia REST, seperti halnya pada dunia *World Wide Web*, *stateless* berarti tidak ada *client session* data yang disimpan pada *server*. *Server* hanya menyimpan dan mengelola *state* dari *resource* yang digunakan.

4. Format Pesan Pertukaran

Saat ini terdapat dua buah format pesan yang dipertukaran (*data interchange format*) yang digunakan pada *web service*, yaitu XML, dan JSON. Pada aplikasi berbasis *javascript*, penggunaan JSON sebagai format pesan pertukaran pesan akan membawa dampak performa yang cukup signifikan dibandingkan bila menggunakan *library* tambahan untuk membaca data dari *Document Object Model* (DOM).

2.2.4. Oauth 2

OAuth (Open Authorization) adalah sebuah protokol standar otorisasi standar terbuka yang memungkinkan pengguna mengakses aplikasi tanpa perlu berbagi *password*. *OAuth* juga mengizinkan otorisasi API yang sudah terlindungi yang berasal dari desktop ataupun aplikasi web melalui metode sederhana dan standard. Mengatur lalu lintas data antar aplikasi dan digunakan saat pembuat API mengetahui siapa yang terlibat dan berkomunikasi di dalam sistem. *OAuth* menyediakan kemampuan terhubung dengan sistem secara aman, sehingga pengguna tidak perlu menyerahkan *password*nya.



Gambar 4. *Oauth 2 Diagram*

2.2.5. API

API Merupakan *software interface* yang terdiri atas kumpulan instruksi yang disimpan dalam bentuk *library* dan menjelaskan bagaimana agar suatu *software* dapat berinteraksi dengan *software* lain

2.2.6. *Swagger*

Swagger adalah salah satu *tools* untuk kita bisa mendokumentasikan API secara lebih efektif, dan memungkinkan kita mendeskripsikan struktur API kita, sehingga mesin dapat membacanya.

2.3. *Spring Framework*

2.3.1. Java

Java adalah bahasa pemrograman serbaguna. Java dapat digunakan untuk membuat suatu program sebagaimana kita membuatnya dengan bahasa seperti Pascal atau C++. Yang lebih menarik, java juga mendukung sumber daya internet yang saat ini saat ini popular, yaitu *World Wide Web* atau yang sering disebut *web* saja. Java juga mendukung aplikasi klien atau server, baik dalam jaringan local (LAN) maupun jaringan berskala luas (WAN).

Program java bersifat tidak bergantung pada *platform*, artinya java dapat dijalankan pada beberapa jenis dan sistem operasi.

2.3.2. *Spring Framework*

Spring framework adalah framework *open source* yang menyediakan infrastruktur yang komprehensif dalam mengembangkan aplikasi *java* dengan mudah dan cepat. Spring akan membantu *programmer* dalam pengembangan aplikasi dengan *build* yang sederhana, *portable*, cepat dan sistem berbasis JVM yang fleksibel. Spring dapat digunakan untuk melakukan pengaturan deklarasi manajemen transaksi, *remote access* dengan menggunakan RMI atau *database*.

2.3.3. *Data Source*

Data Source adalah sumber utama dimana data berasal. Dalam pemrograman komputer, *data source* bisa berasal dari basis data, *data set*, *spreadsheet*, dan lain-lain. Ketika data ditampilkan dalam halaman web atau aplikasi apapun, data diambil dari sumber data dapat disajikan, dalam format yang sudah didefinisikan dalam pemrograman.

2.3.4. *Spring Boot*

Spring Boot merupakan sebuah *framework* Java yang digunakan untuk membuat aplikasi berbasis web dan aplikasi *enterprise*. Pada Spring Boot memberikan fleksibilitas untuk melakukan konfigurasi beans dengan berbagai cara seperti *XML*, *Annotations*, dan *JavaConfig*.

Spring Boot pada penelitian ini digunakan sebagai bagian *backend* yang digunakan untuk membuat API *web service*. Pada *Spring Boot* sendiri *layer-layer* nya jelas dan dapat digunakan untuk membuat sistem *microservice* yang r*eusable* dan s*calable*.

3. METODOLOGI PENELITIAN

3.1. Metode Pengembangan Sistem

Rancang bangun *web service* modul KHS ini menggunakan metode pengembangan *waterfall. Waterfall* mengutamakan interaksi dengan *stakeholders* di awal *project* dikerjakan untuk mendapat gambaran yang jelas, dengan menggunakan proses *waterfall*, setelah menyelesaikan suatu tahap, maka peneliti harus lanjut ke tahap berikutnya dan diharapkan tanpa mengulang proses sebelumnya.

3.2. Metode Penelitian

Metode penelitian yang penulis lakukan dalam penelitian ini menggunakan pendekatan metode kualitatif. yaitu pendeskripsian masalah yang diambil, menjelaskan langkah-langkah apa saja yang diperlukan dalam penelitian atau tahapan analisis.

3.3. Metode Pengumpulan Data

1. Studi Kepustakaan

Studi kepustakaan dilakukan dengan mencari sumber-sumber pustaka yang mendukung penelitian dan memberikan informasi yang memadai dalam menyelesaikan penelitian ini. Studi kepustakaan yang digunakan antara lain, buku, jurnal, artikel dan *paper*.

2. Observasi

Observasi dilakukan dengan pengamatan langsung terhadap objek yang akan diteliti terkait permasalahan yang akan dibahas nantinya, yaitu dengan mengidentifikasi proses kegiatan akademik dengan melihat dan mengamati bisnis proses yang terjadi antara pengguna dan sistem.

3.4. Metode Pengujian

Dalam penelitian ini, metode yang digunakan untuk pengujian *web service* yang dirancang yaitu menggunakan pengujian *Blackbox* dan Postman.

Salah satu metode *Blackbox testing* yaitu *equivalence class partitioning* adalah metode uji coba *Blackbox* yang membagi domain *input* dari program menjadi beberapa kelas data dari kasus uji coba yang dihasilkan. *Equivalence class partitioning* berusaha untuk mendefinisikan kasus uji dapat menemukan sejumlah kesalahan, kasus uji yang didesain untuk *equivalence class partitioning* berdasarkan pada evaluasi dari ekuivalensi jenis atau *class* untuk kondisi *input*. *Class-class* yang ekuivalen merepresentasikan sekumpula keadaan *valid* dan *invalid* untuk kondisi *input*. Kondisi *input* sendiri dapat berupa nilai numeric yang spesifik, kisaran nilai (*range*), sekumpulan nilai yang berhubungan (himpunan), atau kondisi *Boolean* [9].

Pengujian dengan menggunakan metode *Blackbox* berfokus pada persyaratan fungsional dari *web service* [12]. Dengan begitu, metode pengujian *Blackbox* memungkinkan *web service* mendapatkan serangkaian *input* yang sepenuhnya menggunakan persyaratan fungsional untuk *web service* yang dirancang. Pengujian API menggunakan alat bantu Postman dalam menguji fungsionalitas API dengan melakukan HTTP *request* (*GET, POST, PUT, DELETE*) Postman REST *Client* adalah salah satu aplikasi yang mendapatkan *rating* tertinggi dalam *chrome web store*. Lebih dari 348.000

pengguna unik dan lebih dari 63.000 koleksi berbagi via Postman [10].

4. ANALISIS DAN RANCANGAN

4.1. Analisis Kebutuhan

Proses anailisis kebutuhan dibutuhkan untuk menentukan apa yang dibutuhkan dalam proses pengembangan *Web Service*. Hal ini didapatkan melalui kajian dan pengamatan terkait fitur dan fungsinya yang nantinya akan diintegrasikan pada aplikasi, berikut adalah analisisnya :

1. *Web service* dapat diimplementasikan pada platform *mobile* Android dan web.
2. *Web service* dapat menjadi penghubung akses dengan *database* utama sistem.
3. *Web service server* harus terpisah dari *database* dan *web server*.
4. *Web service* harus memperhatikan bandwidth dan resource yang tersedia.

Ketentuan otorisasi (keamanan) :

1. *Web service* harus memiliki sistem keamanan (*Authorization*).
2. Sistem menyediakan fitur token dalam memberikan akses *authorization* kepada *user* yang akan mengakses *web service*.

4.1.1. Analisis Perancangan REST *Web Service*

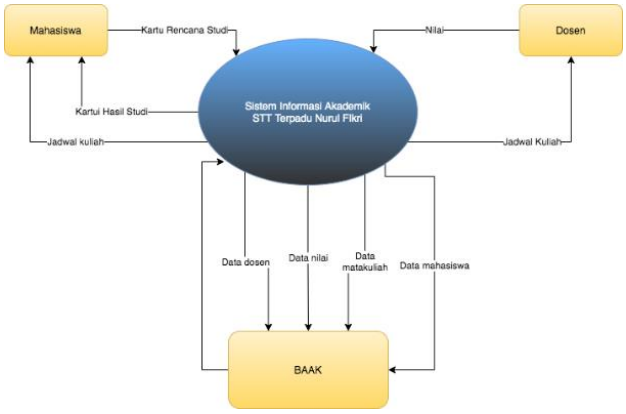
Berdasarkan hasil analisis kebutuhan yang dilakukan dengan observasi wawancara, hasilnya dapat dilihat pada tabel berikut :

Tabel 2. Hasil Analisis Perancangan *Web Service*

No.	Kebutuhan Perancangan	Kondisi Awal
1.	Basis data SIAK *postgreSQL*	Tersedia
2.	Framework untuk membangun web service	Belum tersedia
3.	Rancangan arsitektur *web service*	Belum dirancang
4.	Daftar modul *web service*	Sudah ditentukan
5.	Format data transfer yang digunakan	Sudah ditentukan
6.	Dokumentasi API	Belum tersedia

4.1.2. Analisis Sistem Berjalan

Setelah dilakukan pengamatan pada sistem informasi akademik STT Terpadu Nurul Fikri pada tahun 2021, maka terlihat sistem berjalan yang diilustrasikan melalui *data flow diagram* (DFD) berikut ini :

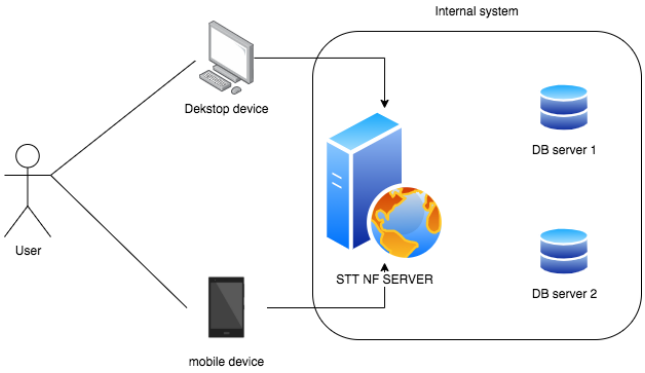


Gambar 5. Analisis Sistem Berjalan

4.1.3. Analisis Arsitektur Berjalan

Seperti yang tertulis pada landasan teori, bahwa sistem informasi akademik yang ada di STT Terpadu Nurul Fikri terdiri dari software berbasis web, yang di dalamnya ada beberapa web server dan terhubung dengan database server sebagai penyimpanan data.perangkat lunak tersebut hanya bisa diakses oleh user melalui web browser dengan jaringan internet lokal.

Jika disederhanakan dalam bentuk diagram, keadaan arsitektur berjalan pada sistem informasi akademik STT NF adalah seperti ini :



Gambar 6. Analisis Arsitektur Berjalan

4.2. Perancangan Sistem

4.2.1. Perancangan Kandidat Modul Web Service

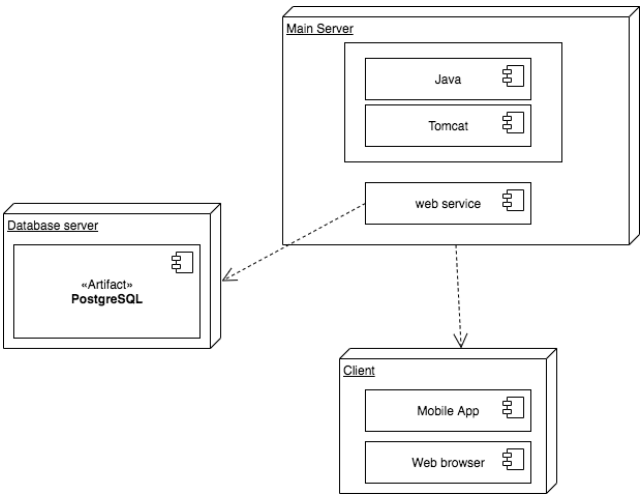
Bagian ini adalah perancangan melalui pendekatan dan prinsip-prinsip dari *SOA* untuk menghasilkan kandidat modul *web service* yang akan tersedia dan digunakan, berikut daftar kandidatnya :

Tabel 3. Kandidat *Modul Web Service*

No.	Kebutuhan	Deskripsi	User
1.	Modul Mahasiswa	Berfungsi untuk menampilkan data mahasiswa di STT Terpadu Nurul Fikri	Mahasiswa, dosen, BAAK.
2.	Modul Matakuliah	Berfungsi untuk menampilkan data yang berkaitan dengan mata kuliah yang ada di STT Terpadu Nurul Fikri.	Mahasiswa, dosen, BAAK
3.	Modul Kartu Hasil Studi	Berfungsi untuk menampilkan nilai hasil studi mahasiswa STT Terpadu Nurul Fikri	Mahasiswa, BAAK.

4.2.2. Perancangan *Deployment* Diagram

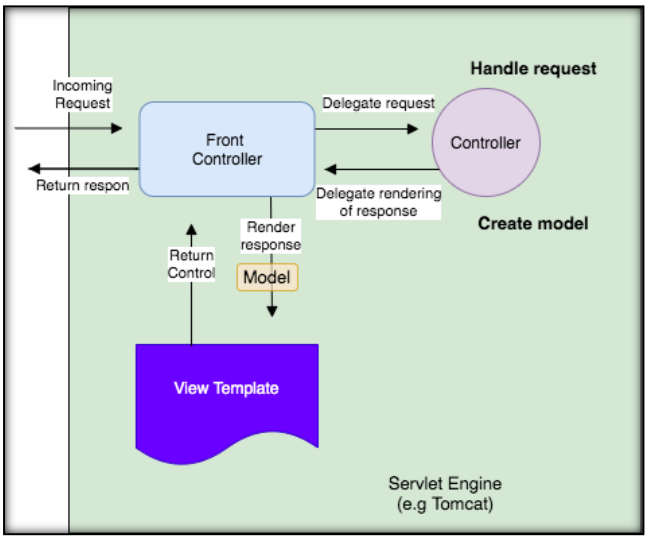
Deployment diagram dibutuhkan untuk menampakkan bagian-bagian *software* yang berjalan dalam hardware, untuk mengimplementasikan sebuah sistem dan keterhubungan antara komponen-komponen tersebut, dan menggambarkan arsitektur fisik dari aplikasi yang melibatkan seluruh perangkat yang berkaitan, baik *software* atau *hardware* yang biasanya nanti disebut dengan *Node*, dan menunjukkan bagaimana komponen ini bekerja sama akan digambarkan dalam diagram deployment, berikut adalah diagram *deployment* tersebut :



Gambar 7. *Deployment Diagram*

4.2.3. Perancangan Model *Web Service*

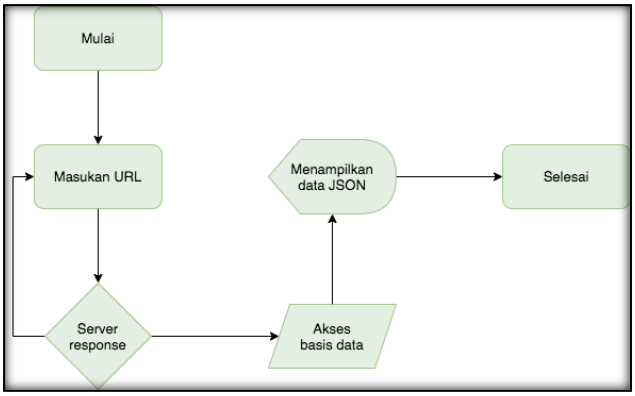
Web service sistem informasi akademik yang dirancang menggunakan Spring Framework, memiliki model seperti yang diilustrasikan pada diagram dibawah ini :



Gambar 8. *Model Web Service*

4.2.4. Perancangan Diagram Alir

Berikut adalah aliran dari operasi yang dirancang untuk REST *web service* sistem informasi akademik STT Terpadu Nurul Fikri, seperti yang ada pada gambar di bawah ini :



Gambar 9. Diagram Alir

4.2.5. Perancangan Akses

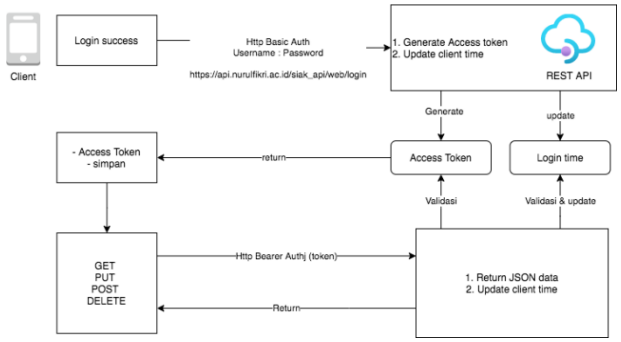
Pada bagian ini, penulis ingin membuat skema perancangan akses contoh *request* dan *respon* ketika *web service* berhasil diimplementasikan melalui tabel dibawah ini :

Tabel 4. Perancangan Akses

Method	***Request***	**Respon yang diharap**
GET	*Input* NIM dan Tahun Akademik.	Memunculkan daftar mata kuliah yang diambil pada semester tersebut.
POST	*Input* NIM dan *password.*	Berhasil login dan meng*enerate* akses token yang digunakan untuk akses *web service.*
PUT	Mengubah data mahasiswa atau nilai pada KHS.	Berhasil untuk merubah data mahasiswa atau nilai pada KHS.
DELETE	Menghapus data tertentu pada sistem.	Berhasil untuk menghapus data tertentu.

4.2.6. Perancangan Skema Otentikasi

Bagian ini menunjukkan bagaimana skema otentikasi yang akan diterapkan pada rancangan API *web service* di sistem informasi akademik STT NF menggunakan Oauth2.



Gambar 10. Skema Otentikasi Token

Pada tahap ini, client akan login terlebih dahulu untuk meminta token untuk *authorization*, kemudian *server* mengautentikasi permintaan melalui *http basic auth*, *server* lalu menggenerasikan token dan diberikan kepada *client*, lalu *client* akan mendapatkan akses untuk masuk ke resource data. Ada 4 kondisi nantinya, yang pertama yaitu berhasil untuk menampilkan data JSON, yang kedua token salah, yang ketiga token mengalami kadaluarsa (*expired*) selama 60 menit.

4.2.7. Perancangan Pengujian

Sebelum *web service* ini layak digunakan pada aplikasi, maka perlu dilakukan rangkaian uji coba untuk memastikan bahwa *web service* ini dapat digunakan dengan baik, diantara metode pengujian yang dilakukan adalah :

Tabel 5. *Tools* Pengujian

No.	Jenis pengujian	Tujuan	Penguji
1.	*Blackbox Testing*	Untuk menguji fungsionalitas	Peneliti (pengembang)
2.	Postman	Proses *development* API	Peneliti (pengembang)

Menguji fungsionalitas fitur-fitur yang ada dengan menggunakan *blackbox* dan Postman, dan berfokus pada pengujian GET, POST, DELETE, UPDATE pada sistem *web service*, apakah sudah dapat berjalan sesuai dengan yang diharapkan atau belum, berikut adalah tabel pengujian yang telah dibuat :

Tabel 6. Rancangan Pengujian *Black Box*

No	Pengujian	Deskripsi uji	Hasil uji	Keterangan
1.	GET	Proses pengambilan data dari *web service*	Berhasil/ Tidak berhasil	Belum diuji
2.	POST	Proses Penambahan data ke *web service*	Berhasil/ Tidak berhasil	Belum diuji
3.	PUT	Proses perubahan data pada *web service*	Berhasil/ Tidak berhasil	Belum diuji
4.	DELETE	Proses penghapusan data dari *web service*	Berhasil/ Tidak berhasil	Belum diuji

5. HASIL IMPLEMENTASI DAN PENGUJIAN

Berikut ini adalah hasil implementasi perangkat lunak dari model REST API *web service* yang telah dirancang.

5.1. Hasil Pengujian Dengan Postman

Parameter yang diuji pada *web service* adalah bagian dari fungsional dari API yang dirancang, sedangkan untuk hasil pengujian yang sesuai dengan apa yang diinginkan, dirinci pada sub bab berikut ini :

5.1.1. Format URI API *Web Service*

Dikarenakan API *web service* yang akan dirancang menggunakan RESTful (*Representational State Transfer*) yang memungkinkan klien dapat melakukan request melalui protokol HTTP dengan mudah menggunakan URI, berikut daftar struktur dari URI yang akan digunakan :

1. Format URI Modul Mahasiswa

Tabel 7. Format URI Model Mahasiswa

No.	*Method*	*Mapping*
1.	GET	/Mahasiswa/get?(nim_mhs) /Mahasiswa/getall
2.	POST	/Mahasiswa/post
3.	PUT	/Mahasiswa/put
4.	DELETE	/Mahasiswa/delete?(nim_mhs)

2. Format URI *Modul* Matakuliah

Tabel 8. Format URI *Modul* Matakuliah

No.	*Method*	*Mapping*
1.	GET	/Matakuliah/get?(kode_mk) /Matakuliah/getall
2.	POST	/Matakuliah/post
3.	PUT	/Matakuliah/put
4.	DELETE	/Matakuliah/delete?(kode_mk)

3. Format URI Modul KHS

Tabel 9. Format URI *Modul* KHS

No.	*Method*	*Mapping*
1.	GET	/Khs/Khsdetail?(nim_mhs)&(tahun_ajaran) /Khs/getall
2.	POST	/Khs/post
3.	PUT	/Khs/put
4.	DELETE	/Khs/delete?(nim_mhs)&(kode_mk)

4. Format URI Modul Oauth

Tabel 10. Format URI *Oauth*

No.	*Method*	*Mapping*
1.	POST	/oauth/token

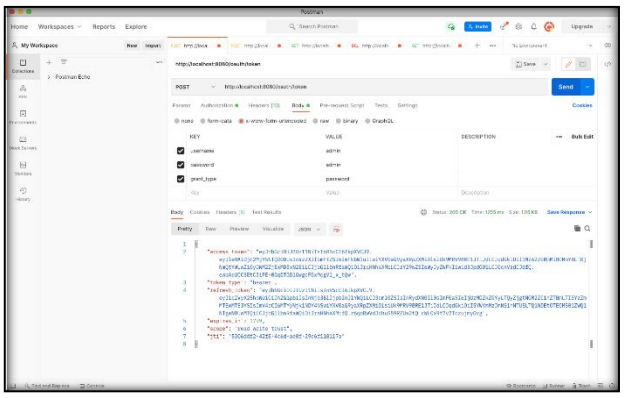
5. Format URI Modul Oauth

Tabel 11. Format URI *User*

No.	*Method*	*Mapping*
1.	POST	/users

5.1.2. Hasil dari Oauth2

Pengujian fungsional yang dilakukan menggunakan aplikasi Postman, data yang diuji adalah data *user* dengan *role admin* dan *user* untuk mendapatkan token agar dapat mengakses mapping lainnya, hasil pengujian dapat dilihat pada gambar dibawah ini :

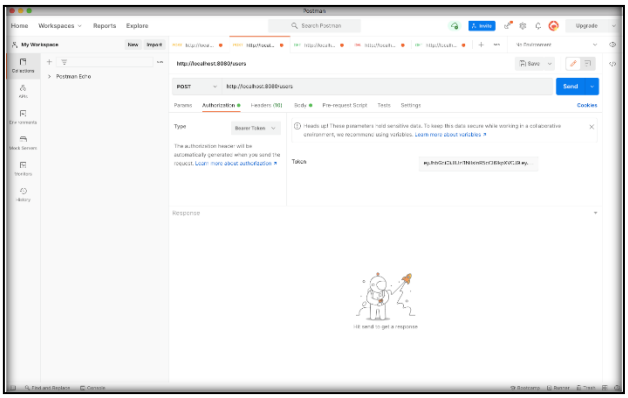


Gambar 11. *Login User* Admin dengan Oauth2

Pengujian dilakukan pada aplikasi Postman, pada alamat URI http://localhost:8080/oauth/token, dengan mengisi pada bagian *body* di kolom x-www-urlecoded, dengan parameter *username*, *password*, dan *grant_type*, *server* akan mengautentikasi permintaan melalui *http basic auth*, *server* lalu menggenerasikan token dan diberikan kepada *client*, dan akan habis masanya pada waktu 30 menit, dan bisa digunakan untuk mapping-mapping selanjutnya.

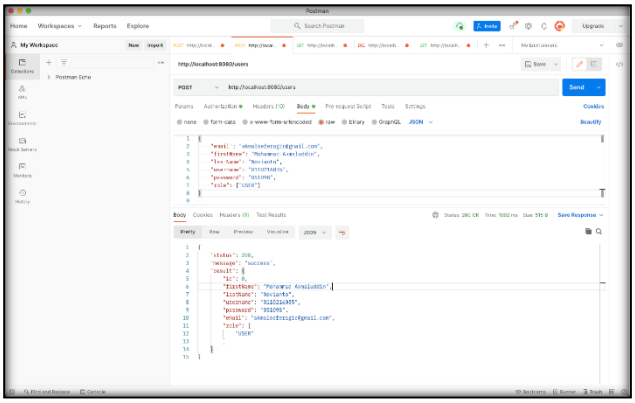
5.1.3. Hasil Modul *User* Model

Pengujian fungsional ini dilakukan dengan Postman, data yang diuji adalah data *user* yang akan diakses setelah mendapatkan token, token tersebut diletakkan pada kolom *Authorization* bagian *bearer* token, dan melakukan method POST dengan alamat URI http://localhost:8080/users untuk membuat akun *user*, hanya akun dengan *role admin* yang bisa membuat akun *user* lainnya.



Gambar 12. Memasukkan Token pada *Bearer* Token

Lalu memasukkan data *user* yang akan ditambahkan melalui kolom *body* dengan format JSON dengan isian *email*, *firstName*, *lastName*, *username*, *password*, dan juga role user. Setelah memasukkan *text* berformat JSON pada *raw body*, lalu melakukan POST dengan harapan berhasil untuk membuat *user* baru dengan *role* [*USER*].



Gambar 13. Hasil dari Membuat Akun *Role User* dengan *Method POST*

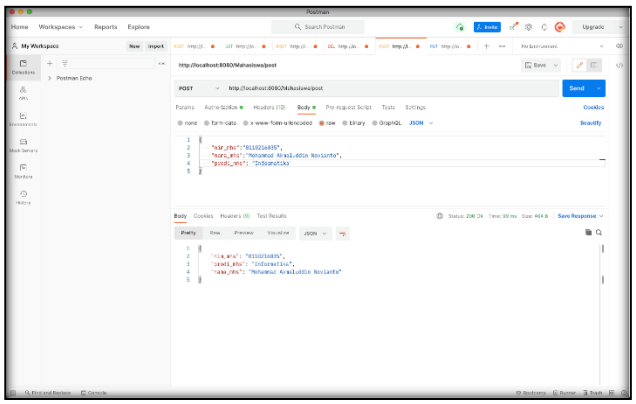
Data user baru berhasil ditambahkan dengan output status 200, dan body dengan *format text* JSON yang dikeluarkan.

5.1.4. Hasil Modul Mahasiswa Model

Pengujian fungsional ini dilakukan dengan Postman, data yang diuji adalah data modul mahasiswa yang akan diakses setelah mendapatkan token, token tersebut diletakkan pada kolom *Authorization* bagian *bearer* token, dan melakukan akses dengan alamat URI http://localhost:8080/Mahasiswa.

1. Melakukan *method* POST

Melakukan *method* POST untuk menambahkan data mahasiswa dengan URI http://localhost:8080/Mahasiswa/post dengan memasukkan *request body* seperti pada gambar 14 dengan memilih *radio button* jenis *raw* dan memasukkan *text* berformat JSON, akan menampilkan hasil seperti pada gambar 14.

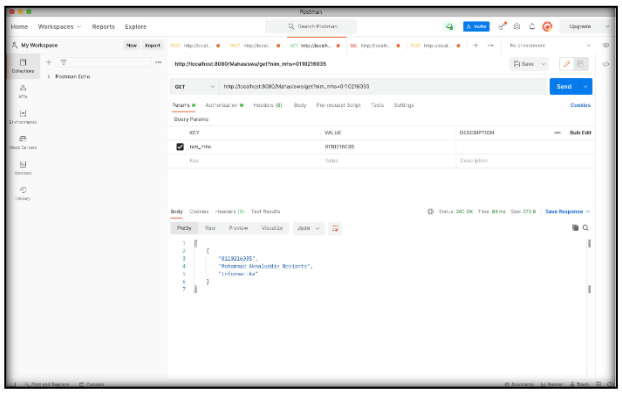


Gambar 14. Hasil dari POST Mahasiswa Model

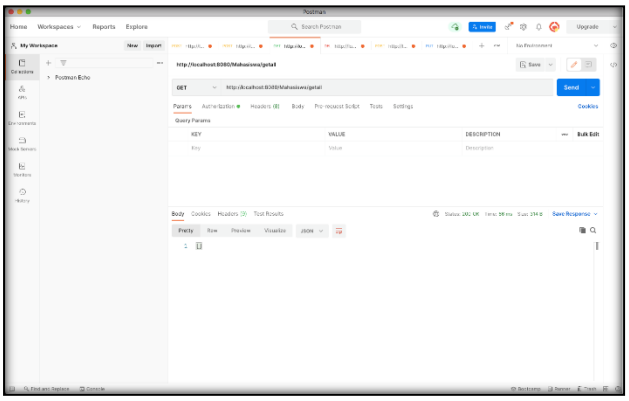
Data yang sudah dimasukkan sesuai dengan gambar diatas, maka data yang diharapkan masuk ke dalam sistem, akhirnya berhasil ditambahkan dengan status 200, dan *body* menampilkan ulang apa yang kita masukkan pada kolom *body* di atasnya.

2. Melakukan *method* GET

Melakukan *method* GET untuk mengecek atau melihat daftar mahasiswa dengan URI http://localhost:8080/Mahasiswa/get dengan memasukkan parameter nim mahasiswa dengan *syntax key* nim_mhs pada kolom *params*, akan menampilkan hasil seperti pada gambar 15, lalu dengan URI http://localhost:8080/Mahasiswa/getall untuk melihat semua data mahasiswa model yang ada pada database sistem.



Gambar 15. Hasil dari GET Mahasiswa model



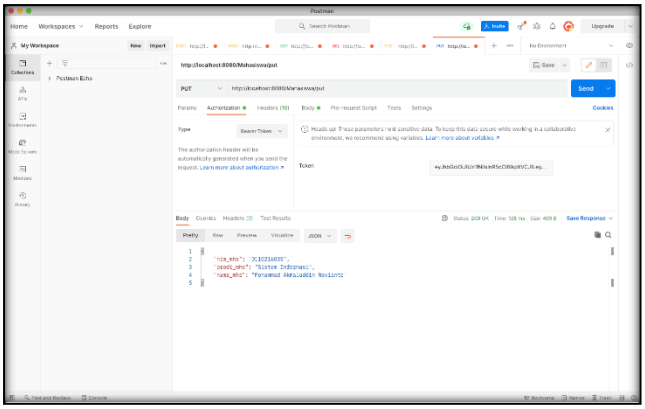
Gambar 16. Hasil dari *method* GET (all) Mahasiswa

Pada hasil keluaran dari *method* GET dengan parameter nim mahasiswa, didapatkan keluaran yang menampilkan data mahasiswa berformat JSON dengan struktur :

- nim_mhs : nim mahasiswa
- nama_mhs : nama mahasiswa
- prodi_mhs : prodi mahasiswa

3. Melakukan *method* PUT

Melakukan *method* PUT untuk mengubah data mahasiswa dengan URI http://localhost:8080/Mahasiswa/put dengan memasukkan *request body* seperti pada gambar 14 dengan memilih *radio button* jenis *raw* dan memasukkan *text* berformat JSON sama seperti akan melakukan method POST, hanya saja ditambahkan id_mhs agar sistem tau *id* mana yang akan diubah.



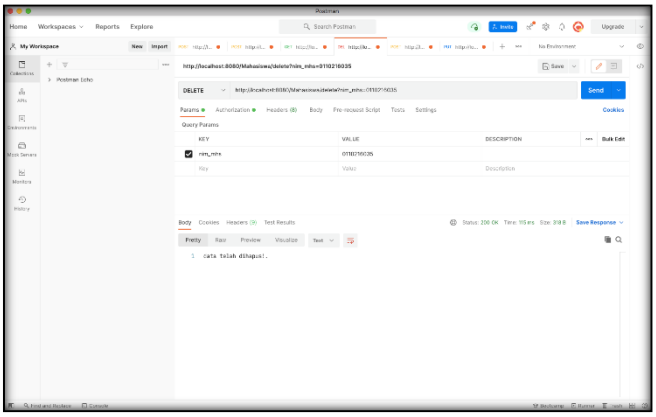
Gambar 17. Hasil dari *method* PUT Mahasiswa model

Jika *output* yang dihasilkan terlihat seperti pada gambar 17, maka pengubahan data dengan method PUT berhasil dilakukan.

4. Melakukan *method* DELETE

Melakukan delete data dengan menggunakan *method* DELETE dengan URI http://localhost:8080/Mahasiswa/delete untuk menghapus data dari mahasiswa model, dengan memasukkan

parameter nim mahasiswa, dengan *syntax key* nim_mhs seperti gambar di bawah ini.



Gambar 18. Hasil dari *method* DELETE Mahasiswa model

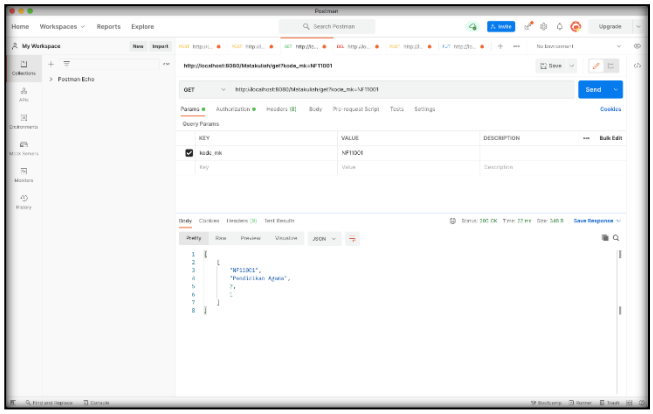
Setelah memasukkan parameter nim mahasiswa yang ada, maka output "data telah dihapus" akan muncul sebagai keluaran seperti yang ada pada gambar 18.

5.1.5. Hasil Modul Matakuliah Model

Pengujian fungsional ini dilakukan dengan *Postman*, data yang diuji adalah data modul matakuliah yang akan diakses setelah mendapatkan token, token tersebut diletakkan pada kolom *Authorization* bagian *bearer* token, dan melakukan akses dengan alamat URI http://localhost:8080/Matakuliah.

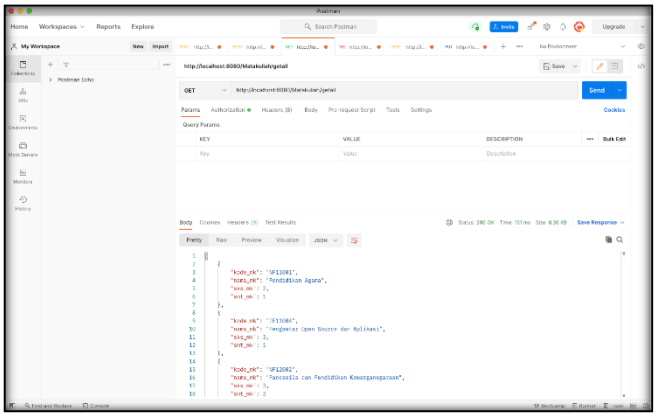
1. Melakukan *Method* GET

Melakukan method GET untuk mengecek atau melihat data yang ada pada daftar matakuliah dengan URI http://localhost:8080/Matakuliah/get, dengan memasukkan parameter kode matakuliah dengan *syntax key* kode_mk pada kolom *params*, akan menampilkan hasil seperti pada gambar 18



Gambar 19. Hasil dari *method* GET modul matakuliah model

Lalu jika ingin melihat data matakuliah secara keseluruhan, dapat menggunakan URI http://localhost:8080/Matakuliah/getall, lalu sistem akan menampilkan semua daftar matakuliah seperti yang ada pada gambar 20



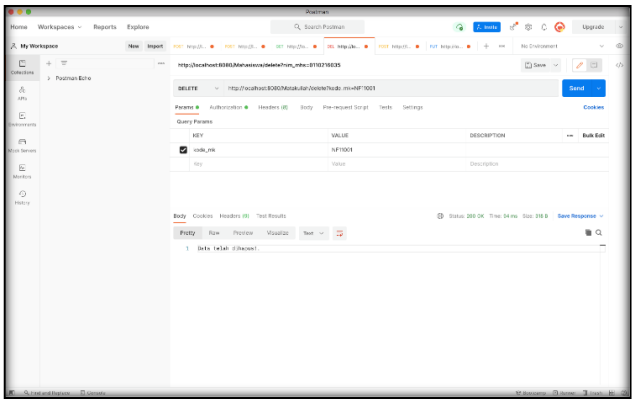
Gambar 20. Hasil dari *method* GET(all) pada modul Matakuliah model

Pada hasil keluaran dari *method* GET dengan atau tanpa parameter kode matakuliah, didapatkan keluaran yang menampilkan data matakuliah berformat JSON dengan struktur :

- kode_mk : kode matakuliah
- nama_mk : nama matakuliah
- sks_mk : jumlah sks matakuliah
- smt_mk : semester matakuliah

2. Melakukan *Method* DELETE

Melakukan delete data matakuliah dengan menggunakan method DELETE dengan URI http://localhost:8080/Matakuliah/delete untuk menghapus data dari matakuliah model, dengan memasukkan parameter kode matakuliah dan menggunakan *syntax key* kode_mk seperti pada gambar 21 di bawah ini.

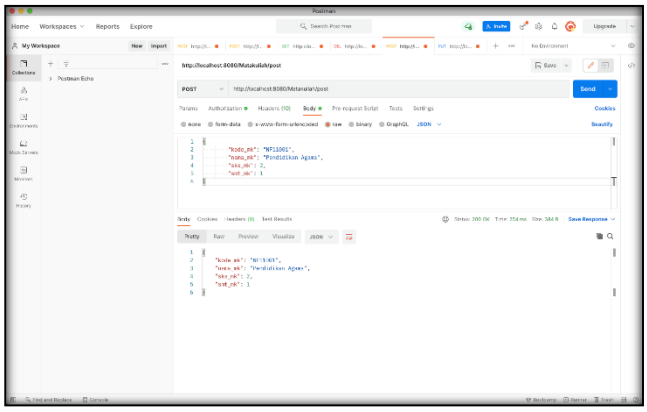


Gambar 21. Hasil dari *method delete* pada modul matakuliah model

Sistem akan mengembalikan keluaran dengan format string "Data telah dihapus!" seperti pada gambar 21, yang mana artinya berhasil menghapus data yang diinginkan.

3. Melakukan *method POST*

Melakukan *method POST* untuk menambahkan data matakuliah dengan URI http://localhost:8080/Matakuliah/post dengan memasukkan *request body* seperti pada gambar 22 dengan memilih *radio button* jenis *raw* dan memasukkan *text* berformat JSON, akan menampilkan hasil seperti pada gambar 22 di bawah ini.

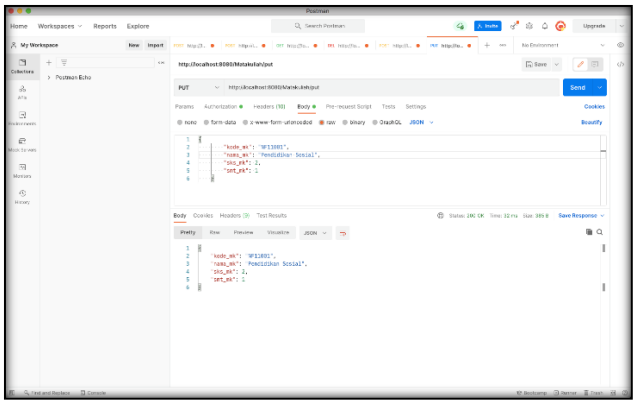


Gambar 22. Hasil dari *Method* POST pada Modul Matakuliah Model

Data yang sudah dimasukkan sesuai dengan gambar diatas, maka data yang diharapkan masuk ke dalam sistem, akhirnya berhasil ditambahkan dengan status 200, dan *body* menampilkan ulang apa yang kita masukkan pada kolom *body* diatasnya.

4. Melakukan *Method PUT*

Melakukan *method PUT* untuk mengubah data matakuliah dengan URI http://localhost:8080/Matakuliah/put dengan memasukkan *request body* seperti pada gambar 23 dengan memilih *radio button* jenis *raw* dan memasukkan *text* berformat JSON sama seperti akan melakukan *method* POST.



Gambar 23. Hasil Keluaran dari *Method* PUT pada Matakuliah

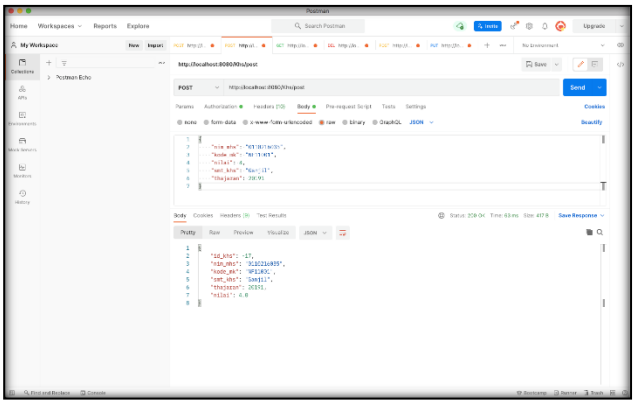
Jika output yang dihasilkan terlihat seperti pada gambar 26, maka pengubahan data dengan method PUT berhasil dilakukan.

5.1.6. Hasil Modul KHS Model

Pengujian fungsional ini dilakukan dengan Postman, data yang diuji adalah data modul kartu hasil studi (KHS) yang akan diakses setelah mendapatkan token, token tersebut diletakkan pada kolom *Authorization* bagian *bearer* token, dan melakukan akses dengan alamat URI utama http://localhost:8080/Khs.

1. Melakukan *Method POST*

Melakukan *method POST* untuk menambahkan data pada modul kartu hasil studi (KHS) dengan URI http://localhost:8080/Khs/post dengan memasukkan *request body* seperti pada gambar 24 dengan memilih *radio button* jenis *raw* dan memasukkan *text* berformat JSON, akan menampilkan hasil seperti pada gambar 18.

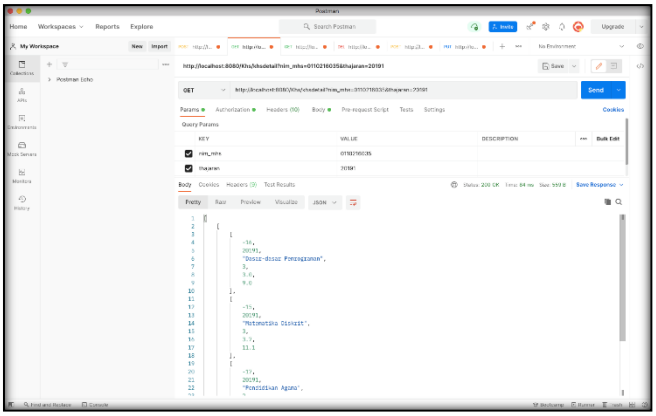


Gambar 24. Hasil dari *Method POST* pada Modul KHS Model

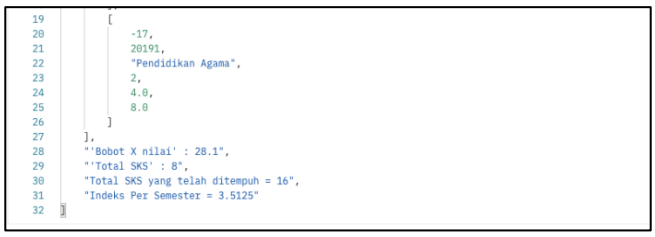
Data yang sudah dimasukkan sesuai dengan gambar diatas, maka data yang diharapkan masuk ke dalam sistem, akhirnya berhasil ditambahkan dengan status 200, dan *body* menampilkan ulang apa yang kita masukkan pada kolom *body* di atasnya.

2. Melakukan *Method* GET

Melakukan *method* GET untuk mengecek atau melihat data yang ada pada daftar kartu hasil studi (KHS) dengan URI http://localhost:8080/Khs/Khsdetail, dengan memasukkan parameter nim mahasiswa dan tahun ajaran dengan *syntax key* nim_mhs dan tahun_ajaran pada kolom *params*, akan menampilkan hasil dari hasil studi yang diambil oleh mahasiswa pada tahun ajaran yang di*input seperti* pada gambar 25.



Gambar 25. Hasil dari *method* GET pada *modul* KHS model



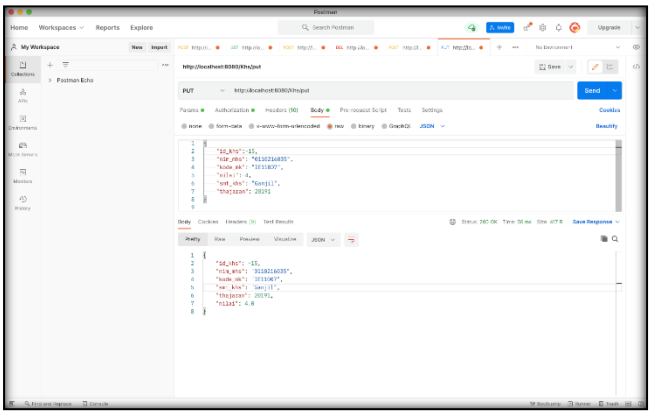
Gambar 26. Hasil dari *method* GET pada *modul* KHS model

Pada hasil keluaran dari *method* GET dengan atau tanpa parameter nim mahasiswa dan tahun ajaran, didapatkan keluaran yang menampilkan data kartu hasil studi berformat JSON dengan struktur :

- id_khs = id dari matakuliah pada modul khs
- tahun_ajaran = tahun ajaran kartu hasil studi
- nama_mk = nama matakuliah yang diambil
- sks_mk = jumlah sks yang ada pada matakuliah tersebut
- nilai = nilai yang didapatkan pada matakuliah tersebut
- bobot sks dikali nilai
- total sks pada semester tersebut
- total sks keseluruhan
- indeks per semester yang didapatkan pada semester tersebut

3. Melakukan *Method* PUT

Melakukan *method* PUT untuk mengubah data kartu hasil studi (KHS) dengan URI http://localhost:8080/Khs/put dengan memasukkan *request body* seperti pada gambar 27 dengan memilih *radio button* jenis *raw* dan memasukkan *text* berformat JSON sama seperti akan melakukan *method* POST, hanya saja ditambahkan id_khs agar sistem tau *id* mana yang akan diubah.

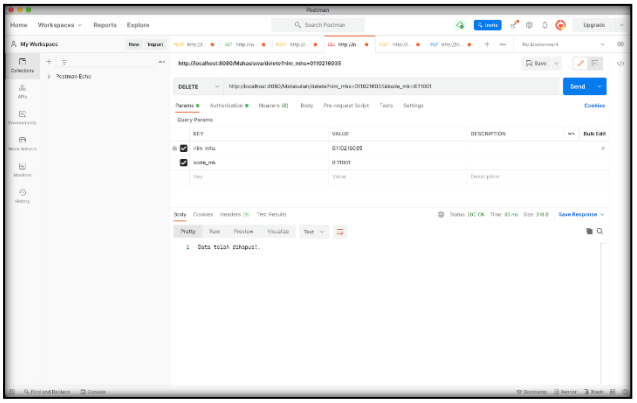


Gambar 27. Hasil dari *method* PUT pada *modul* KHS model

Jika output yang dihasilkan terlihat seperti pada gambar 27, maka pengubahan data dengan method PUT berhasil dilakukan.

4. Melakukan *Method* DELETE

Melakukan *delete* data dengan menggunakan *method* DELETE dengan URI http://localhost:8080/Khs/delete untuk menghapus data dari kartu hasil studi (KHS) model, dengan memasukkan parameter nim mahasiswa dan kode matakuliah, dengan *syntax key* nim_mhs dan kode_mk seperti gambar dibawah ini.

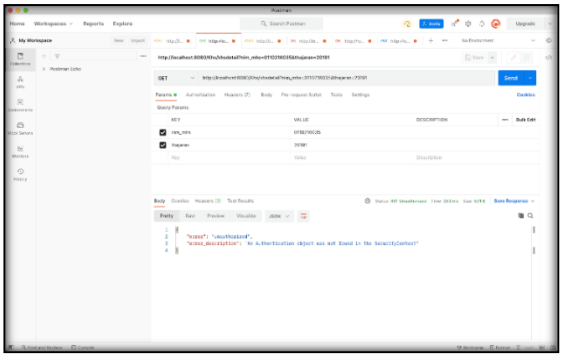


Gambar 28. Hasil dari *method DELETE* pada *modul* KHS model

Setelah memasukkan parameter nim mahasiswa dan kode matakuliah yang ada, maka *output* "data telah dihapus" akan muncul sebagai keluaran seperti yang ada pada gambar 28.

5.1.7. Hasil Pengujian Jika Tanpa Token

Setelah pengujian terhadap API berhasil dilakukan dengan token, kini saatnya menguji bagaimana API yang diakses tanpa menggunakan token pada *Authorization*, berikut adalah *output* yang ditampilkan pada aplikasi Postman saat mencoba untuk GET pada data KHS.

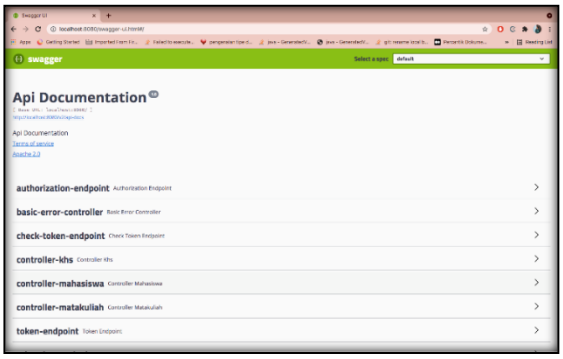


Gambar 29. Hasil Jika Tanpa Token

Pada saat kita tidak memasukkan token *authorization* pada saat mengakses API *web service*, maka hasil keluaran yang muncul adalah 401 *Unauthorized*.

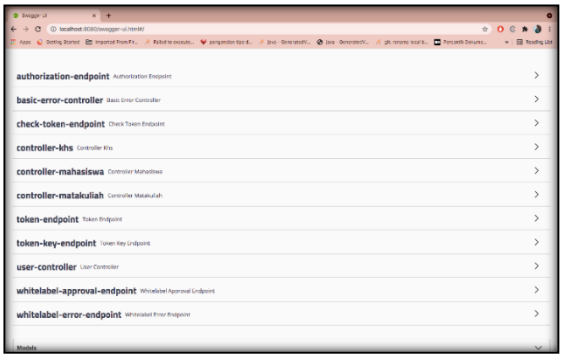
5.2. Dokumentasi API

Dokumentasi berikut bertujuan untuk memberikan penjelasan mengenai API yang telah dirancang, sehingga dapat dengan mudah digunakan oleh pengembang lainnya, berikut adalah tampilan dokumentasi dari API ini, dengan menggunakan Swagger 2.

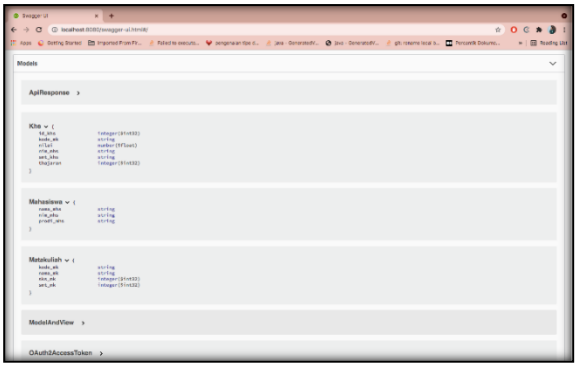


Gambar 30. *Swagger Documentation* (a)

Fungsi utama dalam pembuatan dokumentasi API ini adalah pengguna dapat melihat *method*, dan model apa saja yang ada dan tersedia, tipe data yang digunakan, mapping yang digunakan, dan apa saja parameter yang dibutuhkan untuk sebuah *method*, dengan mengakses http://localhost:8080/Swagger-ui.html .



Gambar 31. *Swagger Documentation* (b)



Gambar 32. *Swagger Documentation* (c)

5.3. Pengujian Menggunakan *Black Box*

Pengujian dengan *Blackbox testing* dengan 13 skenario yang diuji menggunakan metode pengujian *BlackBox* dan Postman, 13 skenario berhasil diuji dan berjalan 100%, dan sistem menghasilkan respon yang sesuai dengan yang diharapkan.

6. KESIMPULAN DAN SARAN

Pada bagian kali ini dijelaskan tentang kesimpulan dan saran dari penelitian tugas akhir yang telah dilakukan, bagian ini diakhiri dengan saran yang dapat dilakukan pada penelitian selanjutnya.

6.1. Kesimupulan

Berdasarkan hasil penelitian tugas akhir ini, dapat disimpulkan bahwa :

1. Rancang bangun API *web service* dilakukan dengan tahapan-tahapan: (1) Pendefinisian masalah dan ruang lingkup kebutuhan modul web KHS (Kartu Hasil Studi) pada sisitem informasi akademik dilakukan pada saat awal, (2) perancangan sistem mulai dari rancangan diagram alir, URI, hak akses dan skema otentikasi, dokumentasi API, (3) Implementasi program menggunakan Spring *Framework* dan database PostgreSQL, dengan sistem otentikasi menggunakan Oauth2, dan dokumentasi dengan menggunakan Swagger, (4) Pengujian aplikasi API *web service*, (5) Penarikan kesimpulan dan saran.
2. REST API *web service* telah berhasil diimplementasikan dengan hasil pengujian terhadap 4 fitur, yaitu GET, POST, PUT, DELETE pada 3 modul (mahasiswa, matakuliah, KHS), dengan 13 skenario yang diuji menggunakan metode pengujian *BlackBox* dan Postman, 13 skenario berhasil diuji dan berjalan 100%, dan sistem menghasilkan respon yang sesuai dengan yang diharapkan.

6.2. Saran

Berdasarkan hasil penelitian tugas akhir ini, beberapa hal dapat dijadikan saran untuk pengembangan penelitian berikutnya :

1. Implementasi *database* pada API *web service* dapat lebih kompleks dan menyerupai database utama AIS STT Terpadu Nurul Fikri.
2. Dapat menampilkan fitur Indeks Per Kumulatif (IPK) pada penelitian selanjutnya.
3. Pembuatan API *web service* untuk fitur aplikasi *mobile* AIS lainnya dibuat menggunakan RESTful Spring Framework.

DAFTAR PUSTAKA

[1] F. Kapojos et.al., "Implemenatasi *Service-Oriented Architecture* dengan *Web Service* untuk Aplikasi Sistem Informasi Akademik," *Jurnal Teknik Elektro dan Komputer*, Vol. 1, No. 1, 2012.

[2] T. Erl, "*Service Oriented Architecture:Concepts, Technology, and Design*," Prentice Hall Professional Technical Reference, 2005.

[3] E Indrayani, "Pengelolaan Sistem Informasi Akademik Perguruan Tinggi berbasis Teknologi Informasi dan Komunikasi (TIK)," *Jurnal Penelitian Pendidikan*, Vol. 12, No. 1, pp. 51-67, 2011.

[4] T Andriyanto, "Rancang Bangun Sistem Informasi Praktek Kerja Lapangan Terintegrasi menggunakan *Web Service,"* Simetris: Jurnal Teknik Mesin, Elektro dan Ilmu Komputer, Vol. 7, No. 2, pp. 551-558, 2016.

[5] R Daigneau, "*Service Design Patterns: Fundamentals Design Solutions for SOAP/WSDL and RESTful Web Services,"* Boston: Pearson educatin.Inc, 2012.

[6] D. Sprott & L. Wilkes, "*Understanding Service-Oriented Architecture*," *Microsoft Architect Journal*, Vol. 1, No. 1, pp. 10-17, 2004.

[7] B. Burke, "*RESTful Java with Jax-RS*," O'Reilly Media, Inc, 2009.

[8] J. Webber, S. Parastatidis, & I. Robinson, "*REST In Practice: Hypermedia and Systems Architecture,"* O'Reilly Media, Inc, 2010.

[9] R. S. Pressman, "*Software Testing Strategies*," In Software Engineering: A Practitioner's Approach.: McGraw-Hill, 2001.

[10] J. Hunt, "*Guide to the Unified Process featuring UML, Java and Design Patterns,"* Springer Science & Business Media, 2006.

[11] A. Mujahid et.al., "Analisis dan Pengembangan Sistem Informasi Pengelolaan Masjid berbasis *Mobile* dengan Teknologi API *Web Service*," *Jurnal Informatika Terpadu*, Vol. 7, No. 2, hal. 80-86, 2021. https://doi.org/10.54914/jit.v7i2.368

[12] M. R. A. Putra, & S. Munir, "Rancang Bangun *Website Customer Relationship Management* (CRM) pada Modul Akuntansi Studi Kasus CV Esindo Multi Tata," *Jurnal Informatika Terpadu*, Vol. 5, No. 1, hal. 24-29, 2019. https://doi.org/10.54914/jit.v5i1.176