PERANCANGAN DAN IMPLEMENTASI RESTFUL API PADA SISTEM INFORMASI MANAJEMEN DOSEN UNIVERSITAS UDAYANA

Ida Ayu Kaniya Pradnya Paramitha[1], Dewa Made Wiharta[2], I Made Arsa Suyadnya[3]
[1]Mahasiswa Teknik Elektro, Fakultas Teknik, Universitas Udayana
[2,3]Dosen Teknik Elektro, Fakultas Teknik, Universitas Udayana
Jl. Raya Kampus UNUD, Bukit Jimbaran, Kuta Selatan, Badung, Bali
kaniyapradnya20@gmail.com[1], wiharta@unud.ac.id[2], arsa.suyadnya@unud.ac.id[3]

ABSTRAK

Sistem Informasi Manajemen (SIM) Dosen Universitas Udayana adalah sebuah sistem informasi manajemen yang berfungsi untuk membantu dosen dalam meningkatkan kinerja dalam penyelenggaraan pendidikan tinggi yang bermutu. SIM Dosen dirancang dengan menggunakan arsitektur monolitik yang akan beralih menjadi arsitektur *microservice* yang bersifat lebih fleksibel karena memecah beberapa aplikasi ke beberapa *service*. Komunikasi antar *service* dilakukan melalui protokol HTTP dengan menggunakan RESTful API. Penelitian ini berfokus pada keberhasilan dalam merancang sebuah RESTful API pada salah satu modul yang tersedia pada SIM Dosen yaitu modul data dosen. RESTful API berhasil dirancang dengan menggunakan bahasa pemrograman Java dan *database* MySQL. Pengujian fungsionalitas RESTful API dilakukan dengan menggunakan metode pengujian *black box* dan Postman pada perangkat lokal. Pengujian memberikan hasil yaitu pada *method* GET, PUT dan DELETE menampilkan kode *response* 200 dengan status OK serta pada *method* POST menampilkan kode *response* 201 dengan status *Created* yang berarti seluruh fungsi masing – masing HTTP *method* yang digunakan pada modul data dosen telah berjalan sesuai dengan fungsinya.
Kata Kunci: RESTful API, HTTP, Sistem Informasi Manajemen, Universitas Udayana

ABSTRACT

Lecturer Management Information System (SIM Dosen) Udayana University is a management information system to assist lecturers in improving performance in providing higher quality education. SIM Dosen was designed using a monolithic architecture will switch to a microservice architecture that is more flexible because it breaks down several applications into several services. Communication between services is carried out through the HTTP protocol using a RESTful API. This research focuses on the success of designing a RESTful API on one of the modules on the SIM Dosen which is "data dosen" module. RESTful API was successfully designed using the Java programming language and MySQL database. Testing the RESTful API functionality using the black box testing method and Postman on local device for GET, PUT, and DELETE method display a 200 response code and OK status, and for POST method display a 201 response code and Created status which indicate all the functions of each HTTP method used in "data dosen" module have been running according to their functions.
***Key Words** : RESTful API, Management Information System, Udayana University*

1. PENDAHULUAN

Perkembangan teknologi internet sangat membantu dalam menunjang pertukaran informasi dan data yang lebih efektif dan akurat sehingga banyak dimanfaatkan dalam meningkatkan produktivitas suatu institusi. Universitas Udayana, sebagai Perguruan Tinggi terbesar di Bali, telah memiliki sistem informasi untuk pengelolaan data dosen yang diberi nama Sistem Informasi Manajemen Dosen atau SIM Dosen. SIM Dosen berfungsi untuk

membantu dosen dalam meningkatkan kinerja dalam penyelenggaraan pendidikan tinggi yang bermutu, dan telah diimplementasikan dengan menggunakan arsitektur monolitik. Beberapa masalah yang ditemui akibat penggunaan arsitektur monolitik antara lain seluruh sistem akan terpengaruh akibat *error* yang terjadi pada salah satu bagian sistem dan memiliki kesulitan dalam proses pengembangan sistem yang kompleks karena seluruh modul berada di dalam sebuah aplikasi yang besar [1]. Solusi yang dapat dilakukan adalah dengan beralih menuju arsitektur *microservice* yang lebih fleksibel. Arsitektur *microservice* memecah beberapa aplikasi menjadi beberapa *service* yang dapat disesuaikan dengan kebutuhan dan fungsi yang diperlukan [2].

Dalam penelitian yang dilakukan sebelumnya dengan judul "Penerapan Arsitektur *Microservice* pada Sistem Tata Kelola Matakuliah Proyek Politeknik Pos Indonesia" disebutkan bahwa adanya permasalahan dalam proses pengembangan fitur dalam sistem informasi yang memerlukan waktu yang lama karena harus menyesuaikan keseluruhan sistem dan banyaknya komponen yang harus disesuaikan. Maka dari itu, solusi yang dilakukan adalah dengan menerapkan arsitektur *microservice* pada sistem [3].

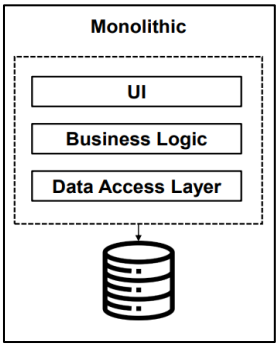
Dalam penelitian dengan judul "Implementasi Teknologi Mikroservice pada Pengembangan *Mobile Learning*" juga disebutkan bahwa penerapan *microservice* dalam pengembangan sebuah perangkat lunak dimana komponen-komponennya dibuat terpisah berdasarkan kegunaan dan fungsinya masing-masing sehingga apabila adanya *error* pada satu bagian, tidak akan mempengaruhi keseluruhan aplikasi [4].

Komunikasi antar *service* dalam *microservice* dilakukan melalui protokol HTTP dengan API atau *Application Programming Interface*. Salah satu penerapan API adalah RESTful API. REST merupakan sebuah metode komunikasi yang menggunakan protokol HTTP sebagai alat pertukaran data. RESTful API yang dirancang menjadi sarana pertukaran data antar *service* yang pada penelitian ini berfokus pada modul SIM Dosen Universitas Udayana. Perancangan RESTful API dilakukan dengan tahapan identifikasi masalah yang ada pada SIM Dosen, pengumpulan data dan informasi yang dapat membantu dalam proses perancangan, perancangan RESTful API dengan menggunakan bahasa pemrograman Java dan *database* MySQL serta dengan menggunakan HTTP *method* (GET, PUT, POST, DELETE). Setelah itu dilakukannya pengujian fungsionalitas RESTful API yang dilakukan dengan menggunakan metode *black box testing* yang bertujuan untuk memastikan apakah fitur – fitur yang dirancang dapat berjalan sesuai dengan yang diharapkan. Pengujian juga dibantu dengan menggunakan Postman yang merupakan alat pengujian REST yang menyediakan fungsi *request* HTTP.

2. LANDASAN TEORI

2.1 Arsitektur *Monolithic*

Arsitektur *monolithic* merupakan arsitektur di mana aplikasi dibangun menjadi satu paket besar yang berisi seluruh fungsionalitas aplikasi yang dibangun [5]. Arsitektir *monolithic* dianggap sebagai cara tradisional untuk membangun aplikasi. Aplikasi yang menggunakan arsitektur ini, dibangun sebagai satu kesatuan. Gambar 1 merupakan gambaran umum dari penerapan arsitektur *monolithic*.



Gambar 1. Arsitektur *Monolithic*

Dalam menerapkan arsitektur *monolithic,* terdapat beberapa kelebihan dan kekurangannya, antara lain dapat dilihat pada Tabel 1.

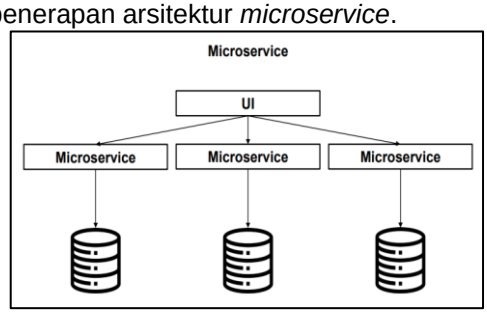
Tabel 1. Kelebihan dan Kekurangan Arsitektur *Monolithic*

Kelebihan	Kekurangan
Proses *debug* dan pengujian lebih mudah dilakukan karena aplikasi	Sistem kode yang kompleks dalam satu aplikasi susah untuk dikelola.

monolithic menjadi satu unit.	
Proses *deployment* yang lebih sederhana karena hanya perlu menangani satu *file*.	Lebih sulit untuk mengimplementasi perubahan karena setiap perubahan kode mempengaruhi keseluruhan sistem sehingga harus dikoordinasikan.
Lebih sederhana untuk dikembangkan apabila memiliki pengetahuan dan kemampuan yang tepat dalam mengembangkan aplikasi *monolithic*.	Adanya hambatan dalam menerapkan teknologi baru karena keseluruhan aplikasi diharuskan untuk diulang.

2.2 Arsitektur *Microservice*

Arsitektur *microservice* merupakan arsitektur yang menyediakan layanan yang lebih fokus dan spesifik, sehingga setiap layanan memiliki tanggung jawabnya tersendiri [6]. Dalam arsitektur *microservice*, setiap layanan memiliki *database*-nya sendiri. Dengan mengimplementasikan arsitektur *microservice*, sebuah sistem informasi dibagi menjadi beberapa layanan yang bersifat independen sehingga dapat memfasilitasi pengembangan sistem [6]. Gambar 2 merupakan gambaran umum penerapan arsitektur *microservice*.



Gambar 2.Arsitektur *Microservice*

Dalam menerapkan arsitektur *microservice,* terdapat beberapa kelebihan dan kekurangannya, antara lain dapat dilihat pada Tabel 2.

Tabel 2. Kelebihan dan Kekurangan Arsitektur *Microservice*

Kelebihan	Kekurangan
Lebih mudah dipelihara karena *service* yang bersifat independen.	Harus mengatur koneksi antara semua modul dan *database*.
Service dapat ditulis dalam bahasa pemrograman yang berbeda tanpa mengurangi kompabilitas.	Terdiri dari banyak modul dan *database* sehingga semua koneksi harus ditangani dengan teratur.
Mengurangi hambatan dalam mengadopsi teknologi baru.	Banyaknya komponen yang dapat digunakan secara independen membuat pengujian solusi berbasis *microservice* menjadi lebih kompleks.

2.3 RESTful API

RESTful API merupakan tipe arsitektur dari *Application Programming Interface* atau API. RESTful API terkadang disebut juga sebagai RESTful web service atau REST API. REST atau *Representational State Transfer* adalah gaya arsitektur dan pendekatan komunikasi yang umum digunakan dalam proses pengembangan *web service*. RESTful API memungkinkan sistem yang berbeda untuk saling berkomunikasi [7]. *Resource* yang berada di *database* sistem dapat dipetakan dengan *endpoint* API di REST [7]. Dalam mengakses data, RESTful API menggunakan perintah HTTP *request* untuk mendapatkan *resource* dan format JSON sering digunakan dalam merepresentasikan *resource* di REST [8].

2.4 HTTP Kode *Response*

HTTP *response code* merupakan kode yang dikirim ketika *client* mengirimkan *request* yang berisi informasi tentang respons server terhadap *request* yang dikirimkan [9]. HTTP mendefinisikan standar *response code* yang dapat dikirim *client* saat mengirimkan hasil permintaan. Tabel 3 merupakan beberapa contoh kode respons HTTP pada RESTful API.

Tabel 3. Kode *Response* HTTP pada RESTful API

Response Code	Deskripsi
200 (OK)	RESTful API berhasil melakukan tindakan yang di-*request client*.

201 (*Created*)	RESTful API akan merespon dengan *response code* 201 setiap kali sebuah *resource* berhasil ditambahkan.
204 (*No Content*)	*Response code* terhadap *request* PUT, POST atau DELETE ketika RESTful API menolak untuk mengirimkan kembali representasi apapun pada *message body*.
404 (*Not Found*)	RESTful API tidak bisa memetakan URI *client* ke *resource*.
500 (*Internal Server Error*)	*Response* yang diberikan saat ditemukan kondisi yang tidak terduga dan tidak ada pesan lebih spesifik yang sesuai.

2.5 Java

Java merupakan bahasa pemrograman *scripting* yang dapat digunakan untuk membangun aplikasi dan juga digunakan untuk menyediakan akses ke objek yang disematkan [10]. Java adalah bahasa berorientasi objek yang memberikan struktur program yang jelas yang meningkatkan fleksibilitas dan memungkinkan penggunaan kembali kode. Java dengan kemampuan *write once run everywhere* yang berarti dapat bekerja dan dijalankan pada platform apapun. Bahasa pemrograman Java dianggap sederhana karena mudah ditulis, dipelajari, dipelihara, dan dipahami, serta kodenya mudah di-*debug*.

2.6 MySQL

MySQL adalah perangkat lunak *database open source* yang banyak digunakan untuk mengolah basis data dan termasuk dalam jenis RDBMS (*Rational Database Management System*) [11]. MySQL menggunakan tipe data relasional dan menyimpan data dalam bentuk tabel yang saling berhubungan untuk memudahkan penyimpanan dan penyajian data.

Proses utama yang terjadi pada MySQL adalah:

1. MySQL membuat *database* untuk menyimpan dan memanipulasi data serta mendefinisikan hubungan setiap tabel.
2. *Client* dapat memasukkan pernyataan SQL tertentu ke MySQL untuk membuat *query*.
3. Aplikasi *server* merespons dengan informasi yang diminta dan ditampilkan di sisi *client*.

2.7 SIM Dosen Universitas Udayana

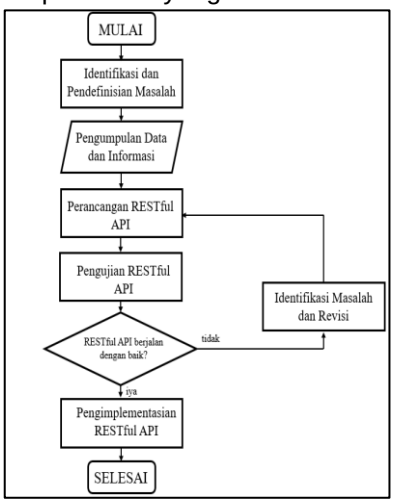
SIM Dosen merupakan sebuah sistem informasi manajemen Universitas Udayana yang dikembangkan dengan tujuan untuk mempermudah pengembangan karir dosen dan pelaporan borang kinerja dosen. Proses perencanaan dan pengembangan karir dosen dilakukan secara terintegrasi dengan harapan meningkatkan kinerja dosen untuk mewujudkan pendidikan tinggi yang bermutu. Dengan SIM Dosen, kinerja dosen selalu dievaluasi setiap semester, sehingga data harus selalu diperbarui. SIM Dosen Universitas Udayana saat ini dikembangkan dengan menggunakan arsitektur *monolithic* sehingga layanan yang ada menjadi satu kesatuan.

2.8 *Black Box Testing*

Black Box testing adalah metode pengujian perangkat lunak yang berfokus pada spesifikasi fungsional tanpa adanya pengujian desain dan kode program itu sendiri. *Black box testing* hanya membutuhkan batas bawah dan batas atas pada data yang diharapkan [12]. *Black box testing* berfokus untuk memastikan bahwa setiap proses yang ada dalam sistem berfungsi sebagaimana mestinya dengan tujuan untuk menemukan *error* yang nantinya akan diperbaiki sehingga sistem dapat dikatakan layak untuk digunakan [13]. *Black box testing* menguji sistem tanpa pengetahuan sebelumnya tentang cara kerja internalnya. Penguji memberikan masukan, dan mengamati keluaran yang dihasilkan oleh sistem yang diuji. Hal ini memungkinkan untuk mengidentifikasi bagaimana sistem merespons tindakan pengguna yang diharapkan dan tidak diharapkan, waktu respons, masalah kegunaan, dan masalah keandalan.

3. METODOLOGI PENELITIAN

Penelitian ini dilakukan di Unit Sumber Daya Informasi (USDI) Universitas Udayana, yang beralamat di Jalan Kampus Bukit Jimbaran, Kecamatan Kuta Selatan, Kabupaten Badung, Bali. Waktu pelaksanaan penelitian ini dimulai pada bulan Oktober 2021. Gambar 3 merupakan alur dari penelitian yang dilakukan.



Gambar 3. Diagram alur penelitian

Langkah-langkah yang dilakukan adalah sebagai berikut:

Langkah 1. Identifikasi dan Pendefinisian Masalah

Penelitian ini diawali dengan identifikasi dan pendefinisian masalah yang ada, yaitu mengenai layanan pertukaran data yang diperlukan oleh SIM Dosen untuk persiapan migrasi dari arsitektur *monolithic* menuju arsitektur *microservice*. Dalam sistem lama, seluruh *service* merupakan satu kesatuan dan memiliki *database* tunggal. Dalam arsitektur *microservice* masing–masing *service* berjalan secara independen dengan *database* terpisah. Untuk itu, diperlukan layanan pertukaran data sebagai penghubung antar *service*. Layanan pertukaran data yang dirancang adalah dalam bentuk RESTful API.

Langkah 2. Pengumpulan Data dan Informasi

Pengumpulan data dan informasi dilakukan melalui suatu diskusi dengan narasumber mengenai kebutuhan dan batasan pada layanan pertukaran data yang dibuat.

Langkah 3. Perancangan RESTful API

RESTful API sebagai media pertukaran data pada SIM Dosen dirancang menggunakan IntelliJ IDEA. Bahasa pemrograman yang digunakan Java dan *database* MySQL. RESTful API akan melakukan akses pada data dan data tersebut akan dibedakan dengan sebuah URIs (*Universal Resource Identifiers*). Proses pembuatan dan perancangan RESTful API menggunakan HTTP *method* yang terdiri dari GET, PUT, POST dan DELETE. GET untuk menampilkan data, PUT untuk memperbarui data, POST untuk mengirim data, dan DELETE untuk menghapus data.

Langkah 4. Pengujian RESTful API

Pengujian RESTful API SIM Dosen Universitas Udayana yang dirancang pada penelitian ini dilakukan dengan metode *black box testing*. *Black box testing* merupakan sebuah pengujian terhadap fungsionalitas atau kegunaan sebuah perangkat lunak. Pengujian dilakukan dengan cara meninjau kesesuain antara data *input* dan *output* pada SIM Dosen Universitas Udayana.

Langkah 5. Pengimplementasian dan Penarikan Kesimpulan

Implementasi dilakukan jika pengujian sudah memberikan hasil sesuai dengan harapan.

4. HASIL DAN PEMBAHASAN

4.1 Hasil

Hasil dari penelitian ini adalah sebuah RESTful API yang dapat digunakan sebagai jembatan ataupun antarmuka untuk integrasi modul–modul yang ada pada Sistem Informasi Manajemen Dosen Universitas Udayana. Penelitian ini hanya fokus pada modul data dosen. RESTful API yang dirancang menggunakan bahasa pemrograman Java dan *database* MySQL.

Perancangan RESTful API pada SIM Dosen menggunakan HTTP *method* yang terdiri dari GET, POST, PUT, serta DELETE. Pada perancangan *method* PUT dan DELETE digunakan *primary key* pada masing–masing tabel sebagai parameternya. Perancangan dilakukan pada server lokal dengan menggunakan Intellij

IDEA dan DBeaver sebagai sarana dalam mengelola *database*.

Pada *class* data dosen kode program berisikan *query* untuk menghubungkan *class* dengan *database* yang digunakan. Query yang dibuat disesuaikan dengan kebutuhan pada masing-masing *method*. Sedangkan pada *resource class* sebagai tempat untuk mengimplementasikan RESTful API dimana anotasi *path* berfungsi untuk mengakses *resourse* serta GET, PUT, POST dan DELETE menentukan HTTP *request* yang digunakan dalam menjalankan RESTful API.

4.2 Pembahasan

Daftar *endpoint* beserta *method* yang digunakan pada modul data dosen ditampilkan pada Tabel 4.

Tabel 4. Daftar *Endpoint* Modul Data Dosen

Method	*Endpoint*
getListDataDosen	/api/v1/dosen
createDataDosen	/api/v1/dosen
getDetailDataDosen	/api/v1/dosen/{id}
updateDataDosen	/api/v1/dosen/update/{id}
deleteDataDosen	/api/v1/dosen/{id}

A. getListDataDosen

Method getListDataDosen digunakan untuk menampilkan keseluruhan data dosen yang ada pada *database*. Kode program 1 merupakan kode program pada *resource class* yang digunakan pada *method* getListDataDosen. *Method* getListDataDosen menggunakan HTTP *method* GET dengan *endpoint* yaitu api/v1/dosen.

Kode Program 1. *Resource Class* getListDataDosen

```
@GET
public Multi<dataDosen>
getListDataDosen() {
    return
dataDosen.findAllDataDosen(pool);
}
```

B. createDataDosen

Method createDataDosen digunakan untuk menambahkan data dosen yang ada pada *database*. Kode program 2 merupakan kode program pada *resource class* yang digunakan pada *method* createDataDosen. Method createDataDosen menggunakan HTTP *method* POST dengan *endpoint* yaitu api/v1/dosen.

Kode Program 2. *Resource Class* createDataDosen

```
@POST
public Uni<Response>
createDataDosen(@QueryParam("semester"
) Integer semester,
@QueryParam("nidn") String  nidn,
@QueryParam("nip") String  nip,
@QueryParam("nama") String nama,
@QueryParam("nama_tercetak") String
nama_tercetak,
@QueryParam("gelar_depan") String
gelar_depan,
@QueryParam("gelar_belakang") String
gelar_belakang, @QueryParam("id_unit")
Integer id_unit,
@QueryParam("id_sunit") Integer
id_sunit, ......,
@QueryParam("akun_sinta") String
akun_sinta,
@QueryParam("akun_sinta_status")
Integer akun_sinta_status,
@QueryParam("akun_publons") String
akun_publons,
@QueryParam("akun_publons_status")
Integer akun_publons_status) {
    return dataDosen.save(pool,
semester, nidn, nip, nama,
nama_tercetak, gelar_depan,
gelar_belakang, id_unit, id_sunit,
......, akun_sinta, akun_sinta_status,
akun_publons, akun_publons_status)
            .onItem().transform(dosen
-> URI.create("api/v1" + dosen))
            .onItem().transform(uri ->
Response.created(uri).build());
}
```

C. getDetailDataDosen

Method getDetailDataDosen digunakan untuk menampilkan data dosen yang ada pada *database* berdasarkan ID dosen yang diinginkan. Kode program 3 merupakan kode program pada *resource class* yang digunakan pada *method* getDetailDataDosen. *Method* getDetailDataDosen menggunakan HTTP *method* GET dengan *endpoint* yaitu api/v1/dosen/{id}.

Kode Program 3. *Resource Class* getDetailDataDosen

```
@GET
@Path("/{id}")
public Uni<Response>
getDetailDataDosen(@PathParam("id")
Integer id_dosen) {
    return
dataDosen.findByIdDataDosen(pool,
id_dosen)

.onItem().transform(dataDosen ->
dataDosen != null ?
Response.ok(dataDosen) :
Response.status(Response.Status.NOT_FO
UND))
```

```
.onItem().transform(Response.ResponseB
uilder::build);
}
```

D. updateDataDosen

Method updateDataDosen digunakan untuk memperbarui data dosen yang ada pada *database* berdasarkan ID dosen yang diinginkan. Kode program 4 merupakan kode program pada *resource class* yang digunakan pada *method* updateDataDosen. *Method* updateDataDosen menggunakan HTTP *method* PUT dengan *endpoint* yaitu api/v1/dosen/update/{id}.

Kode Program 4. *Resource Class* updateDataDosen

```
@PUT
@Path("/update/{id}")
public Uni<Response>
updateDataDosen(@PathParam("id")
Integer id_dosen,
@QueryParam("semester") Integer
semester, @QueryParam("nidn") String
nidn, @QueryParam("nip") String  nip,
@QueryParam("nama") String nama,
@QueryParam("nama_tercetak") String
nama_tercetak,
@QueryParam("gelar_depan") String
gelar_depan,
@QueryParam("gelar_belakang") String
gelar_belakang, @QueryParam("id_unit")
Integer id_unit, ......,
@QueryParam("akun_sinta") String
akun_sinta,
@QueryParam("akun_sinta_status")
Integer akun_sinta_status,
@QueryParam("akun_publons") String
akun_publons,
@QueryParam("akun_publons_status")
Integer akun_publons_status) {
    return
dataDosen.updatedatadosen(pool,
semester, nidn, nip, nama,
nama_tercetak, gelar_depan,
gelar_belakang, id_unit, ......,
akun_sinta, akun_sinta_status,
akun_publons, akun_publons_status,
id_dosen )
.onItem().transform(updated -> updated
? Response.Status.OK :
Response.Status.NOT_FOUND)
            .onItem().transform(status
-> Response.status(status).build());
}
```

E. deleteDataDosen

Method deleteDataDosen digunakan untuk menghapus data dosen yang ada pada *database* berdasarkan ID dosen yang diinginkan. Kode program 5 merupakan kode program pada *resource class* yang digunakan pada *method* deleteDataDosen. *Method* deleteDataDosen menggunakan HTTP *method* DELETE dengan *endpoint* yaitu api/v1/dosen/{id}.

Kode Program V. *Resource Class* deleteDataDosen

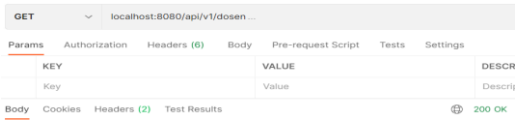
```
@DELETE
@Path("/{id}")
public Uni<Response>
deleteDataDosen(@PathParam("id")
Integer id_dosen) {
    return dataDosen.delete(pool,
id_dosen)
            .onItem().transform(delete
d -> deleted ? Response.Status.OK :
Response.Status.NOT_FOUND)
            .onItem().transform(status
-> Response.status(status).build());}
```

4.3 Pengujian RESTful API

Pengujian dilakukan dalam bentuk *functional testing* terhadap hasil yang didapatkan dengan menggunakan metode *blackbox testing* untuk memastikan fitur–fitur yang dirancang dapat berjalan sesuai dengan yang diharapkan. Pengujian RESTful API dilakukan pada perangkat lokal dengan menggunakan Postman.

A. getListDataDosen

Hasil uji coba getListDataDosen pada Postman menampilkan *response code* 200 dengan status OK yang berarti *request* untuk menampilkan keseluruhan data dosen berhasil dilakukan. Gambar 4 merupakan tampilan hasil dari uji coba getListDataDosen dengan menggunakan Postman.



Gambar 4. Hasil Uji Coba getListDataDosen

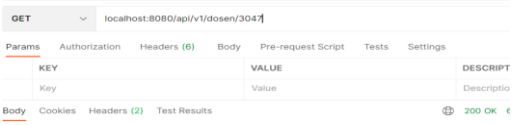
B. createDataDosen

Hasil uji coba createDataDosen pada Postman menampilkan *response code* 201 dengan status *Created* yang berarti *request* untuk menambahkan data dosen berhasil dilakukan. Gambar 5 merupakan tampilan hasil dari uji coba createDataDosen dengan menggunakan Postman.

Gambar 5. Hasil Uji Coba createDataDosen

C. getDetailDataDosen

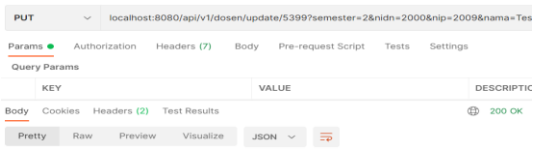
Hasil uji coba getDetailDataDosen pada Postman menampilkan *response code* 200 dengan status OK yang berarti *request* untuk menampilkan data dosen berdasarkan ID dosen berhasil dilakukan. Gambar 6 merupakan tampilan hasil dari uji coba getDetailDataDosen dengan menggunakan Postman.



Gambar 6.Hasil Uji Coba getDetailDataDosen

D. updateDataDosen

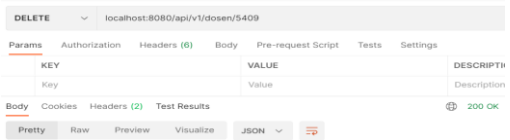
Hasil uji coba updateDataDosen pada Postman menampilkan *response code* 200 dengan status OK yang berarti *request* untuk memperbarui data dosen berdasarkan *primary key* berhasil dilakukan. Gambar 7 merupakan tampilan hasil dari uji coba updateDataDosen dengan menggunakan Postman.



Gambar 7.Hasil Uji Coba updateDataDosen

E. deleteDataDosen

Hasil uji coba deleteDataDosen pada Postman menampilkan *response code* 200 dengan status OK yang berarti *request* untuk menghapus data dosen berdasarkan *primary key* berhasil dilakukan. Gambar 8 merupakan tampilan hasil dari uji coba deleteDataDosen dengan menggunakan Postman.



Gambar 8.Hasil Uji Coba deleteDataDosen

5. KESIMPULAN

Berdasarkan analisa dari hasil penelitian dapat disimpulkan yaitu RESTful API untuk Sistem Informasi Manajemen Dosen Universitas Udayana pada modul data dosen berhasil dirancang dengan tahapan yang pertama yaitu identifikasi masalah, pengumpulan data dan informasi terkait kebutuhan dan batasan pada layanan pertukaran data yang dibuat, perancangan RESTful API dengan menggunakan bahasa pemrograman Java dan *database* MySQL. Pengujian RESTful API dilakukan menggunakan Postman dan tahapan yang terakhir adalah implementasi RESTful API.

Hasil *functional testing* terhadap RESTful API dengan menggunakan metode *black box* pada modul data dosen Sistem Informasi Manajemen Dosen Universitas Udayana menunjukkan bahwa seluruh fungsi masing – masing HTTP *method* yang digunakan telah berhasil berjalan dengan baik. *Method* GET digunakan untuk menampilkan data, PUT untuk memperbarui data, POST untuk menambahkan atau mengirim data serta DELETE untuk menghapus data.

6. DAFTAR PUSTAKA

[1] Mufrizal, R., Indarti, D. 2019. Refactoring Arsitektur Microservice pada Aplikasi Absensi PT. Graha Isaha Teknik. *Jurnal Nasional Teknologi dan Sistem Informasi*, 5(1), 57 – 68

[2] Uminingsih, Handayani, S.D. 2020. Pengorganisasian Kerja Sistem Parkir Menggunakan Arsitektur Microservice. *Jurnal Teknologi*, 13(1), 27 – 35

[3] Saputra, M.H.K., Nabil, L.M. 2021. Penerapan Arsitektur Microservice pada Sistem Tata Kelola Matakuliah Proyek Politeknik Pos Indonesia. Jurnal Teknik Informatika, 13(3), 22 – 28

[4] Sendiang, M., Kasenda, S., Purnama, J. 2018. Implementasi Teknologi Mikroservice pada Pengembangan Mobile Learning. Journal of Applied Informatics and Computing (JAIC), 2(2), 63 – 66

[5] Alchuluq, L.M., Nurzaman, F. 2021. Analisis pada Arsitektur Microservice untuk Layanan Bisnis Toko Online. TEKINFO, 22(2), 61-68

[6] Munawar, G., Hodijah, A. 2018. Analisis Model Arsitektur Microservice pada Sistem Informasi DPLK. *Jurnal & Penelitian Teknik Informatika*, 3(1), 232 - 239

[7] Choirudin, A., Adil, A. 2019. Implementasi REST API Web Service dalam Membangun Aplikasi

Multiplatform untuk Usaha Jasa. *Jurnal MATRIK*, 18(2), 284 – 293

[8] Prityana, A.I., Arfandy, H., Surasa, H. 2021. Pengembangan Servio Menggunakan Full REST API untuk Mendukung Layanan Multiplatform. Jurnal KHARISMA Tech, 16(2), 15-22

[9] Perdana, M.A.K. 2018. Pengembangan REST API Layanan Penyimpanan Menggunakan Metode Rapid Application Development (Studi Kasus: PT. XYZ). *Jurnal Nasional Informatika dan Teknologi Jaringan*, 3(1), 100 – 104

[10] Sallaby, A.F., Utami, F.H., Arliando, Y. 2015. Aplikasi Widget Berbasis Java. *Jurnal Media Infotama*, 11(2), 171-180

[11] Hermiati, R., Asnawari, Kanedi, I. 2021. Pembuatan E – Commerce pada Raja Komputer Menggunakan Bahasa Pemograman PHP dan Database MySQL. *Jurnal Media Infotama*, 17(1), 54 – 66

[12] Cholifah, W.N., Yulianingish, Sagita, S.M. 2018. Pengujian Black Box Testing pada Aplikasi Action & Strategy Berbasis Android dengan Teknologi Phonegap

[13] Wijaya, Y.D., Astuti, M.W. 2021. Pengujian Blackbox Sistem Informasi Penilaian Kinerja Karyawan PT INKA (Persero) Berbasis Equivalence Partitions. *Jurnal Digital Teknologi Informasi*, 4(1), 22 – 26