Majalah Ilmiah Teknologi Elektro, Vol. 21, No.2, Juli - Desember 2022
DOI: https://doi.org/10.24843/MITE.2022.v21i02.P12

Perancangan RESTful API Menggunakan Java Quarkus Untuk Modul Mahasiswa Pada Layanan SIMAK-NG Universitas Udayana

I Kadek Ari Wijaya[1], Dewa Made Wiharta[2], Nyoman Putra Sastra[3]

[Submission: 20-08-2022, Accepted: 05-10-2022]

Abstract*— *Information systems are very important in today's technological era. The services or methods used to deliver information are always evolving to provide better services. SIMAK-NG Udayana University is an information system built by the Udayana University Resources and Information Unit (USDI) to provide convenience to users (academic operators, lecturers and students) in online academic administration activities, which is Integrated Information System, known as Integrated Management Information System the Strategic of Udayana (IMISSU).

Previously built on a monolithic architecture, SIMAK-NG migrate the service architecture to microservices. Therefore, need to customize the program code by implementing a RESTful API that supports the performance of SIMAK-NG service. In this study, the design of a RESTful API model was made This can be used by other applications that require student data from the SIMAK service and access RESTful API services instead of accessing the database directly. API was built using the Java programming language, the Quarkus framework, and a database schema that matched the database at the SIMAK-NG service at Udayana University. The test was run locally and show that the RESTful API program's response results are as designed and work well as shown in the tests using the Postman script. this indicates that the designed API program is ready for implementation.

***Keywords*— *Information*, RESTful API, SIMAK-NG Udayana *University*, Java, Quarkus**

***Intisari*— Sistem Informasi merupakan hal yang sangat penting dalam era teknologi sekarang ini. Layanan atau metode yang digunakan untuk penyampaian informasi selalu berkembang untuk bisa memberikan layanan yang lebih baik. SIMAK-NG Universitas Udayana adalah sistem informasi yang dibangun oleh Unit Sumber Daya dan Informasi (USDI) Universitas Udayana untuk memberikan kemudahan kepada pengguna (operator akademik, dosen dan mahasiswa) dalam kegiatan administrasi akademik secara *online* yang tergabung dalam sebuah Sistem Informasi Terpadu, dengan nama *Integrated Management Information System the Strategic* of Udayana.**

SIMAK-NG sebelumnya dibangun dengan menggunakan arsitektur monolithic dan akan dilakukan migrasi arsitektur layanan menjadi *microservices*, sehingga perlu dilakukan penyesuaian kode program dengan melakukan implementasi RESTful API yang akan menunjang kinerja dari layanan SIMAK-NG. Pada penelitian ini telah dilakukan perancangan model RESTful API yang dapat digunakan oleh aplikasi lain yang memerlukan data mahasiswa dari layanan SIMAK tidak lagi melakukan akses secara langsung ke *database* namun melakukan akses ke layanan RESTful API. API dibangun dengan menggunakan bahasa pemrograman Java, framework Quarkus, dan skema *database* yang sesuai dengan *database* pada layanan SIMAK-NG Universitas Udayana. Pengujian yang dilakukan secara lokal mendapatkan bahwa hasil respon dari program RESTful API yang dibuat sudah sesuai dengan perancangan dan dapat bekerja dengan baik yang ditunjukkan dengan pengujian dengan menggunakan script Postman, hal ini menunjukkan bahwa program API yang dirancang siap untuk diimplementasikan.

***Kata Kunci*— Informasi, RESTful API, SIMAK-NG Universitas Udayana, Java, Quarkus**

I. PENDAHULUAN

Informasi merupakan hal yang sangat berharga di era modern saat ini. Semakin cepat, akurat, dan efektif suatu informasi dapat tersampaikan akan memberi dampak yang besar terhadap pengguna. Seiring dengan perkembangan teknologi, sistem informasi memberikan bantuan pada berbagai tingkat bisnis, mulai dari transaksi operasional hingga inisiatif strategis [1], dengan adanya teknologi informasi dan komunikasi pada Perguruan Tinggi akan memudahkan pendistribusian program dengan kinerja *real-time* sesuai dengan tuntutan pengguna seperti menyalin dokumen, jadwal mengajar, silabus, pemeringkatan, dokumen keuangan mahasiswa [2]. Pemanfaatan teknologi komunikasi dan informasi pada institusi dapat meningkatkan daya saing institusi dengan memberikan pelayanan kepada *stakeholders*-nya [3].

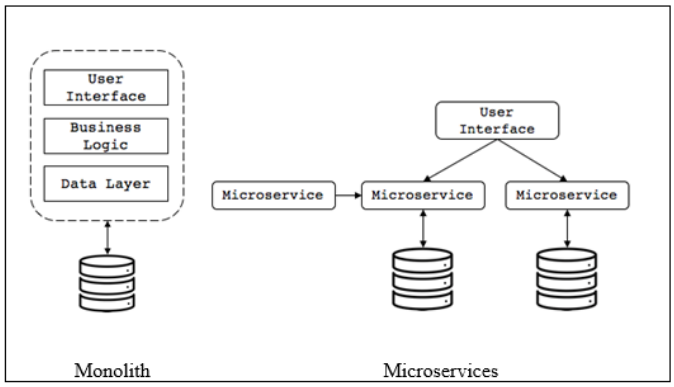
Unit Sumber Daya dan Informasi (USDI) merupakan salah satu unit di Universitas Udayana yang bertanggung jawab terhadap pengembangan aplikasi dan keamanan data. Salah satu layanan informasi yang disediakan oleh USDI adalah Sistem Informasi Terintegrasi (IMISSU) yang terdiri dari beberapa modul, salah satunya adalah SIMAK (Sistem Informasi Manajemen Akademik). SIMAK merupakan suatu sistem informasi yang dibangun untuk memberikan kemudahan kepada pengguna (operator akademik, dosen dan mahasiswa) dalam kegiatan administrasi akademik secara *online*. SIMAK versi NG (*Next Generation*) mengelola proses

[1]*Mahasiswa, Jurusan Teknik Elektro Fakultas Teknik Universitas Udayana, Jln. Kampus Bukit Jumbaran 80361 INDONESIA (tlp: 0361-555225; fax: 0361-4321982; e-mail: ikadekariwijaya1@gmail.com)*

[2, 3] *Dosen, Jurusan Teknik Elektro dan Komputer Fakultas Teknik Universitas Udayana, Jln. Jalan Kampus Bukit Jimbaran 80361 INDONESIA (telp: 0361-703315; fax: 0361-4321; e-mail: wiharta@unud.ac.id, putra.sastra@unud.ac.id)*

 p-ISSN:1693 – 2951; e-ISSN: 2503-2372

input data dari sistem penerimaan mahasiswa baru (E-Registrasi), pengelolaan data mahasiswa, kurikulum, mata kuliah, penawaran mata kuliah, pengisian Kartu Rencana Studi (KRS), pengelolaan rubrik akademik, absensi perkuliahan, pengisian nilai, cetak transkrip, riwayat keaktifan mahasiswa, pengelolaan data skripsi/tesis/disertasi, kerja praktik, publikasi jurnal, integrasi dengan SIMDOS untuk data dosen, integrasi dengan sistem UKT-Ku untuk data pembayaran UKT, integrasi dengan SIM Beasiswa, integrasi dengan sistem pelaporan PDDIKTI hingga proses pendaftaran wisuda. Selain itu, SIMAK juga dapat berfungsi sebagai pendukung untuk analisis data dalam pengambilan keputusan terkait bidang akademik di Universitas Udayana. Saat ini arsitektur dari layanan SIMAK masih dalam bentuk *monolithic*. Kelemahan dari arsitektur *monolithic* yaitu semakin kompleks dan bertambahnya pengguna aplikasi maka akan berpengaruh terhadap biaya yang dikeluarkan karena seluruh operasional akan dibebankan pada satu buah *server* yang sama, selain itu akan berimbas kepada perkembangan dari aplikasi karena setiap terjadi perubahan akan berpengaruh terhadap keseluruhan sistem. Solusi yang ditawarkan untuk mengatasi permasalahan dalam arsitektur *monolithic* adalah dengan melakukan migrasi sistem arsitektur *microservies*, layanan-layanan akan dibagi menjadi beberapa *services* yang diharapkan dapat bekerja lebih efektif dan optimal. Gambar 1 menampilkan skema umum dari kedua arsitektur tersebut.



Gambar 1: Perbandingan Arsitektur *Monolith* dan *Microservices*

Layanan arsitektur *microservices* bekerja dengan menghubungkan satu layanan dengan layanan lain melalui protokol *Hypertext Transfer Protocol* (HTTP) melalui *Application Programming Interface* (API) sehingga apabila terdapat satu layanan yang bermasalah tidak akan mengganggu keseluruhan sistem dan hanya berdampak pada layanan tersebut [13], [15].

API merupakan sebuah *software* yang menghubungkan satu aplikasi dengan aplikasi yang lain. Tujuan pembuatannya adalah untuk saling berbagi data antar aplikasi. REST API merupakan salah satu desain arsitektur yang terdapat di dalam API itu sendiri. Cara kerja dari RESTful API yaitu REST (*Representational State Transfer*) *client* akan melakukan akses pada *data*/*resource* pada REST *server*. Masing-masing *resource* atau *data-resource* tersebut akan dibedakan oleh sebuah global ID atau *Universal Resource Identifiers*, sehingga bahasa pemrograman maupun sistem operasi yang berbeda dapat bekerja bersama dalam sistem ini.

Beberapa penelitian mengenai implementasi *microservice* dan penggunaan sistem penghubung pada sistem *multi-platform* telah dilakukan. Penelitian [4] membahas perkembangan kegiatan pariwisata di pulau Lombok yang menyebabkan berbagai macam profesi yang ditekuni atau dimanfaatkan oleh masyarakat khususnya dibidang jasa, sehingga dibuatlah aplikasi *multi-platform* yang dapat memberikan kemudahan kepada pengguna dalam memilih *platform* yang akan digunakan. Sedangkan pada [5] membahas kinerja dan kompleksitas aplikasi web yang dibangun menggunakan *framework* yang berbeda untuk Bahasa pemrograman Java, adapun *framework* yang dibandingkan yaitu Spring Boot, Micronaut, Quarkus dan Javalin.

Penelitian [6] membahas penanganan *garbage collection* (GC) yang pada versi terbaru 9.0.1 Java SE Development Kit (JDK) *default* GC diubah menjadi *Garbage-First* (G1) GC yang sekarang diadopsi secara luas selain Paralel GC dan *Concurrent Mark* & *Sweep* (CMS) atau GC yang sebelumnya digunakan.

Referensi [7] secara garis besar menjabarkan penggunaan *framework* Quarkus dalam penerapan arsitektur REST API pada layanan *microservices*.

Referensi [8] menjabarkan teori mengenai prinsip dasar pemrograman yang diperlukan dalam membuat API seperti data, transaksi, dan *Object Relational Mapping* hingga pembahasan menggunakan HTTP sebagai media transfer data pada layanan *microservices*.

Referensi [9] membahas perbedaan antara *framework* Quarkus dan Spring khususnya dalam penerapan API pada arsitektur aplikasi RESTful hingga meng-*handle* event.

Berdasarkan penelitian dan referensi yang telah ada sebelumnya khususnya dalam pengembangan sistem komunikasi antar aplikasi dan juga penerapan arsitektur layanan yang dapat mengurangi penggunaan *resource*, sehingga dilaksanakan penelitian ini dengan melakukan perancangan modul mahasiswa pada layanan SIMAK-NG Universitas Udayana yang dibangun dengan menggunakan bahasa pemrograman Java dan *framework* Quarkus.

II. TINJAUAN PUSTAKA

Perancangan RESTful API dilaksanakan dengan menggunakan bahasa pemrograman Java, *framework* Quarkus yang didukung dengan *platform* Intellij sebagai *Integrated Development Environment Application,* dan MySQL sebagai *database*.

A. Arsitektur API

1) RPC: Remote Procedure Call merupakan suatu teknologi yang digunakan guna membantu kinerja pada sisi *client side* dan *server side* dalam hal komunikasi, serta dapat dilakukan dengan konsep yang sederhana. Terdapat dua jenis RPC, yaitu XML-RPC dan JSON-RPC. Perbedaan dari kedua jenis terletak pada media perpindahan datanya. XML-RPC menggunakan media berupa XML (*Extensible Markup Language*). Sedangkan JSON-RPC menggunakan JSON (*JavaScript Object Notation*) sebagai media untuk memindahkan suatu data.

2) SOAP: Simple Object Access Protocol adalah salah satu standar media pertukaran pesan berbasis XML melalui jaringan komputer, atau sistem operasi yang sama ataupun berbeda. Program yang berjalan pada satu sistem operasi menggunakan HTTP dan XML sebagai mekanisme pertukaran data dalam aplikasi yang berbeda, meskipun ada perbedaan dalam sistem operasi, teknologi, dan bahasa pemrograman.

3) RESTful API: Merupakan penerapan dari API (*Application Programming Interface*), sedangkan REST atau *Representational State Transfer* merupakan suatu model arsitektur komunikasi berbasis layanan web atau *webservice*. Aplikasi dengan arsitektur REST umumnya menggunakan HTTP sebagai protokol standar untuk mengkomunikasikan data dari aplikasi DBMS ke aplikasi pihak ketiga [10]. Adapun 4 komponen yang dimiliki oleh RESTful API yaitu: URL Design, HTTP *Verbs*, HTTP *Response Code*, dan *Format Response* [16], [17], [19].

B. Intellij IDEA

Integrated Development Environment Application dikembangkan oleh Jetbrains tersedia dalam 2 (dua) versi, yaitu versi *Community Edition* dengan lisensi *Apache* 2 *Licensed* yang bisa digunakan secara gratis dan versi *Ultimate Edition* untuk penggunaan secara komersial. IDE yang ditulis dengan menggunakan bahasa Java dan Kotlin ini versi pertamanya dirilis pada Januari 2011, dan langsung menjadi IDE pertama untuk pengembangan aplikasi berbasis Java, hal ini dikarenakan IDE ini mendukung navigasi kode tingkat lanjut dan kemampuan *refactoring* yang saling terintegrasi.

C. MySQL

Mysql merupakan layanan *database* yang menjadi elemen standar dan umum digunakan pada layanan aplikasi ataupun *webservice* namun menurut dokumentasi MySQL pada seri 8.0 ada pembaharuan terkait keamanan sehingga untuk melakukan koneksi ke *database* harus menggunakan *nativeuser* agar layanan dapat terhubung dengan *database*, yang mana ini berbeda dengan MySQL seri sebelumnya. MySQL mempunyai beberapa kelebihan dibanding database lain, diantaranya adalah MySQL sebagai Relation & Database Management System (DBMS & RDBMS), menerima *query* dalam bentuk *multi-threading*, mampu menyimpan data hingga *gigabyte*, dan MySQL merupakan *server database* yang *multi-user* yang berarti dapat diakses lebih dari satu pengguna [11].

D. Java

Java merupakan bahasa pemrograman yang dapat dijalankan di berbagai perangkat seperti komputer pribadi hingga *smartphone*. Bahasa ini awalnya dikembangkan oleh James Gosling di *Sun Microsystems* yang sekarang menjadi bagian dari Oracle, dan dirilis pada tahun 1995. Bahasa Java memiliki sintaksi model objek yang lebih sederhana dari bahasa C dan C++. Aplikasi berbasis Java biasanya dikompilasi dalam p-*code* (*bytecode*) dan dapat berjalan di berbagai *Java Virtual Machines* (JVM). Java merupakan suatu bahasa pemrograman yang memiliki sifat umum/non-spesifik (*general purpose*) [12].

E. Arsitektur Layanan

Kualitas layanan dapat dipengaruhi oleh arsitektur yang digunakan pada layanan, saat ini terdapat dua bentuk layanan arsitektur yang umumnya digunakan yaitu arsitektur *microservices* dan *monolithic* [13]. [20]*:*

1) Arsitektur Microservices: merupakan pendekatan *Software Development Life Cycle* (SDLC) yang didasarkan pada aplikasi dengan ukuran lebih besar yang dibangun sebagai kumpulan modul fungsional kecil. Modul fungsional ini dapat diterapkan secara independen, dapat diskalakan, menargetkan sasaran bisnis tertentu, dan berkomunikasi satu sama lain melalui protokol standar seperti *request*/*response* HTTP dengan API *source* [14], [15], [18].

2) Arsitektur Monolithic: merupakan suatu arsitektur yang keseluruhan kodenya akan dikompilasi menjadi satu aplikasi (menjadi satu *binary* atau *artifact*). Penggunaan arsitektur *monolithic* dapat memberikan penghematan dalam penggunaan *server* atau *service* sehingga tidak diperlukan server tambahan karena aplikasi ini telah mencakup seluruh kode yang dibutuhkan [19].

3) Perbandingan Arsitektur: monolithic memiliki arsitektur yang berbeda dengan *microservices* yaitu *microservices* membagi layanan menjadi beberapa *services* yang lebih kecil, sementara *monolithic* semua layanan dibebankan pada *service* yang sama. Tabel 1 merupakan perbandingan pada layanan *monolithic* dan *microservices*

TABEL I
PERBANDINGAN MONOLITH DAN MICROSERVICES

Monolithic	*Microservices*
Semakin kompleks sistem dan bertambahnya pengguna layanan akan berakibat pada beban *service* yang lebih berat jika dibandingkan *microservices.*	Layanan bersifat *scalable*, *secure*, dan *reliable* yang proses *scaling*-nya bisa menggunakan metode, *scaling up* dan *scaling side*.
Satu *server* untuk menangani semua layanan.	Setiap layanan memiliki infrastruktur sendiri.
Tidak bisa menggunakan modul yang tidak ter-*install* pada *server* utama, seperti menggunakan *no-sql* atau *sql*, *nodejs* dan lain sebagainya.	Setiap layanan memiliki infrastruktur sendiri sehingga dapat ketergantungan aplikasi dengan layanan yang lain.
Proses *update* berimbas ke	Proses pembaruan hanya

p-ISSN:1693 – 2951; e-ISSN: 2503-2372

seluruh layanan.	mempengaruhi layanan terkait.
Seluruh *code* atau *component* menjadi satu *server*.	*Code* berbasis layanan sehingga harus memiliki dokumentasi yang lebih baik dari infrastruktur *monolithic*
Latensi komunikasi antar modul sangat rendah karena berada dalam *server* yang sama.	Adanya kemungkinan komunikasi antar modul akan mengalami kegagalan sehingga perlu dan harus selalu mempersiapkan cara mengantisipasinya.

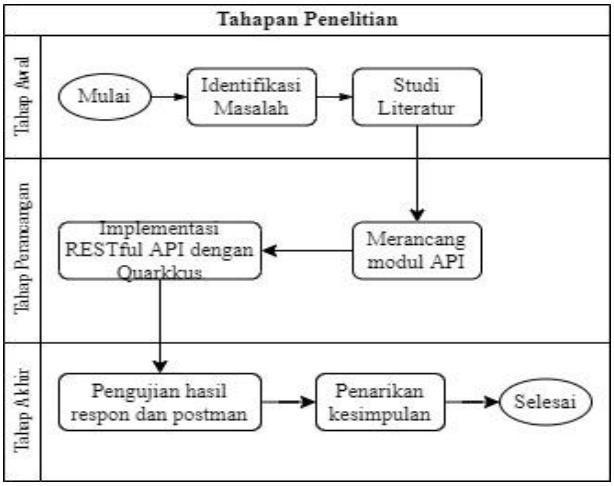
F. Quarkus

Pada penelitian ini *framework* yang digunakan adalah Quakurs, dimana Quarkus merupakan *framework* Java yang dirancang secara khusus untuk bekerja dengan optimal melalui Kubernetes dengan menghasilkan waktu *start up* dan waktu *response* yang lebih cepat ketika *request* pertama [21].

III. METODOLOGI PENELITIAN

Susunan tahap penelitian ditunjukkan pada Gambar 2, diawali dengan mengidentifikasikan masalah, yaitu melakukan diskusi mengenai kebutuhan yang diperlukan dalam penelitian. Tahapan selanjutnya adalah studi literatur yang dilakukan melalui buku ataupun sumber lain yang berkaitan dengan RESTful API, Quarkus, arsitektur layanan, dan lain-lain. Tahapan selanjutnya dilakukan dengan melakukan perancangan terhadap modul API yang dibangun dengan menggunakan bahasa pemrograman Java.

Tahapan implementasi RESTful API dengan Quarkus dilakukan dengan menjalankan program menggunakan *framework* Quarkus. Tahapan selanjutnya adalah pengujian yang dilakukan dengan membandingkan hasil rancangan dengan output yang didapat dan melakukan pengujian *script* melalui aplikasi Postman. Tahapan selanjutnya adalah penarikan kesimpulan dilakukan berdasarkan pengujian yang telah dilakukan.



Gambar 2: Tahapan Penelitian

A. Skema Database

Database digunakan untuk penyimpanan data dalam perancangan program API. Pada penelitian ini *database* yang dirancang adalah sebagai berikut.

1) *Tabel m_mahasiswa:* Tabel ini menyimpan data-data seputar mahasiswa.
2) *Tabel m_jurusan:* Tabel ini menyimpan seluruh jurusan yang terdapat pada Universitas Udayana.
3) *Tabel m_fakultas:* Tabel ini menyimpan seluruh fakultas yang terdapat pada Universitas Udayana.
4) *Tabel m_keaktifan:* Tabel ini menyimpan jenis status keaktifan yang terdapat pada Universitas Udayana.
5) *Tabel m_agama:* Tabel ini menyimpan berbagai macam Agama yang diakui di Indonesia.
6) *Tabel m_jenis_kelamin:* Tabel ini menyimpan data jenis kelamin.
7) *Tabel m_konsentrasi:* Tabel ini menyimpan data berbagai konsentrasi yang terdapat pada program studi setiap fakultas di Universitas Udayana.
8) *Tabel m_status_perkawinan:* Tabel ini menyimpan data status *pernikahan*.
9) *Tabel m_jalur_masuk:* Tabel ini menyimpan berbagai data jalur masuk di Universitas Udayana.
10) *Tabel m_negara:* Tabel ini menyimpan data nama negara.

B. Perancangan Modul Mahasiswa

Modul mahasiswa terdiri dari tiga buah fungsi dengan tugas yang berbeda, sehingga aplikasi lain yang memerlukan data mahasiswa pada simak tidak lagi melakukan akses secara langsung ke *database* namun akses dilakukan dengan menggunakan *endpoint* pada Tabel II. Tabel II merupakan daftar fungsi yang ada pada modul mahasiswa.

TABEL III
ENDPOINT DAN FUNGSI PADA MODUL

Jenis	Method	Endpoint	Fungsi
GET	getListMahasiswa	/mahasiswa	findAllMahasiswa
GET		/mahasiswa?nama=	findMahasiswaByQueryParam
GET	getBiodataMahasiswa	/mahasiwa/{id_mahasiswa}	findBiodataMahasiswa

Metode GET digunakan dalam perancangan penelitian ini dengan tiga buah fungsi yang berperan sebagai berikut.

1) *findAllMahasiswa:* fungsi ini memiliki peran untuk mendapatkan list mahasiswa yang datanya terdapat pada database
2) *findMahasiswaByQuery:* fungsi ini memiliki peran untuk mendapatkan seluruh data mahasiswa berdasarkan nama, sehingga akan dilakukan filter apabila namanya sesuai dengan query yang dilakukan
3) *findBiodataMahasiswa:* fungsi ini memiliki peran untuk mendapatkan data mahasiswa dengan spesifik dan

mendetail dengan menggunakan input id mahasiswa sebagai path parameter

C. Metode Pengujian

Pengujian dilakukan secara lokal dan terbatas pada fungsional modul untuk melihat apakah data yang didapatkan sudah sesuai dengan rancangan dan memperhatikan status *code* dan juga *response time* yang dihasilkan dari pengujian, berikut merupakan pengujian yang akan dilakukan untuk tiap fungsi:

1) Struktur response findAllMahasiswa: Tahap ini memeriksa apakah struktur *response* yang dihasilkan oleh program sudah sesuai dengan struktur yang dirancang. Listing Program 1 merupakan rancang struktur *response* yang akan dikerjakan

Listing Program 1: Rancang struktur *response* findAllMahasiswa

```
[
 {
  "id": integer,
  "id_jurusan": integer,
  "nama_jurusan": "string",
  "nama_fakultas": "string",
  "status_keaktifan_terakhir": integer,
  "nama_status_keaktifan_terakhir":
"string",
  "nim": "string",
  "nama": "string",
  "nama_tercetak": "string"
 },
 {
  Object lainnya
 }
]
```

2) Struktur response findMahasiswaByQueryParam: Tahap ini akan memeriksa apakah struktur *response* yang dihasilkan oleh program sudah sesuai dengan struktur yang dirancang. Listing Program 2 merupakan rancang struktur *response* yang akan dikerjakan

Listing Program 2: Rancang struktur *response* findMahasiswaByQueryParam

```
[
 {
  "id": integer,
  "id_jurusan": integer,
  "nama_jurusan": "string",
  "nama_fakultas": "string",
  "status_keaktifan_terakhir": integer,
  "nama_status_keaktifan_terakhir":
"string",
  "nim": "string",
  "nama": "string",
  "nama_tercetak": "string"
 },
 {
  Object lainnya
 }
]
```

3) Struktur response findBiodataById: Tahap ini akan memeriksa apakah struktur *response* yang dihasilkan oleh program sudah sesuai dengan struktur yang dirancang. Listing Program 3 merupakan rancang struktur *response* yang akan dikerjakan

Listing Program 3: Rancang struktur response findBiodataById

```
{
 "id": integer,
 "id jurusan": integer,
 "nama jurusan": "string",
 "nama fakultas": "string",
 "status keaktifan terakhir": integer,
 "nama_status_keaktifan_terakhir":
"string",
 "nim": "string",
 "nama": "string",
 "nama_tercetak": "string",
 "gelar_depan": "string",
 "gelar_belakang": "string",
 "tanggal lahir": "string",
 "tanggal lahir str": "string",
 "tempat lahir": "string",
 "alamat_asal": "string",
 "kecamatan": "string",
 "telepon_alamat_asal": "string",
 "alamat tinggal": "string",
 "telepon alamat tinggal": "string",
 "email": "string",
 "id_agama": integer,
 "nama_agama": "string",
 "id_jenis_kelamin": ineteger,
 "nama jenis kelamin": "string",
 "tanggal masuk": "string",
 "id konsentrasi": integer,
 "nama_konsentrasi": "string",
 "status_perkawinan": integer,
 "nama_status_perkawinan": "string",
 "id golongan darah": integer,
 "id_jalur_masuk": integer,
 "nama_jalur_masuk": "string",
 "ipk_akhir": integer,
 "index_capaian": integer,
 "id negara asal": integer,
 "nama negara asal": "string",
 "nama_ayah": "string",
 "nama_ibu": "string",
 "website": "string",
 "no_ktp": "string",
 "lama studi": integer,
 "image foto": "string",
 "file_ktp": "string"
}
```

4) Pengujian menggunakan script pada Postman*:* Pengujian ini akan dilakukan terhadap *response code* dan *response time* program dengan menggunakan *script* pada Postman melalui konfigurasi pada Listing Program 4

Listing Program 4: Script pengujian pada Postman

```
pm.test("Status code is 200", function
() {
      pm.response.to.have.status(200);
   }
);
```


p-ISSN:1693 – 2951; e-ISSN: 2503-2372

```
pm.test("Response time is less than 200ms", function() {
pm.expect(pm.response.responseTime).to.be.below(200);
});
```

IV. *HASIL DAN PEMBAHASAN*

Hasil dan pembahasan berdasarkan hasil yang diperoleh dari perancangan dan implementasi dijelaskan sebagai berikut.

A. Handler Modul

Satu *handler* dirancang, yaitu program dengan nama *file* MahasiswaResource. *Handler* berfungsi untuk mengatur bagaimana kode pada modul bekerja seperti yang terlihat pada Listing Program 5

Listing Program 5: Handler Modul mahasiswa

```
@Path("/mahasiswa")
@Tag(name = "Modul Mahasiswa", description = "Modul API ini digunakan untuk mendapatkan data mahasiswa")
@Produces(MediaType.APPLICATION_JSON)
public class MahasiswaResource {
 @Inject
 MySQLPool pool;
 @GET
 @Operation(
  summary = "getListMahasiswa",
  description = "endpoint ini akan menampilkan data mahasiswa dan dapat menggunakan query"
  )
  public Multi<ListMahasiswa> getListMahasiswa(@QueryParam("nama") String nama) {
   if (nama == null) {
    return ListMahasiswa.findAllMahasiswa(pool);
   } else {
    return ListMahasiswa.findMahasiswaByQueryParam(pool, nama);
   }
  }
 @GET
 @Path("/{id_mahasiswa}")
 @Operation(
  summary = "getBiodataMahasiswa",
  description = "endpoint ini akan menampilkan data lengkap dari mahasiswa tersebut"
  public Uni<Response> getBiodataMahasiswa(@PathParam("id_mahasiswa") Long id) {return BiodataMahasiswa.findBiodataById(pool, id)
   .onItem().transform(mahasiswa -> mahasiswa != null ? Response.ok(mahasiswa) : Response.status(Response.Status.NOT_FOUND))
   .onItem().transform(Response.ResponseBuilder::build);       }
}
```

Pada modul ini *path* diatur menjadi "/mahasiswa" yang memiliki dua buah *method* GET dengan nama operasi "getListMahasiswa" dan "getBiodataMahasiswa". Program akan memanggil fungsi "findAllMahasiswa" apabila menerapkan *method* GET pada *endpoint* "/mahasiswa" tanpa *query*, namun apabila pencarian dilakukan dengan menggunakan parameter *query* "/mahasiswa?nama=" maka akan memanggil fungsi "findMahasiswaByQueryParam" hal ini terjadi akibat *value* queryparam pada "nama" memiliki nilai atau bukan sama dengan *null* atau program akan memanggil fungsi "findBiodataById" apabila menerapkan *method* GET pada *endpoint* "/mahasiswa/{id_mahasiswa}".

B. Hasil Uji Fungsi findAllMahasiswa

Fungsi findAllMahasiswa yang merupakan turunan dari operasi getListMahasiswa memiliki program seperti pada Listing Program 6.

Listing Program 6: Fungsi findAllMahasiswa

```
public class ListMahasiswa {
 public Long id;
 public Integer id_jurusan;
 public String nama_jurusan;
 public String nama_fakultas;
 public Integer status_keaktifan_terakhir;
 public String nama_status_keaktifan_terakhir;
 public String nim;
 public String nama;
 public String nama_tercetak;
//skip to function
 public static Multi<ListMahasiswa> findAllMahasiswa(MySQLPool pool) {return pool.query("select
    m_mahasiswa.id,
    m_mahasiswa.id_jurusan,
    m_jurusan.nama_jurusan,
    m_fakultas.nama_fakultas,
    m_mahasiswa.status_keaktifan_terakhir,
    m_keaktifan.nama as nama_status_keaktifan_terakhir,
    m_mahasiswa.nim,
    m_mahasiswa.nama,
    m_mahasiswa.nama_tercetak
    from
    m_mahasiswa
    left join m_jurusan on
     m_mahasiswa.id_jurusan = m_jurusan.id
    left join m_fakultas on
     m_jurusan.id_fakultas = m_fakultas.id
    left join m_keaktifan on
     m_mahasiswa.status_keaktifan_terakhir = m_keaktifan.id
    order by
    m_mahasiswa.id desc").execute()
     .onItem().transformToMulti(set -> Multi.createFrom().iterable(set))
     .onItem().transform(ListMahasiswa::from);
}
```

Fungsi findAllMahasiswa bekerja dengan melakukan *request* menuju *endpoint* "/mahasiswa" yang mana fungsi ini akan bekerja untuk mendapatkan seluruh data mahasiswa yang terdapat pada *database* dengan melakukan SQL *join* dengan beberapa tabel antara lain tabel m_jurusan, m_fakultas, dan m_keaktifan. *Join* dilakukan guna mendapatkan data yang tidak tersedia pada tabel m_mahasiswa, seperti nama_jurusan, nama_fakultas, dan nama_status_keaktifan_terakhir. Listing Program 7 merupakan hasil *response* yang didapat melalui fungsi findAllMahasiswa

Listing Program 7: Hasil response findAllMahasiswa

```
[
 {
   "id": ******,
   "id_jurusan": 4,
   "nama_jurusan": "Teknik Elektro",
   "nama_fakultas": "Fakultas Teknik",
   "status_keaktifan_terakhir": 1,
   "nama_status_keaktifan_terakhir": "Aktif",
   "nim": "180554****",
   "nama": "I Kadek Ari Wijaya",
   "nama_tercetak": "I Kadek Ari Wijaya"
 },
 {
  Object lainnya
 }
```

```
]
```

Perbandingan antara perancangan struktur *response* fungsi findAllMahasiswa pada Listing Program 1 dan hasil yang didapat pada Listing Program 7 menunjukkan bahwa kode program sudah mendapatkan hasil response yang sesuai dengan perancangan.

C. Hasil Uji Fungsi findMahasiswaByQuery

Fungsi findMahasiswaByQuery yang merupakan turunan dari operasi getBiodataMahasiswa memiliki program seperti pada Listing Program 8.

Listing Program 8: Fungsi findMahasiswaByQuery

```
public class ListMahasiswa {
 public Long id;
 public Integer id_jurusan;
 public String nama_jurusan;
 public String nama_fakultas;
 public Integer status_keaktifan_terakhir;
 public String nama_status_keaktifan_terakhir;
 public String nim;
 public String nama;
 public String nama_tercetak;
//skip to function
 public static Multi<ListMahasiswa>
findMahasiswaByQueryParam(MySQLPool pool, String
nama) {
  String namasql = "'%" + nama + "%'";
   return pool.query("SELECT m_mahasiswa.id," +
    "m_mahasiswa.id_jurusan," +
    "m_jurusan.nama_jurusan," +
    "m_fakultas.nama_fakultas," +
    "m_mahasiswa.status_keaktifan_terakhir," +
    "m_keaktifan.nama as
nama_status_keaktifan_terakhir," +
    "m_mahasiswa.nim," +
    "m_mahasiswa.nama," +
    "m_mahasiswa.nama_tercetak " +
    "FROM m_mahasiswa " +
    "LEFT JOIN m_jurusan ON m_mahasiswa.id_jurusan =
m_jurusan.id " +
    "LEFT JOIN m_fakultas ON m_jurusan.id_fakultas =
m_fakultas.id " +
    "LEFT JOIN m_keaktifan ON
m_mahasiswa.status_keaktifan_terakhir =
m_keaktifan.id " +
    "WHERE m_mahasiswa.nama LIKE "+namasql+
    " ORDER BY nim DESC").execute()
     .onItem().transformToMulti(set ->
Multi.createFrom().iterable(set))
     .onItem().transform(ListMahasiswa::from);
}
```

Fungsi findMahasiswaByQuery bekerja dengan melakukan *request* menuju *endpoint* "/mahasiswa?nama=" yang mana fungsi ini akan bekerja untuk mendapatkan seluruh data mahasiswa yang sesuai dengan *query param* yang diterapkan dan terdapat pada *database* dengan melakukan SQL *join* dengan beberapa tabel antara lain tabel m_jurusan, m_fakultas, dan m_keaktifan. *Join* dilakukan guna mendapatkan data yang tidak tersedia pada tabel m_mahasiswa, seperti nama_jurusan, nama_fakultas, dan nama_status_keaktifan_terakhir. Listing Program 9 merupakan hasil *response* yang didapat melalui fungsi findMahasiswaByQuery.

Listing Program 9: Hasil *response* findMahasiswaByQuery

```
[
 {
  "id": ******,
  "id_jurusan": 4,
  "nama_jurusan": "Teknik Elektro",
  "nama_fakultas": "Fakultas Teknik",
  "status_keaktifan_terakhir": 1,
  "nama_status_keaktifan_terakhir":
"Aktif",
  "nim": "180554****",
  "nama": "I Kadek Ari Wijaya",
  "nama_tercetak": "I Kadek Ari Wijaya"
 },
 {
  Object lainnya dengan query serupa
 }
]
```

Perbandingan antara perancangan struktur *response* fungsi findMahasiswaByQuery pada Listing Program 2 dan hasil yang didapat pada Listing Program 9 menunjukkan bahwa kode program sudah mendapatkan hasil response yang sesuai dengan perancangan.

D. Hasil Uji Fungsi findBiodataById

Fungsi findBiodataById yang merupakan turunan dari operasi getBiodataMahasiswa memiliki program seperti pada Listing Program 10.

Listing Program 10: Fungsi findMahasiswaById

```
public class BiodataMahasiswa {
 public Long id;
 public Integer id_jurusan;
 public String nama_jurusan;
 public String nama_fakultas;
 public Integer status_keaktifan_terakhir;
 public String nama_status_keaktifan_terakhir;
 public String nim;
 public String nama;
 public String nama_tercetak;
 public String gelar_depan;
 public String gelar_belakang;
 public LocalDate tanggal_lahir;
 public String tanggal_lahir_str;
 public String tempat_lahir;
 public String alamat_asal;
 public String kecamatan;
 public String telepon_alamat_asal;
 public String alamat_tinggal;
 public String telepon_alamat_tinggal;
 public String email;
 public Integer id_agama;
 public String nama_agama;
 public Integer id_jenis_kelamin;
 public String nama_jenis_kelamin;
 public LocalDate tanggal_masuk;
 public Integer id_konsentrasi;
 public String nama_konsentrasi;
 public Integer status_perkawinan;
 public String nama_status_perkawinan;
```


p-ISSN:1693 – 2951; e-ISSN: 2503-2372

```
 public Integer id_golongan_darah;
 public Integer id_jalur_masuk;
 public String nama_jalur_masuk;
 public Float ipk_akhir;
 public Float index_capaian;
 public Integer id_negara_asal;
 public String nama_negara_asal;
 public Integer angkatan;
 public String nama_ayah;
 public String nama_ibu;
 public String website;
 public String no_ktp;
 public Integer lama_studi;
 public String image_foto;
 public String file_ktp;
//skip to function
 public static Uni<BiodataMahasiswa>
findBiodataById(MySQLPool pool, Long id)
{return pool.preparedQuery("SELECT
  m_mahasiswa.id," +
   "m_mahasiswa.id_jurusan," +
   "m_jurusan.nama_jurusan," +
   "m_fakultas.nama_fakultas," +
   "m_mahasiswa.status_keaktifan_terakhir," +
   "m_keaktifan.nama as
nama_status_keaktifan_terakhir," +
   "m_mahasiswa.nim," +
   "m_mahasiswa.nama," +
   "m_mahasiswa.nama_tercetak," +
   "m_mahasiswa.gelar_depan," +
   "m_mahasiswa.gelar_belakang," +
   "m_mahasiswa.tanggal_lahir," +
   "m_mahasiswa.tanggal_lahir_str," +
   "m_mahasiswa.tempat_lahir," +
   "m_mahasiswa.alamat_asal," +
   "m_mahasiswa.kecamatan," +
   "m_mahasiswa.telepon_alamat_asal," +
   "m_mahasiswa.alamat_tinggal," +
   "m_mahasiswa.telepon_alamat_tinggal," +
   "m_mahasiswa.email," +
   "m_mahasiswa.id_agama," +
   "m_agama.nama as nama_agama," +
   "m_mahasiswa.id_jenis_kelamin," +
   "m_jenis_kelamin.nama as
nama_jenis_kelamin," +
  "m_mahasiswa.tanggal_masuk," +
   "m_mahasiswa.id_konsentrasi," +
   "m_konsentrasi.nama_konsentrasi," +
   "m_mahasiswa.status_perkawinan," +
   "m_status_perkawinan.nama as
nama_status_perkawinan," +
   "m_mahasiswa.id_golongan_darah," +
   "m_mahasiswa.id_jalur_masuk," +
   "m_jalur_masuk.jalur_masuk as
nama_jalur_masuk," +
   "m_mahasiswa.ipk_akhir," +
   "m_mahasiswa.index_capaian," +
   "m_mahasiswa.id_negara_asal," +
   "m_negara.nama as nama_negara_asal," +
   "m_mahasiswa.angkatan as angkatan,"+
   "m_mahasiswa.nama_ayah," +
   "m_mahasiswa.nama_ibu," +
   "m_mahasiswa.website," +
   "m_mahasiswa.no_ktp," +
   "m_mahasiswa.lama_studi," +
   "m_mahasiswa.image_foto," +
   "m_mahasiswa.file_ktp " +
   "FROM m_mahasiswa " +
   "LEFT JOIN m_jurusan ON
m_mahasiswa.id_jurusan = m_jurusan.id " +
   "LEFT JOIN m_fakultas ON
m_jurusan.id_fakultas = m_fakultas.id " +
   "LEFT JOIN m_keaktifan ON
m_mahasiswa.status_keaktifan_terakhir =
m_keaktifan.id " +
   "LEFT JOIN m_agama ON m_mahasiswa.id_agama =
m_agama.id " +
   "LEFT JOIN m_jenis_kelamin ON
m_mahasiswa.id_jenis_kelamin =
m_jenis_kelamin.id " +
   "LEFT JOIN m_konsentrasi ON
m_mahasiswa.id_konsentrasi = m_konsentrasi.id
" +
   "LEFT JOIN m_status_perkawinan ON
m_mahasiswa.status_perkawinan =
m_status_perkawinan.id " +
   "LEFT JOIN m_jalur_masuk ON
m_mahasiswa.id_jalur_masuk = m_jalur_masuk.id
" +
   "LEFT JOIN m_negara ON
m_mahasiswa.id_negara_asal = m_negara.id " +
   "WHERE m_mahasiswa.id
LIKE ?").execute(Tuple.of(id))
    .onItem().transform(RowSet::iterator)
    .onItem().transform(iterator ->
iterator.hasNext() ? from(iterator.next()) :
null);
}
```

Fungsi findMahasiswaById bekerja dengan melakukan *request* menuju *endpoint* "/mahasiswa/{id_mahasiswa}" yang mana fungsi ini akan bekerja untuk mendapatkan data mahasiswa secara spesifik sesuai dengan id_mahasiswa yang diterapkan sebagai *pathparam* dan terdapat pada *database* dengan melakukan SQL *join* dengan beberapa tabel antara lain tabel m_jurusan, m_fakultas, m_keaktifan, m_agama, m_jenis_kelamin, m_konsentrasi, m_status_perkawinan, m_jalur_masuk, dan m_negara. *Join* dilakukan guna mendapatkan data yang tidak tersedia pada tabel m_mahasiswa, seperti nama_jurusan, nama_fakultas, nama_status_keaktifan_terakhir, nama_jenis_kelamin, nama_status_perkawinan, dan nama jalur_masuk. Listing Program 11 merupakan hasil *response* yang didapat melalui fungsi findMahasiswaById.

Listing Program 11: Hasil *response* findMahasiswaById

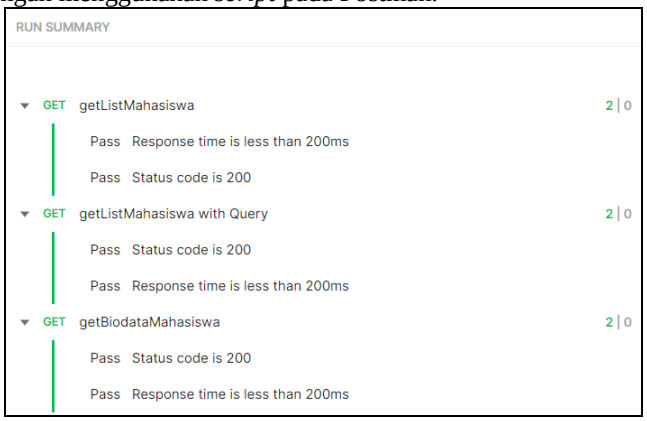
```
{
 "id": ******,
 "id_jurusan": 4,
 "nama_jurusan": "Teknik Elektro",
 "nama_fakultas": "Fakultas Teknik",
 "status_keaktifan_terakhir": 1,
 "nama_status_keaktifan_terakhir":
"Aktif",
 "nim": "180554****",
 "nama": "I Kadek Ari Wijaya",
 "nama_tercetak": "I Kadek Ari Wijaya",
 "gelar_depan": null,
 "gelar_belakang": null,
 "tanggal_lahir": "******",
 "tanggal_lahir_str": "*******",
 "tempat_lahir": "******",
 "alamat_asal": "******",
 "kecamatan": "*******",
 "telepon_alamat_asal": "******",
 "alamat_tinggal": "******",
 "telepon_alamat_tinggal": "******",
 "email": "******",
 "id_agama": 4,
 "nama_agama": "Hindu",
 "id_jenis_kelamin": 1,
 "nama_jenis_kelamin": "Laki",
 "tanggal_masuk": "2018-07-20",
 "id_konsentrasi": 652,
 "nama_konsentrasi": null,
 "status_perkawinan": 2,
 "nama_status_perkawinan": "Belum
Kawin",
```

```
  "id_golongan_darah": 1,
  "id_jalur_masuk": 5,
  "nama_jalur_masuk": "SBMPTN",
  "ipk_akhir": 3.46,
  "index_capaian": 25.164,
  "id_negara_asal": 63,
  "nama_negara_asal": "Indonesia",
  "angkatan": 2018,
  "nama_ayah": "I Ketut Dangin",
  "nama_ibu": "Ni Nyoman Rastiti",
  "website": "",
  "no_ktp": "******",
  "lama_studi": 1,
  "image_foto": "******",
  "file_ktp": "******"
}
```


Perbandingan antara perancangan struktur *response* fungsi findMahasiswaById pada Listing Program 3 dan hasil yang didapat pada Listing Program 11 menunjukkan bahwa kode program sudah mendapatkan hasil response yang sesuai dengan perancangan.

E. Hasil Pengujian Script

Pengujian dilakukan terhadap modul mahasiswa dengan menggunakan *script* dan dinyatakan bahwa program sudah berhasil pada kedua *test* yang dilakukan seperti yang terlihat pada Gambar 3 merupakan hasil pengujian yang didapat dengan menggunakan *script* pada Postman.



Gambar 3: Hasil Pengujian Script

Berdasarkan pengujian yang telah dilakukan dapat dilihat bahwa kode program yang dikerjakan telah berhasil berjalan ditandai dengan *response code* 200 dan memiliki *response time* dibawah 200 ms.

V. KESIMPULAN

Perancangan program dengan arsitektur RESTful API untuk layanan SIMAK-NG Universitas Udayana khususnya modul mahasiswa telah berhasil dirancang dan diimplementasikan pada *server local*. Perancangan struktur program pada fungsi modul mahasiswa dirancang dengan menggunakan penerapan bahasa pemrograman Java, *framework* Quarkus, dan *database* MySQL.

RESTful API yang telah dirancang dapat digunakan sebagai sarana atau media komunikasi data antar modul pada layanan SIMAK-NG Universitas Udayana sehingga aplikasi lain yang memerlukan data mahasiswa pada layanan SIMAK-NG dapat diakses dengan menggunakan REStful API ini. Untuk mengimplementasikan perancangan ini perlu dilakukan beberapa penyesuaian antara lain menyiapkan *database* dengan skema yang sesuai.

Berdasarkan pengujian yang dilakukan dengan melakukan perbandingan *response* yang didapat dengan rancangan yang dibuat, terlihat bahwa struktur *respone* yang dihasilkan sudah sesuai dengan perancangan. Pada pengujian dengan menggunakan *script* pada Postman didapatkan bahwa program sudah bekerja dengan baik hal ini ditandakan dengan dua buah pengujian yang dilaksanakan pada tiap fungsi telah berhasil sebesar 100% yang didasarkan pada status hit yang berhasil menghasilkan *response code* 200 serta memiliki *response time* yang berada dibawah 200 ms.

REFERENSI

[1] Varajão, J. (2018). The many facets of information systems (+ projects) success. International Journal of Information Systems and Project Management, 5-13. https://doi.org/10.12821/ijispm060401.

[2] Almaiah, M. A., & Man, M. (2016). Empirical investigation to explore factors that achieve high quality of mobile learning system based on students' perspectives. Engineering Science and Technology an International Journal, 1314-1320.

[3] Sligo, J., Gauld, R., Roberts, V., & Villa, L. (2017). A literature review for large-scale health information system project planning, implementation and evaluation. International journal of medical informatics, 86-97.

[4] Choirudin, R (2019). Implementasi REST API Web Service Dalam Membangun Aplikasi Multiplatform Untuk Usaha Jasa. https://doi.org/10.30812/matrik.v18i2.407

[5] Michal ,B., Marek, p., & Piotr, K. (2021). Comparison of lightweight frameworks for Java by analyzing proprietary web applications. http://dx.doi.org/10.35784/jcsi.2645

[6] Grgic, H., Mihaljević, B., & Radovan, A. (2018). Comparison of garbage collectors in Java programming language. https://doi.org/10.23919/MIPRO.2018.8400277

[7] Goncalves, A., & Escoffier, C. (2020). Practising Quarkus. RedHat Developer.

[8] Goncalves, A., & Escoffier, C. (2020). Understanding Quarkus. RedHat Developer

[9] Deandrea, E., Oh, D., Mouliiard, C., & Verburg, M. (2021). Quarkus for Spring Developer. RedHat Developer.

[10] Manuaba, I., & Rudiastini, E. (2018). API REST Web service and backend system Of Lecturer's Assessment Information System on Politeknik Negeri Bali. Journal of Physics: Conference Series, 1-7. http://dx.doi.org/10.1088/1742-6596/953/1/012069

[11] Nugroho, B. (2004). PHP dan MySQL dengan EditorDreamweaverMX. Yogyakarta: Andi.

[12] Maria.W.H Barri, A. S. (2015). Perancangan Aplikasi SMS GATEWAY Untuk Pembuatan Kartu Perpustakaan di Fakultas Teknik Unsrat. E-journal Teknik Elektro dan Komputer, 25.

[13] Martin Fowler (2014) Microservices. [Online]. Available: https://www.martinfowler.com/articles/microservices.html?ref=wellarchitected

[14] P. Giessler, M. Gebhart, D. Sarancin, R. Steinegger, S. Abeck, 2015, Best Practices for the Design of RESTful Web Services, International Conferences of Software Advances (ICSEA).
[15] Nicola Dragoni, Saverio Giallorenzo, Alberto Lafuente, Manuel Mazzara, Fabrizio Montesi, et al.. Microservices: yesterday, today, and tomorrow. Manuel Mazzara; Bertrand Meyer. Present and Ulterior Software Engineering, Springer, 2017, 978-3-319-67425-4. hal-01631455
[16] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Technical Report RFC 2616, The Internet Society, http://www.ietf.org/rfc/rfc2616.txt, 1999.
[17] R. Fielding. Architectural Styles and the Design of Network-based Software Architectures. Ph.d. dissertation, University of California, Irvine, 2007.
[18] Y. Gan and C. Delimitrou, "The Architectural Implications of Cloud Microservices," IEEE Computer Architecture Letters, vol. 17, no. 2, pp. 155-158, 2018.
[19] JULIAWAN PAWANA, I Wayan Adi; WIHARTA, Dewa Made; SASTRA, Nyoman Putra. Identifikasi Kandidat Microservices Dengan Analisis Domain Driven Design. Majalah Ilmiah Teknologi Elektro, [S.l.], v. 20, n. 2, p. 273-280, dec. 2021. ISSN 2503-2372. Available at: <https://ojs.unud.ac.id/index.php/jte/article/view/73745>. Date accessed: 31 aug. 2022. doi: https://doi.org/10.24843/MITE.2021.v20i02.P11.
[20] KURNIAWAN, Ketut Adi; PUTRA SASTRA, N; SUDARMA, M. Analisis Performansi Dan Efisiensi Cloud Computing Pada Sistem Perbankan. Majalah Ilmiah Teknologi Elektro, [S.l.], v. 19, n. 1, p. 11-18, oct. 2020. ISSN 2503-2372. Available at: <https://ojs.unud.ac.id/index.php/jte/article/view/53917>. Date accessed: 31 aug. 2022. doi: https://doi.org/10.24843/MITE.2020.v19i01.P02.
[21] Koleoso, Tayo. (2020). Microservices with Quarkus. 10.1007/978-1-4842-6032-6_3.