JURNAL PINTER
VOL 5. NO. 1 JUNI 2021
E - ISSN: 2597 - 4475
http://doi.org.10.21009/pinter.5.1.4

PENGEMBANGAN *WEB SERVICE* SISTEM PEMBAYARAN *MULTIBANK* UNIVERSITAS NEGERI JAKARTA

Achmad Ahlar Ridha[1], Hamidillah Ajie[2], M. Ficky Duskarnaen[3]

[1] Mahasiswa Prodi Pendidikan Teknik Informatika dan Komputer, Teknik Elektro, FT – UNJ
[2,3] Dosen Prodi Pendidikan Teknik Informatika dan Komputer, Teknik Elektro, FT – UNJ
[1] aar.frontline@gmail.com, [2] hamidillah@unj.ac.id, [3] duskarnaen@unj.ac.id

Abstrak

Transaksi pembayaran di Universitas Negeri Jakarta bekerja sama dengan empat bank. Meski pun bekerja sama dengan empat bank, sistem pembayaran yang ada tidak bersifat multibank. Masing-masing bank secara terpisah menangani jenis pembayaran mahasiswa berdasarkan fakultas dan/atau tahun masuk. Hal ini dapat mengurangi fleksibilitas bank sebagai media pembayaran. Perlu adanya sistem yang menangani setiap pembayaran secara luwes dan terpadu, serta mampu beradaptasi dengan perubahan baik dalam hal pengembangan maupun kerja sama dalam hal bisnis. Tujuan dari penelitian ini adalah untuk mengembangkan dan menghasilkan web service yang dapat digunakan untuk mengimplementasikan multibank di Universitas Negeri Jakarta dan bersifat adaptif terhadap perubahan-perubahan yang akan datang. Web Service Multibank dikembangkan dengan konsep Service Oriented Architecture, model pengembangan spiral, ruang lingkup Node.js, framework Express, dan database MySQL menghasilkan 8 unit endpoint. Unit Testing akan digunakan sebagai metode pengujian terhadap 8 unit endpoint yang dikembangkan. Hasil pengujian menunjukan bahwa 8 unit endpoint yang telah diuji berfungsi dengan baik dan layak digunakan sebagai web service untuk sistem multibank di Universitas Negeri Jakarta.

Kata kunci : *Multibank*, *Web Service*, *Service Oriented Architecture*, Model Spiral, Node.js, ExpressJS, MySQL, *Unit Testing*, Universitas Negeri Jakarta.

1. Pendahuluan

1.1. Latar Belakang

Universitas Negeri Jakarta (UNJ) merupakan salah satu Perguruan Tinggi Negeri (PTN). Program pendidikan dilakukan dengan kegiatan pembelajaran terhadap jenjang Diploma, Sarjana, Magister, hingga Doktor. Program tersebut dilakukan dalam menunjang pelayanan pendidikan masyarakat di Indonesia.

Biaya operasional dalam rangka pelaksanaan kegiatan dan pembangunan pada PTN menerapkan sistem Biaya Kuliah Tunggal (BKT) dan Uang Kuliah Tunggal (UKT) yang ditetapkan berdasarkan Peraturan Menteri Riset, Teknologi, dan Pendidikan Tinggi (PERMENRISTEKDIKTI) Nomor 39 Tahun 2017 yang berisi bahwa BKT merupakan keseluruhan biaya operasional mahasiswa per semester, sedangkan UKT merupakan biaya yang akan ditanggung atau dibayarkan setiap mahasiswa. UKT merupakan sebagian dari dana masyarakat yang diatur oleh pemerintah untuk memenuhi kebutuhan pendidikan yang di Indonesia.

UKT terdiri atas beberapa kelompok yang diusulkan oleh pihak PTN untuk ditetapkan oleh menteri RISTEKDIKTI. UKT dikelompokan berdasarkan kemampuan ekonomi mahasiswa atau orang tua/wali yang membiayainya. Pimpinan PTN dapat memberlakukan UKT yang sesuai dengan kondisi ekonomi mahasiswa. UKT tersebut selanjutnya dibayarkan oleh mahasiswa sesuai dengan media pembayaran yang telah ditentukan oleh PTN terkait seperti bank.

Sistem pembayaran di UNJ menerapkan sistem yang dinamakan Host to Host (H2H). H2H merupakan sebuah solusi otomatis untuk melakukan pembayaran dengan media transfer data elektronik yang aman antara bank dengan klien korporat. H2H memiliki beberapa kelebihan seperti Seamless, Security, Multi Transaction, Currency Support, dan lain-lain.

Dikutip dari beberapa aplikasi web UNJ, kelompok pembayaran di UNJ saat ini terbagi menjadi beberapa kelompok. Pembayaran jenjang Strata-1 (S1) untuk angkatan 2016 ke bawah, dibayarkan melalui sistem H2H BNI. Seluruh angkatan jenjang Diploma-3, dan seluruh jenjang S1 pada angkatan 2016 ke atas yakni: 1) Fakultas Ilmu Sosial (FIS); 2) Fakultas Teknik; dan 4) Fakultas Ekonomi, menggunakan sistem H2H Bank Mandiri. Untuk seluruh jenjang S1 pada angkatan 2016 ke atas yakni: 1) Fakultas Ilmu Olahraga (FIO); 2) Fakultas Bahasa dan Seni (FBS); 3) Fakultas Matematika; 4) Ilmu Pengetahuan Alam (FMIPA); 5) Fakultas Ilmu Pendidikan (FIP); dan 6) Fakultas Pendidikan Psikologi (FPPsi), menggunakan sistem H2H BTN.

Sedangkan pembayaran untuk jenjang Pascasarjana, menggunakan H2H Bank Bukopin. Setiap sistem H2H tersebut akan dikembangkan dan diteliti untuk pengembangan sistem pembayaran multibank. Sistem pembayaran ini dikembangkan oleh Unit Pelaksana Teknis Teknologi Informasi dan Komunikasi (UPT TIK) yang ada di UNJ.

Untuk meningkatkan layanan agar dapat menerapkan sistem pembayaran yang fleksibel dan mudah untuk dikontrol, UPT TIK UNJ berencana mengembangkan sistem multibank yang berbasis *web service*.

Pengembangan *web service* akan membentuk sebuah aplikasi backend dengan implementasi arsitektur Service-Oriented Architecture (SOA) dan data interchange format JSON. Arsitektur atau gaya penulisan SOA pada sebuah *web service* merupakan sekumpulan dari beberapa alamat terakhir yang disebut sebagai endpoint dari sebuah URL. Selain arsitektur, *web service* juga akan mengimplementasikan bahasa pemrograman JavaScript. Dengan bantuan teknologi Node.js, aplikasi bahasa JavaScript dapat bekerja diluar browser, sehingga pengembangan *web service* dengan JavaScript tidak menjadi mustahil. *Web Service* yang dikembangkan, menggunakan salah satu framework JavaScript yaitu Express dalam menerapkan arsitektur SOA secara teknisnya.

1.2. Identifikasi Masalah

Berdasarkan latar belakang yang telah diuraikan, dapat diidentifikasi permasalahan-permasalahan yang ada pada berikut ini:

1. Meskipun sudah bekerja sama dengan empat bank, kebijakan yang ditetapkan untuk setiap mahasiswa yaitu hanya dapat membayar pada bank yang telah ditentukan;
2. Masing-masing H2H memiliki perbedaan struktur *database*, sehingga menyulitkan pada saat melacak dan mengambil data;
3. Belum adanya penerapan sistem multibank di Universitas Negeri Jakarta

1.3. Batasan Masalah

Melihat luas ruang lingkup dari permasalahan yang diidentifikasi sebelumnya, batasan masalah yang dapat diidentifikasi adalah:

1. Penelitian menyelaraskan struktur *database* dan menyatukan setiap H2H menjadi sebuah *web service*;
2. Fokus penelitian hanya pada pengembangan *web service*, tidak menghasilkan aplikasi yang lengkap.

1.4. Rumusan Masalah

Berdasarkan latar belakang, identifikasi masalah, dan pembatasan masalah, maka perumusan masalah yang akan dibahas pada penelitian ini adalah "Bagaimana Mengembangkan *web service* untuk sistem pembayaran multibank di Universitas Negeri Jakarta?".

1.5. Tujuan Penelitian

Berdasarkan rumusan masalah pada pembahasan sebelumnya, tujuan dari penelitian ini adalah:

1. Meskipun sudah bekerja sama dengan empat bank, kebijakan yang ditetapkan untuk setiap mahasiswa yaitu hanya dapat membayar pada bank yang telah ditentukan;

1.6. Manfaat Penelitian

Berdasarkan rumusan masalah pada pembahasan sebelumnya, tujuan dari penelitian ini adalah:

1. Secara teoritis, hasil penelitian ini diharapkan dapat menjadi sumber kajian dan bahan referensi penelitian lainnya, terkait dengan bidang ilmu pengembangan *database*, *web service* dan teknologi *web* yang lain;
2. Secara praktis, hasil aplikasi penelitian ini dapat digunakan dan disinkronkan dengan aplikasi *frontend*, sehingga membentuk sebuah aplikasi yang lengkap.

2. Dasar Teori

2.1. Bank

Kata bank dipinjam dari bahasa Itali kuno "banca" atau bahasa pertengahan Perancis "banque" yang sama-sama memiliki arti meja, diambil dari bahasa Jerman Kuno "banc" yang berarti bangku. Gagasan etimologis dari kata bank adalah merupakan sebuah meja pertukaran peminjam uang atau rentenir (*www.etymonline.com*).

Menurut Undang-Undang (UU) RI Nomor 10 Pasal 1 Tahun 1998, yang dimaksud dengan Bank adalah "badan usaha yang menghimpun dana dari masyarakat dalam bentuk kredit dan/atau bentuk-bentuk lainnya dalam rangka meningkatkan taraf hidup rakyat banyak."

Menurut Bustari M. (2016: 53), bank secara sederhana dapat diartikan sebagai lembaga yang kegiatan utamanya adalah menghimpun dana dari masyarakat dan menyalurkannya kembali dana tersebut ke masyarakat serta memberikan jasa bank lainnya.

2.2. Sistem Pembayaran di Universitas Negeri Jakarta

Sistem Pembayaran di UNJ saat ini menggunakan sistem pengelompokan pembayaran bank untuk setiap jenjang dan angkatannya. Pengelompokan tersebut dianggap dapat mempermudah rekapitulasi dan rekonsiliasi per jenjang dan angkatan. Pengelompokan juga

dilakukan untuk memudahkan sistem dalam memberikan basis data secara independen.

UNJ saat ini menggunakan H2H untuk menunjang sistem pembayarannya. Setiap H2H merupakan sebuah *database* yang dapat diakses oleh setiap bank yang telah bekerja sama (empat bank sama dengan empat unit H2H). Masing-masing H2H memiliki *server* dan struktur *database* yang berbeda-beda. Setiap struktur *database* H2H merepresentasikan informasi dan kebutuhan apa saja yang diperlukan pada setiap bank yang terkait. Perbedaan antara H2H satu bank dengan bank lainnya menimbulkan proses seperti rekapitulasi data yang dapat memakan waktu.

2.3. Sistem Pembayaran *Multibank*

Dikutip dari *www.thefreedictionary.com* dalam buku Collins English Dictionary, *multibank* terdiri dari dua subyek kategori yaitu Banking Finance dan Computer Science. Dalam kedua subyek tersebut, multibank memiliki makna yaitu melibatkan lebih dari satu bank. Dalam kamus *www.oxforddictionaries.com*, *multi-* yang berarti lebih dari satu, serta bank yang merupakan "sebuah perusahaan keuangan yang menggunakan uang yang disimpan oleh pengguna layanan untuk diinvestasikan, dibayarkan pada saat diperlukan, memberikan pinjaman dengan bunga, dan menukar mata uang".

Multibank atau *Multi-bank* dapat disimpulkan sebagai kegiatan yang melibatkan lebih dari satu bank untuk pelayanan yang melibatkan keuangan seperti investasi, pembayaran, pinjaman, dan penukaran.

2.4. Sistem *Host to Host* (H2H)

Dikutip dari situs *www.mindgate.in*, *Host-to-Host* (H2H) merupakan sebuah solusi otomatis untuk transfer data elektronik yang aman antara bank dengan klien korporat yang bekerja sama.

Dalam laman *www.bankmandiri.co.id*, H2H adalah layanan pengiriman instruksi pemindahan dana/ pembayaran yang terintegrasi langsung dari sistem nasabah *Enterprise Resource Planning* (ERP) ke bank.

2.5. *Website*

Website menurut Kadir (2003: 374), merupakan kumpulan beberapa halaman pribadi, organisasi, atau perusahaan yang menampilkan dan memuat informasi dalam *web server* (suatu unit komputer yang berfungsi untuk menyimpan informasi dan mengelola jaringan komputer).

Menurut Berners Lee, dkk (1989), *website* adalah suatu halaman *web* yang saling terhubung yang pada umumnya berada pada *server* yang sama, berisikan sekumpulan informasi yang disediakan secara perorangan, kelompok, atau organisasi.

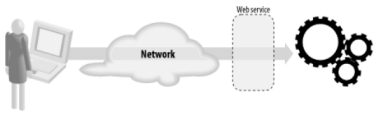
Website merupakan sekumpulan halaman yang saling berhubungan, umumnya berlokasi pada *server* yang sama, disediakan dan dikelola sebagai koleksi informasi seseorang, grup, atau organisasi (*www.thefreedictionary.com*).

Berdasarkan beberapa pengertian tentang *website* yang telah disebutkan, dapat disimpulkan bahwa *website* merupakan sekumpulan halaman yang berada pada satu *server* dan memiliki kumpulan informasi tentang perorangan, grup, atau organisasi.

2.6. *Web Service*

Menurut World Wide Web Consortium (W3C) (2004), *web service* adalah sebuah sistem perangkat lunak yang diidentifikasi dengan sebuah URI, yang antarmuka publik dan pengikatnya ditentukan dan digambarkan menggunakan XML. *web service* tersebut dapat ditemukan dan diidentifikasi dengan sistem perangkat lunak lain. Sistem dapat berinteraksi dengan *web service* sesuai dengan ketentuan yang sudah ditentukan, menggunakan eXtensible Markup Language (XML) sebagai basis pesan dan disampaikan dengan protokol internet.

Menurut Snell, dkk (2001: 6), *web service* merupakan antarmuka yang berada diantara kode aplikasi dan pengguna kode tersebut, bekerja sebagai lapisan abstraksi, memisahkan antara *platform* dan spesifikasi detil bahasa pemrograman tentang bagaimana kode dari aplikasi tersebut sebenarnya dipanggil. Jika sebuah aplikasi dapat diakses dengan jaringan menggunakan kombinasi protokol seperti Hypertext Transfer Protocol (HTTP), XML, Simple Mail Transfer Protocol (SMTP), atau Jabber, aplikasi tersebut dapat disebut sebagai *web service*. Cara kerja dari *web service* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Cara Kerja Web Service

2.7. *Database*

Database atau basis data merupakan sebuah struktur yang berisi informasi tentang berbagai kategori, serta relasi antara berbagai kategori tersebut (Pratt dan Adamski, 2011: 4). Sebuah *database* terdiri dari beberapa ketentuan yang menjadi bagian inti, seperti entitas, atribut, dan relasi. Entitas merupakan sebuah objek yang akan disimpan atau diproses datanya. Atribut adalah karakteristik atau properti dari entitas tersebut. Sedangkan relasi merupakan asosiasi dari setiap entitas (Pratt dan Adamski, 2011: 4-5).

2.8. *Service-Oriented Architecture* (SOA)

Menurut Erl (2013: 37), *Service-Oriented Architecture* (SOA) adalah teknologi model arsitektural terhadap solusi berbasis layanan yang

memiliki karakteristik yang berbeda untuk membantu dalam mewujudkan basis layanan dan tujuan strategi layanan terkait. Sebagai bentuk dari teknologi arsitektur, implementasi SOA dapat berupa kombinasi dari beberapa teknologi, produk, API, ektensi infrastruktur yang membantu, dan berbagai bagian lainnya.

SOA adalah pendekatan dalam membangun sistem IT dengan memanfaatkan aset yang ada dan mempermudah setiap perubahan kebutuhan mendatang yang tak dapat dihindari untuk membantu dalam kegiatan berbisnis. SOA membantu bisnis dalam membuat keputusan bisnis yang didukung dengan teknologi, bukan ditentukan atau dibatasi dengan teknologi (Hurwitz, *et al*. 2007: 7-8).

2.9. *Framework*

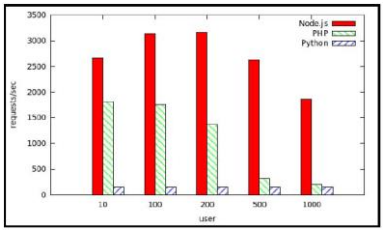
Menurut Riehle (2000:1) dalam dissertasinya, framework adalah bagian terpenting untuk mengembangkan sistem perangkat lunak yang berbasis objek dan berskala besar. Framework membantu dalam hal produktifitas yang tinggi dan waktu yang relative cepat terhadap pemasaran melalui desain dan penggunaan kembali kode program.

Framework adalah suatu file atau software yang di dalamnya terdapat suatu rangkaian sistem untuk mempermudah pengkodean dalam membuat sebuah aplikasi. Dengan framework, sebuah aplikasi dapat tersusun rapi dan pengembangannya pun akan lebih mudah.

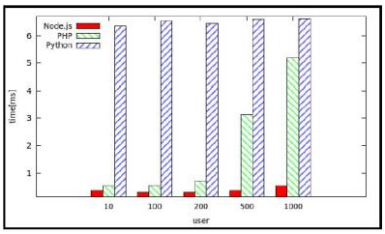
2.10. Node.js

Dilansir dari situs resmi *nodejs.org*, Node.js atau Node merupakan sebuah aplikasi *open-source* sistem yang bergerak pada ruang lingkup asinkron Node mengadaptasi JavaScript sebagai bahasa pemrograman utamanya. Node digunakan dalam berbagai bentuk pengembangan *web*, dari pengembangan aplikasi khusus berbasis jaringan hingga pengembangan aplikasi berbasis *server*.

Dalam segi performa, Node berjalan lebih baik pada situasi seperti *concurrent request* yang tinggi. Menurut Kai Lei, Yining Ma, dan Zhi Tan (2014), Node menangani rata-rata 1500 – 2000 permintaan per detik terhadap 1000 pengguna dalam mengoperasikan perintah "*Select…*" pada *database*. Dibandingkan dengan PHP dan Python yang hanya menangani rata-rata 0 – 500 permintaan per detik terhadap 1000 pengguna secara bersamaan. Node mengeksekusi operasi "*Select…*" dalam waktu yang relatif cepat pada rata-rata 0,1 – 1,0 milidetik per permintaannya.



Gambar 2.2 Grafik *request/sec* Node.js, PHP, dan Python



Gambar 2.3 Perbandingan *execution/ms* antara Node.js, PHP, dan Python

2.11. Express.js

Express.js atau Express merupakan *framework* dari Node yang minimalis dan fleksibel, serta menyediakan fitur yang kuat untuk aplikasi *web* dan *mobile*. Express menyediakan banyak fitur aplikasi *web* yang fundamental tanpa harus menghalangi fitur dari Node (*www.expressjs.com*). Express menjadi salah satu *framework* yang populer JavaScript yang ada di kalangan Node. Contoh program dari Express dapat dilihat pada Gambar 2.4.

```
var express = require('express');
var app = express();

app.get('/', function(req, res) {
  res.send('Hello World!');
});

app.listen(3000, function() {
  console.log('Example app listening on port 3000!');
});
```

Gambar 2.4 Contoh Program pada Express

Express menyediakan metode yang digunakan untuk setiap kata kerja dalam HTTP *request* (GET, POST, SET, dll.), pola URL (Route), penentuan *template engine* (view) yang akan digunakan, dan metode yang akan digunakan untuk mencetak sebuah *response*. Express juga memiliki sebuah *middleware*, yaitu sebuah perangkat lunak penengah, untuk menambahkan fitur lain seperti *cookies*, *sessions*, dll.

2.12. JavaScript Object Notation (JSON)

JavaScript Object Notation atau JSON merupakan sebuah format pertukaran data yang ringan, mudah dibaca oleh manusia, mudah bagi mesin (komputer) dalam menerjemahkan dan membentuknya, merupakan standar yang didasari dari pemrogramman bahasa JavaScript (*www.json.org*). JSON merupakan format teks yang independen dan familiar dengan bahasa C, C++, C#,

Java, JavaScript, Perl, Python, dll. Hal ini membuktikan bahwa JSON merupakan format bahasa pertukaran data yang ideal. JSON dapat menggantikan XML sebagai hasil yang diberikan dari sebuah *web service*.

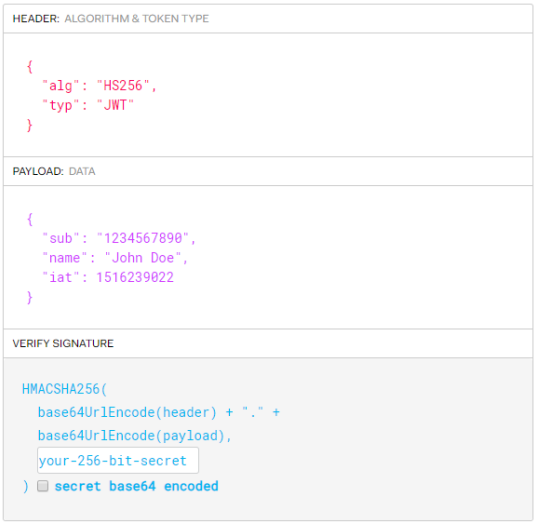
Sebuah *web* dapat memiliki otentikasi, yaitu sebuah skema yang digunakan dalam menentukan akses dan hak dalam menggunakan sumber daya yang ada dalam *web* tersebut. Otentikasi dari *web* dapat berupa sebuah *session*, atau dalam bentuk sebuah *token*. Penggunaan *token* terutama JSON *web Token* (JWT) dapat digunakan dan dijadikan solusi otentikasi dalam pembuatan API dari sebuah *web service*. Karena sifat dari JSON yang universal, JWT menjadi format otentikasi independen yang dapat digunakan di dalam bahasa Java, PHP, Python, dan JavaScript. Sebuah JWT memiliki kombinasi yang tersusun seperti "xxxxx.yyyyy.zzzzz", setiap bagian dipisahkan dengan sebuah simbol "." (titik) dan dikodekan atau *encode* menjadi sebuah Base64Url (*jwt.io*).

Berikut ini contoh dari JWT yang dihasilkan dan JWT yang sudah diterjemahkan oleh situs *jwt.io* pada Gambar 2.5 dan Gambar 2.6.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
zdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJ
SMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Sumber: *jwt.io*

Gambar 2.5 Contoh JWT yang dihasilkan



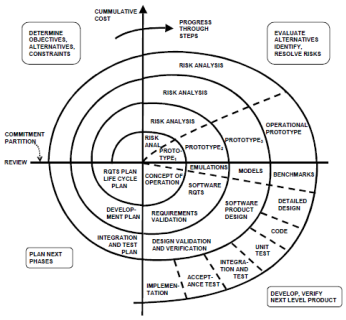
Sumber: *jwt.io*

Gambar 2.6 Hasil Token yang sudah diterjemahkan

2.13. Pengembangan dengan Model Spiral

Model Spiral menurut Barry Boehm (2000) yaitu sebuah famili dari proses pengembangan perangkat lunak yang berkarakteristik perulangan terhadap beberapa set elemen proses pengembangan, serta pengelolaan risiko yang secara aktif berkurang secara berkala.

Model Spiral merupakan model proses yang berbasis terhadap risiko. Setiap pola risiko yang berbeda dapat menyebabkan pemilihan dari inkrementasi, model *waterfall*, *evolutionary prototyping*, atau subset lain dari elemen proses dalam diagram model spiral tersebut. Hasil dari perencanaan dan analisis risiko, setiap proyeknya dapat memilih proses yang berbeda.



Gambar 2.7 Diagram Model Spiral Barry Boehm

3. Metodologi

3.1. Alat dan Bahan Penelitian

Alat Penelitian yang digunakan merupakan satu buah perangkat komputer dan perangkat lunak untuk pengembangannya, keterangan detil tentang alat penelitian tersebut dapat dilihat pada Tabel 3.1.

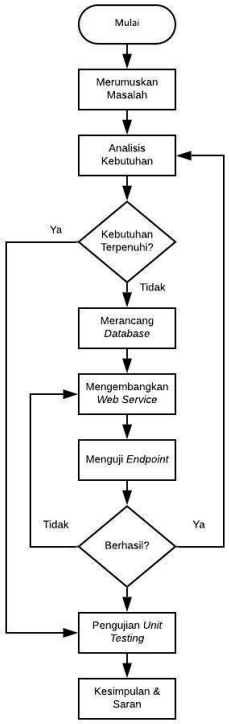
Tabel 3.1 Alat Penelitian

Perangkat Keras	Perangkat Lunak
Intel Core i7-4790U CPU 3.60GHz 8MB L3 Cache *Processor*	Windows 10 64-bit *Operating System*
RAM DDR3 4x4GB	Microsoft Visual Studio Code *Text Editor*
1TB *Hard Drive*	Node.js HTTP *Web Server*
	Postman API *Development*

Bahan yang digunakan untuk penelitian adalah data pembayaran *database* H2H dari UPT TIK UNJ dan dokumen SRS *multibank*.

3.2. Diagram Alir Penelitian

Secara garis besar, penelitian akan dilaksanakan berdasarkan diagram alir pada Gambar 3.1.



Gambar 3.1 Alur Penelitian

3.3 Pengujian

Pengujian akan menggunakan metode *Unit Testing*, yaitu pengujian yang dilakukan terhadap seluruh *endpoint* yang telah dikembangkan pada *web service*. *Endpoint* yang dihasilkan memiliki sebuah *prefix* yaitu "/tb-bill-detail" dan "/user" yang terletak pada bagian depan dari mekanisme *routing* yang telah dibuat sehingga URL yang akan dihasilkan adalah "http://[*HOSTNAME*]:[*PORT*]/tb-bill-detail/[*ENDP-OINT*]" dan "http://[*HOSTNAME*]:[*PORT*]/user/[*E-NDPOINT*]". Daftar dari *endpoint* yang telah dikembangkan dapat dilihat pada Tabel 3.2.

Tabel 3.2 Daftar Endpoint *Multibank* pada modul bank

Requirement	***Prefix***	***Endpoint***	**Tipe *Request***
Menampilkan tagihan berdasarkan nomor tagihan	/tb-bill-detail	/bill/[bill]	GET
Menampilkan satu tagihan atau lebih berdasarkan nomor induk mahasiswa		/nim/[nim]	GET
Melakukan pembayaran berdasarkan nomor tagihan		/update/[bill]	PUT
Melakukan pembayaran untuk satu nomor tagihan atau lebih		/bulk-transaction	PUT
Melakukan *reversal* terhadap satu nomor tagihan		/reverse/[bill]	GET
Melakukan *reversal* berdasarkan nomor pembayaran		/reverse/payment/[payment_number]	GET
Melakukan *reversal* untuk satu nomor tagihan atau lebih		/bulk-reverse	PUT
Melakukan *Login* untuk menghasilkan *token*	/user	/x-login	POST

Pengujian fungsional dilakukan dengan skema yang telah ditentukan dan dicontohkan pada Tabel 3.3 – Tabel 3.4.

Tabel 3.3 Uji Fungsional Modul 2.a Menampilkan Tagihan Berdasarkan Nomor

Prefix	*Endpoint*	Skenario 1	Skenario 2	*Response* (JSON)
/tb-bill-detail	/bill/[bill]	Success, Status Code: 200	-	Menampilkan data tagihan dan mahasiswa terkait
		Error, Status Code: 205	Code: 101	msg: Nomor tagihan tidak ditemukan
		Error, Status Code: 500	-	*Internal Server Error*

Tabel 3.4 Uji Fungsional Modul 2.b Menampilkan Satu atau Lebih Tagihan

No	***Prefix***	***Endpoint***	**Skenario 1**	**Skenario 2**	***Response* (JSON)**
1.	/tb-bill-detail	/nim/[nim]	Success, Status Code: 200	-	Menampilkan seluruh data tagihan yang sedang aktif berdasarkan nomor induk mahasiswa
			Error, Status Code: 205	Code: 101	msg: Tidak ada tagihan pembayaran yang aktif / semua tagihan sudah dibayarkan
				Code: 104	msg: Gagal mengambil data, NIM tidak ditemukan
				Code: 105	msg: Input NIM Salah! (10 karakter)
				-	Bad Request
			Error, Status Code: 500	-	*Internal Server Error*

4. Hasil dan Analisis

4.1. Hasil Penelitian

Web Service yang dikembangkan menghasilkan 8 unit *endpoint*. Setiap *endpoint* dikembangkan dengan proses yaitu: 1) analisis kebutuhan; 2) penyesuaian dengan *database*; 3) penentuan

algoritma; 4) penulisan kode program; dan 4) pengujian *endpoint*. Pengembangan menggunakan model spiral dengan membentuk set-set artefak yang berjalan secara berurutan pada saat membentuk satu *endpoint* dengan *endpoint* lainnya. Setiap *endpoint* dikembangkan hingga sesuai dengan target pengujian dan *requirement*, lalu dilanjutkan dengan mengembangkan *endpoint* berikutnya.

4.2. Analisis Data Penelitian

Pengujian dengan metode *Unit Testing* dilakukan dengan menguji *endpoint* yang dibuat, lalu dilakukan analisis terhadap keluaran yang diberikan oleh *endpoint* tersebut. Pengujian dilakukan dengan melakukan *request* terhadap *endpoint* dengan http *method* seperti GET, POST, dan PUT. Contoh Hasil pengujian dari setiap *endpoint* tersebut dapat dilihat pada Tabel 4.1 – Tabel 4.2. Skema tabel pengujian yaitu berupa status bekerja, skenario, URL, dan *output*. Status bekerja menunjukan keberhasilan program dari *endpoint* tersebut, skenario merupakan bentuk uji coba yang akan dilakukan, URL merupakan alamat dari *endpoint*, dan *output* menampilkan hasil respon yang akan diberikan oleh *endpoint*.

Tabel 4.1 Pengujian Uji Fungsionalitas *Endpoint* Modul 2.a dengan *Method* GET Menampilkan Detil dari Satu Nomor Tagihan

Status Bekerja (Ya/Tidak)	Skenario (1 & 2)	URL Skenario	*Output*
Ya	Status: 200	/tb-bill-detail/bill/ 5235117081110004	{ "status": 200, "code": 100, "bill_xxxx": "5235161088110004", ….. }
Ya	Status: 205; Code: 101	/tb-bill-detail/bill/ 5235139999110006	{ "status": 205, "code": 101, "msg": "Nomor tagihan tidak ditemukan atau tidak aktif" }
Ya	Status: 205; Code: 102	/tb-bill-detail/bill/ 5235161088110004	{ "status": 200, "code": 102, "bill_xxxx": "5235161088110004", ….. }
Tidak	Status: 500.	/tb-bill-detail/bill/ 5235117081110005	*Internal Server Error*

Tabel 4.2 Pengujian Uji Fungsionalitas *Endpoint* Modul 2.b dengan *Method* GET Menampilkan beberapa Detil Tagihan dari Nim

Status Bekerja (Ya/Tidak)	Skenario (1 & 2)	URL Skenario	*Output*
Ya	Status: 200; Code: 100	/tb-bill-detail/nim/ 5235117081	{ "status": 200, "code": 100, "bill_xxxx": "5235117081110004", "amount": 100, ….. }
Ya	Status: 205; Code: 101	/tb-bill-detail/nim/ 5235117082	{ "status": 205, "code": 101, "msg": "Tidak ada tagihan pembayaran yang aktif / semua tagihan sudah dibayarkan" }
Ya	Status: 205; Code: 104	/tb-bill-detail/nim/ 5235117083	{ "status": 205, "code": 101, "msg": "Gagal mengambil data, NIM tidak ditemukan" }
Ya	Status: 205; Code: 105	/tb-bill-detail/nim/ 52351170833	{ "status": 205, "code": 105, "msg": "Input NIM Salah! (10 karakter)" }
Tidak	Status: 500.	/tb-bill-detail/nim/ 5235117081	Internal Server Error

5. Kesimpulan dan Saran

Pengembangan *web service multibank* menghasilkan 8 unit *endpoint* (Terlampir hal: 58 – 74) yang bekerja pada modul bank atau modul layanan transaksi pembayaran. Pengembangan 8 unit *endpoint* tersebut menggunakan bahasa pemrogramman JavaScript dan ruang lingkup aplikasi *backend* Node.js. Pengembangan *endpoint-endpoint* tersebut mengacu kepada kebutuhan fungsional yang ada pada *System Requirement Specification* yang disediakan oleh pihak pengembang aplikasi. Setelah dikembangkan, 8 unit *endpoint* tersebut dilakukan uji coba dengan metode *Unit Testing* dalam menentukan kelayakan dari setiap *endpoint* yang ada pada *web service*.

Hasil pengujian fungsional dengan metode *Unit Testing* menunjukan bahwa setiap *endpoint* yang dikembangkan berfungsi dengan baik sesuai dengan skenario yang telah dipetakan pada Tabel 4.1 – Tabel 4.8, serta layak digunakan sebagai *web service multibank*.

Untuk pengembangan dan penelitian *web service* selanjutnya, disarankan untuk menggunakan teknologi *web service* yang lebih mutakhir. Teknologi tersebut meliputi bahasa pemrogramman dan *framework* atau *library* yang akan digunakan. Selain teknologi, penelitian terhadap rancangan dan struktur *database* yang lebih baik juga dapat dilakukan. Pengembangan tersebut bergantung dengan kebutuhan dan perubahan dari *requirement* yang akan datang.

Daftar Pustaka:

Admin. 2003. *Introducing JSON*. Diambil dari https://www.json.org/. Diakses pada tanggal 18 April 2019 Pukul 12:45 WIB.

Admin. 2015. *MVC: Model, View, Controller*. Diambil dari https://www.codecademy.com/articles/mvc. Diakses pada tanggal 18 April 2019 Pukul 12:29 WIB.

Admin. 2015. *Node.js: Introduction*. Diambil dari https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm. Diakses pada tanggal 18 April 2019 Pukul 12:33 WIB.

Austin, D., Abbie B., Christopher F., Sharad G. 2004. *Web Services Architecture Requirements (W3C Working Group Note, 11 February 2004)*. Diambil dari https://www.w3.org/TR/wsa-reqs/#id2604831. Diakses pada tanggal 6 April 2019 Pukul 23:21 WIB.

Auth0. 2015. *Introduction to JSON Web Tokens*. Diambil dari https://jwt.io/introduction/. Diakses pada tanggal 18 April 2019 Pukul 12:48 WIB.

American Heritage. 2011. Dictionary of the English Language, Fifth Edition. *Website*. Diambil dari https://www.thefreedictionary.com/website. Diakses pada tanggal 6 April 2019 Pukul 21:55 WIB.

Berners-Lee, T., dkk. 1989. *The World Wide Web*. Diambil dari http://www.emeraldgrouppublishing.com/products/backfiles/pdf/backfiles_sample_5.pdf. Diakses pada tanggal 6 April 2019 Pukul 23:11 WIB.

Berners-Lee, T., Thomas Roy Fielding, dan L. Masinter. 2005. *Uniform Resource Identifier (URI): Generic Syntax*. Diambil dari https://tools.ietf.org/html/rfc3986#section-3. Diakses pada tanggal 12 Januari 2019 Pukul 23:35 WIB.

Boehm, B. 2000, *Spiral Development: Experience, Principles, and Refinements*. Editor oleh Hansen, J. W., Pittsburgh: Software Engineering Institute, Carnegie Mellon University.

Brookshear, J. G. dan Dennis Brylow. 2015. *Computer Science: An Overview Global Ed.* Ed-12. Harlow: Pearson Education Limited.

Erl, T., dkk. 2013. *SOA with REST : principles, patterns & constraints for building enterprise solutions with REST*. Munich: Prentice Hall.

Etymonline. *Bank (n.1)*. Diambil dari https://www.etymonline.com/word/bank. Diakses pada tanggal 12 Januari 2019 Pukul 17:31 WIB.

Fielding, R. T. dan J. Reschke. 2014a. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. Diambil dari https://tools.ietf.org/html/rfc7230#appendix-A.1. Diakses pada tanggal 12 Januari 2019 Pukul 21:46 WIB.

Fielding, R. T. dan J. Reschke. 2014b. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. Diambil dari https://tools.ietf.org/html/rfc7231#section-4. Diakses pada tanggal 12 Januari 2019 Pukul 23:01 WIB.

Fielding, R. T. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. Disertasi. Irvine: University of California.

HarperCollins Publishers. 2014. Collins English Dictionary – Complete and Unabridged, *12th Edition*. S.v. "multibank.". Diambil dari https://www.thefreedictionary.com/multibank. Diakses pada tanggal 12 Mei 2019 Pukul 15:10 WIB.

Hurwitz, J., Robin B., Carol B., dan Marcia K. 2007. *Service Oriented Architecture For Dummies*. Indianapolis: Willey Publishing.

Kadir, A. 2003. Pengenalan Sistem Informasi. Yogyakarta: Andi.

McKinney, R., dkk. 2019. *Express: Fast, unopinionated, minimalist web framework for Node.js*. Diambil dari http://expressjs.com. Diakses pada tanggal 10 April 2019 Pukul 12:30 WIB.

Muktar, B. 2016. Bank dan Lembaga Keuangan lain. Jakarta: Kencana.

Node.js Foundation. 2018. *About Node.js*. Diambil dari https://nodejs.org/en/about/#about-node-js. Diakses pada tanggal 10 April 2019 Pukul 12:10 WIB.

Ono. 2015. 12 Kelebihan dan Kekurangan MySQL Server. Diambil dari https://dosenit.com/software/dbms/mysql/kelebihan-dan-kekurangan-mysql-server. Diakses pada tanggal 6 April 2019 Pukul 23.51 WIB.

Oxford University Press. 2019. *Bank*. Diambil dari https://en.oxforddictionaries.com/definition/bank#h70034908274000. Diakses pada tanggal 12 Mei 2019 Pukul 20:00 WIB.

Oxford University Press . 2019. *Multi-*. Diambil dari https://en.oxforddictionaries.com/definition/multi-. Diakses pada tanggal 12 Mei 2019 Pukul 21:00 WIB.

Pratt, P. J., dan J. J. Adamski. 2012. *Concepts of Database Management*. Ed-7. Boston: Course Technology.

Pustikom UNJ. 2017a. *Frequently Asked Question*. Diambil dari http://penmaba.unj.ac.id/jalur/pascasarjana. Diakses pada tanggal 18 Juli 2019 Pukul 13:16 WIB.

Pustikom UNJ. 2017b. Pascasarjana: Penerimaan Mahasiswa Baru Jenjang Magister(S-2) dan Doktor(S-3) Universitas Negeri Jakarta. Diambil dari http://penmaba.unj.ac.id/jalur/pascasarjana.

Diakses pada tanggal 18 Juli 2019 Pukul 13:22 WIB.

Riehle, D. 2000. *Framework Design: A Role Modeling Approach*. *Tesis*. Swiss Federal Institute Of Technology Zurich.

Snell, J., Doug T., dan Pavel K. 2002. *Programming Web Services with SOAP: Building Distributed Applications*. USA: O'Reilly.

Strassner, T. 2015. *XML vs JSON*. Diambil dari https://www.cs.tufts.edu/comp/150IDS/final_papers/tstras01.1/FinalReport/FinalReport.html. Diakses pada tanggal 18 Juli 2019 Pukul 9:54 WIB.

The Open Group. 2016. *Service-Oriented Architecture – SOA Features and Benefits*. Diambil dari https://www.opengroup.org/soa/source-book/soa/p4.htm. Diakses pada tanggal 18 Juli 2019 Pukul 9:45 WIB.

Wahyono, D. 2018. Mengenal Apa itu Framework. Diambil dari https://www.plimbi.com/article/169037/mengenal-apa-itu-framework-. Diakses pada tanggal 10 April 2019 Pukul 11:40 WIB.