Implementasi Web Service pada Aplikasi Mobile untuk Mendukung Sistem Informasi di Bandung N-Max Community

R. Sandhika Galih A.[1)], Faerul Salamun[2)]
[1,2]Program Studi Teknik Informatika, Universitas Pasundan Bandung
[1,2]Jl. Dr. Setiabudhi 193 Bandung, 40153
e-mail: sandhikagalih@unpas.ac.id[1)], faerulsalamun@gmail.com[2)]

Abstrak

Web service merupakan sebuah perangkat lunak yang digunakan sebagai jembatan yang memungkinkan berbagai sistem dapat berkomunikasi tanpa terpengaruh dengan perbedaan platform. Maka web service akan menyediakan layanan–layanan yang ada pada sistem lama untuk bisa berkomunikasi dengan sistem yang baru tanpa melakukan perubahan pada keduanya. Pada paper ini akan dijelaskan mengenai pembuatan web service yang dapat mengintegrasikan aplikasi web dan aplikasi mobile yang ada di Sistem Informasi Bandung N-Max Community. Menggunakan teknologi RESTful pada pertukaran data dengan bentuk JSON melalui URI (Uniform Resource Identifier). Web service akan dibuat menggunakan metodologi WSIL atau Web Service Implementation Lifecycle, dimana di dalamnya terdapat tahapan mulai dari analisis, desain, koding, testing dan deployment.

Kata kunci: *Web Service, RESTful, WSIL (Web Service Implementation Lifecycle), Sistem Informasi*

1. Pendahuluan

Sistem komputer terdistribusi saat ini semakin penting dan banyak sekali digunakan mengingat dunia yang semakin didominasi oleh Teknologi Informasi. Hal tersebut salah satunya disebabkan oleh adanya standarisasi dalam arsitektur komputasi terdistribusi yang dinamakan dengan *Web Service* [1]. Di beberapa literatur kita dapat temukan artikel-artikel ilmiah [2], [3], [4] yang menyebutkan permasalahan arsitektur pada komputasi dan aplikasi terdistribusi dapat diselesaikan menggunakan *web service*, terutama di bidang pendidikan, kesehatan, bisnis dan lain-lain.

Bandung N-Max Community adalah komunitas kendaraan bermotor yang cukup aktif di kota Bandung. Didirikan pada 14 Maret 2015 dengan anggota awal sekitar 170 orang. Berdasarkan data yang diterima dari sekretariat Bandung N-Max Community, setiap bulan hampir ada sekitar 50 anggota yang melakukan pendaftaran. Untuk melakukan pendaftaran, anggota baru harus datang ke sekretariat untuk mengisi formulir pendaftaran. Selain itu dalam berbagi informasi mengenai kegiatan-kegiatan komunitas tersebut serta informasi mengenai tips dan trik seputar sepeda motor nmax masih menggunakan surat pemberitahuan ataupun pada saat melakukan perkumpulan. Walaupun komunitas Bandung Nmax Community sudah menggunakan sosial media dan aplikasi komunikasi seperti telegram tetapi masih kurang dapat mengakomodir yang dibutuhkan oleh komunitas tersebut, salah satunya adalah bagaimana cara untuk melihat lokasi anggota yang terdekat jika salah satu anggota mengalami kesulitan dengan kendaraanya. Hal ini dikarenakan belum adanya sebuah aplikasi khusus untuk komunitas tersebut [5].

Untuk membuat sebuah sistem informasi komunitas yang baik, diperlukan arsitektur perancangan yang baik pula, karena aplikasi tidak hanya berada di sisi *client* dalam bentuk *mobile app*, tetapi dibutuhkan juga sebuah *CMS* di sisi *server* untuk mengelola data. Selain itu dibutuhkan juga *web service* agar *client mobile app* dan *CMS* dapat saling bertukar informasi yang berada di dalam database *server*.

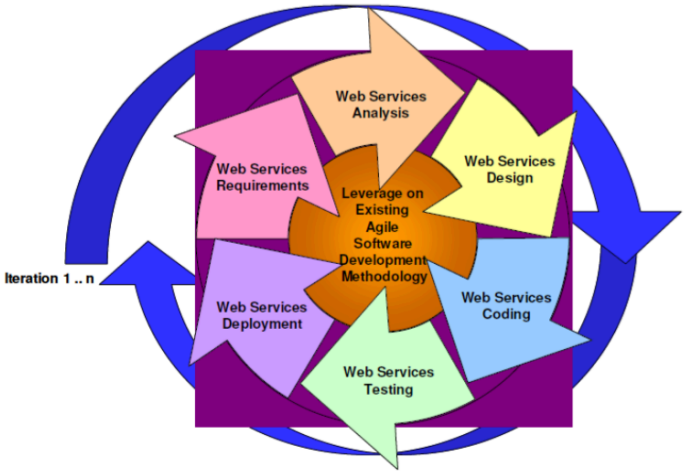
Berdasarkan pemaparan diatas, pada paper ini akan dijelaskan mengenai bagaimana mengimplementasikan *web service* pada sistem informasi Bandung N-Max Community menggunakan metode WSIL (*Web Service Implementation Lifecycle*), agar aplikasi-aplikasi terdistribusi yang ada di dalam sistem informasi tersebut dapat terintegrasi dengan baik.

2. Metode Penelitian

2.1. *Web Service Implementation Methodology*

Web Service Implementation Methodology (WDIM) mendefinisikan pendekatan sistematis untuk pengembangan web service dengan memanfaatkan metodologi pengembangan perangkat lunak *agile* dan memperluas metodologi tersebut dengan menetapkan kegiatan spesifik web service dan peran yang sesuai dan produk kerja yang dihasilkan dalam proses [13]. Metodologi ini akan menentukan seperangkat praktek

umum yang menciptakan metode kerangka independen, yang dapat diterapkan oleh sebagian besar para pengembang perangkat lunak untuk mengembangkan *web service*. Siklus pengembangan web service dapat dilihat pada gambar 1.



Gambar 1. *Web Service Implementation Lifecycle*

Fase *Requirement*

Tujuan dari fase requirement ini adalah untuk memahami kebutuhan bisnis dan menerjemahkannya kepada kebutuhan *web service* dalam hal fitur, fungsional dan kebutuhan non fungsional dan kendala dimana *web service* harus mematuhinya.

Fase *Analysis*

Pada tahap analysis, requirement *web service* lebih disempurnakan dan diterjemahkan ke dalam model konseptual dimana tim pengembangan teknis dapat memahami. Pada fase ini juga dapat dilakukan analisis arsitektur untuk mendefinisikan struktur *high-level* dan mengidentifikasi antarmuka dari *web service.*

Fase *Design*

Desain rinci *web service* dilakukan dalam fase ini. Dalam fase ini, para *designer* harus mendefinisikan kontrak antarmuka *web service* yang telah diidentifikasikan dalam tahap analisis. Pendefinisian antarmuka *web service* harus mengidentifikasikan unsur – unsur dan jenis data yang sesuai dengan jenis interaksi antara *web service* dan klien

Fase *Coding*

Fase *coding* dan *debugging* untuk implementasi *web service* pada dasarnya sangat mirip dengan fase *coding* dan *debugging* pada perangkat lunak berbasis komponen.

Fase *Testing*

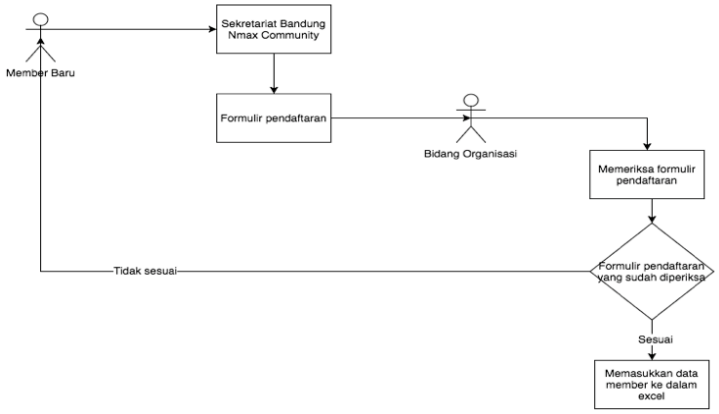
Untuk pengujian *web service*, selain pengujian untuk kebenaran dan kelengkapan fungsional, penguji harus juga melakukan pengujian interoperabilitas antara *platform* yang berbeda dan program klien. Selain itu, pengujian kinerja harus dilakukan untuk memastikan bahwa *web service* mampu menahan beban maksimum sebagaimana ditentukan dalam *requirement* non fungsional.

Fase *Deployment*

Tujuan dari fase *deployment* adalah untuk memastikan bahwa *web service* digunakan dengan benar. Fase ini dilaksanakan setelah dilakukan fase pengujian. *Deployment web service* dilakukan pada *platform* tertentu. Titik akhir dari layanan *web service* adalah menentukan dimana layanan ini digunakan dan menyesuaikan identifikasi dan konfigurasi. Tugas utama *deployer* adalah untuk memastikan bahwa *web service* telah dikonfigurasi dan dikelola dengan benar, dan menjalankan tes pasca *deployment* untuk memastikan bahwa *web service* memang siap untuk digunakan.

2.3. Analisis Sistem Informasi Bandung N-Max *Community*

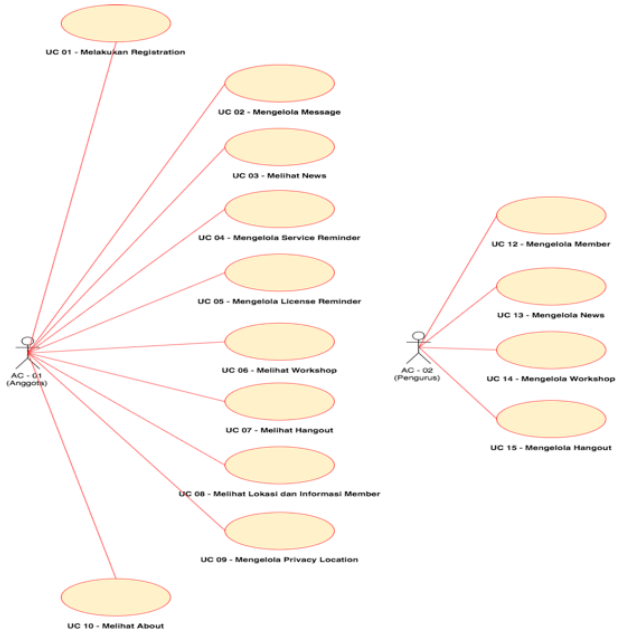
Analisis kebutuhan bertujuan untuk menentukan komponen-komponen yang dibutuhkan dalam proses pengembangan/pembangunan aplikasi. Proses ini merupakan langkah awal untuk menentukan aplikasi yang akan dihasilkan. Proses penganalisisan kebutuhan aplikasi pada penelitian ini dilakukan dengan melakukan wawancara dengan salah satu member komunitas Bandung N-Max Community. Gambar 2 menunjukkan mengenai mekanisme saat ini untuk melakukan pendaftaran member baru di Bandung N-Max Community



Gambar 2. Alur pendaftaran member baru di Bandung N-Max Community

Diagram *Use-Case*

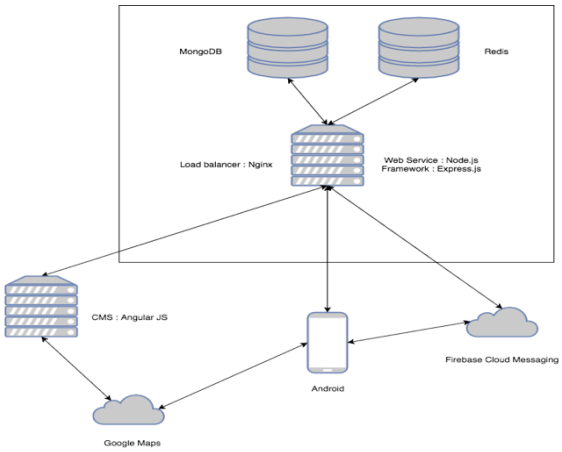
Use Case adalah teknik untuk menangkap kebutuhan fungsional dari sistem. *Use Case* bekerja dengan menggambarkan interaksi yang khas antara pengguna dari sistem dan sistem itu sendiri, memberikan narasi tentang bagaimana sistem yang digunakan. Gambar 3 berikut ini menggambarkan interaksi pada sistem yang akan dibangun.



Gambar 3. Diagram *Use-Case* Aplikasi Bandung N-Max Community

Arsitektur Teknologi

Gambar 4 berikut ini menjelaskan mengenai gambaran umum dari sistem yang akan dibangun untuk Bandung N-Max Community. Bagian yang ada di dalam kotak merupakan *web service* yang akan dibuat.



Gambar 4. Arsitektur Teknologi Aplikasi Bandung N-Max Community

3. Hasil dan Pembahasan

3.1. Fase *Analysis*

Pada fase analisis ini akan menentukan mengenai arsitektur dari *web service* yang akan dibangun, memilih platform yang akan digunakan, menentukan kandidat *web service*, menentukan detil *web service*, mengidentifikasi *web service* yang dapat digunakan kembali dan mengidentifikasi *web service interface*.

Kandidat *Web Service*

Kandidat web service merupakan objek yang akan dijadikan web service. Kandidat web service yang akan dibangun ditentukan berdasarkan kebutuhan fungsional web service yang telah ditentukan sebelumnya pada tahap analisis. Untuk lebih rincinya kandidat web service bisa dilihat pada Tabel 2 berikut ini.

Tabel 1. Kandidat Web Service

No	Use Case	Service	Deskripsi
1	UC - 01	Penambahan data anggota	Digunakan untuk menambahkan data user
2	UC – 02	Mengelola data *conversation*	Digunakan untuk mengelola data *conversation message*
		Mengelola data *message*	Digunakan untuk mengelola data *message*
3	UC – 03	Mengelola data *news*	Digunakan untuk mengelola data *news*
4	UC – 04	Mengelola data *service reminder*	Digunakan untuk mengelola data *service reminder*
5	UC – 05	Mengelola data *license reminder*	Digunakan untuk mengelola data *license reminder*
6	UC – 07	Mengelola data *hangout*	Digunakan untuk mengelola data *hangout*
7	UC – 08	Mengelola data lokasi	Digunakan untuk mengelola informasi lokasi
8	UC – 11	Mengelola data *member*	Digunakan untuk mengelola data *member*
9	UC – 13	Mengelola data *workshop*	Digunakan untuk mengelola data *workshop*
10	UC - 14	Mengelola data *hangout*	Digunakan untuk mengelola data *hangout*

Identifikasi *Web Service Interface*

Web service interface berfungsi sebagai antarmuka dari *web service* yang akan berinteraksi dengan aplikasi klien. Teknologi *web service* yang digunakan adalah *RESTful web service*, dan karena itu maka akan digunakan *HTTP Method* sebagai *interface*. *HTTP Method* yang digunakan ditentukan berdasarkan operasi yang dilakukan oleh *service* tersebut. Tabel 3 berikut ini akan menjelaskan tentang identifikasi *web service interface* pada aplikasi yang akan dibangun.

Tabel 2. Identifikasi *Web Service Interface*

No	Use Case	Service	HTTP Method
1	UC - 01	Penambahan data anggota	POST
2	UC – 02	Mengelola data *conversation*	GET, POST, DELETE
		Mengelola data *message*	GET, POST, DELETE
3	UC – 03	Mengelola data *news*	GET, POST, DELETE, PUT
4	UC – 04	Mengelola data *service reminder*	GET, PATCH
5	UC – 05	Mengelola data *license reminder*	GET, PATCH
6	UC – 07	Mengelola data *hangout*	GET, POST
7	UC – 08	Mengelola data lokasi	GET, PATCH
8	UC – 11	Mengelola data *member*	GET, PUT, DELETE

No	Use Case	Service	HTTP Method
9	UC – 13	Mengelola data *workshop*	GET, POST, DELETE, PUT
10	UC - 14	Mengelola data *hangout*	GET, POST, DELETE, PUT

3.2. Fase *Design*

Web service design dilakukan dengan cara mendefinisikan kontrak web service interface yang sudah diidentifikasi pada tahap analisis, seperti menentukan tipe data.

Desain *URI*

Desain *URI* dimaksudkan supaya setiap *resource* dari *service* memiliki *URI* yang unik, sehingga klien dapat mengidentifikasi setiap *resource* berdasarkan *URI*nya. Dalam proses desain *URI* didasarkan pada *database sistem dashboard* yang sudah ada, sehingga untuk setiap *service* yang dibangun, penamaanya diambil berdasarkan nama tabel dari *database* sistem *dashboard*. Kemudian untuk parameter yang harus dikirimkan pada saat mengirimkan permintaan terhadap *service*, diambil dari kolom yang ada pada masing – masing tabel. Tabel 4 berikut ini menjelaskan mengenai desain URI *web service*.

Tabel 3. Desain *URI (Uniform Resource Identifier)*

No	Use Case	Service	Desain *URI*
1	UC - 01	Penambahan data anggota	api/v1/signin
2	UC – 02	Mengelola data *conversation*	api/v1/conversations
		Mengelola data *message*	api/v1/ conversation s/{id_conversation}
3	UC – 03	Mengelola data *news*	api/v1/news
4	UC – 04	Mengelola data *service reminder*	api/v1/accout/service
5	UC – 05	Mengelola data *license reminder*	api/v1/account/profile
6	UC – 07	Mengelola data *hangout*	api/v1/hangouts
7	UC – 08	Mengelola data lokasi	api/v1/account/location
8	UC – 11	Mengelola data *member*	api/v1/users
9	UC – 13	Mengelola data *workshop*	api/v1/workshops
10	UC - 14	Mengelola data *hangout*	api/v1/hangouts

3.3. Fase *Coding*

Tahap pengkodean *web service* yang akan dibangun mengunakan bahasa pemrograman Node.js dengan framework Express.js serta database MongoDB dan redis.

Inisialisasi *Web Service*

Dibutuhkan sebuah file inti yang berguna untuk mengatur alur kerja dari *web service.* Pada app.js yang merupakan file inti tersebut dilakukan inisialisasi kebutuhan-kebutuhan utama dari *web service* seperti menkonfigurasi *framework* express.js, melakukan koneksi ke *database*, autentifikasi, dan routing *endpoint* yang dapat dilihat pada gambar 5.

app.js

```
'use strict';

// node modules
const debug = require(`debug`)(`app`);

const express = require(`express`);
const swig = require(`swig`);
const logger = require(`morgan`);
const cookieParser = require(`cookie-parser`);
const bodyParser = require(`body-parser`);
const session = require(`express-session`);
const passport = require(`passport`);
const flash = require(`connect-flash`);
const i18n = require(`i18n`);
const RedisStore = require(`connect-redis`)(session);
const cors = require(`cors`);
const config = require(`./config/${process.env.NODE_ENV || ``}`);

// Passport Config
require(`./app/helpers/passport`);

// Reload database
require(`./app/helpers/database`);

// Cronjob
require(`./app/helpers/cronjob`);

// i18n
i18n.configure({
  locales: [`en_US`, `id`],
  directory: `${__dirname}/app/locales`,
  cookie: config.cookie.secret,
  defaultLocale: `en_US`,
});

// express setup
const app = express();

// view engine setup
swig.setDefaults(config.swig);
app.engine(`swig`, swig.renderFile);
app.set(`views`, `${__dirname}/app/views`);
app.set(`view engine`, `swig`);
app.use(cors());
app.disable(`x-powered-by`);
```

Gambar 5. Kode file app.js sebagai file inti *web service*

Controllers* pada *Web Service

Controllers pada *web service* berguna untuk melakukan pemrosesan data yang diminta oleh pengguna *web service* untuk nantinya dihubungkan ke database serta menampilkan data kepada pengguna dalam bentuk *json.* Daftar *controller* pada *web service* Bandung N-Max Community dapat dilihat pada tabel 5 berikut ini.

Tabel 5. *Controller web service*

No	Controller	Method
1	ConversationController	getDataApi, getDetailDataApi, createDataApi, createMessageConversationDataApi, deleteConversationDataApi, deleteMessageConversationDataApi
2	HangoutController	getDatatableApi, getDataApi, getDataDetailDataApi, createDataApi, createDataRatingApi, putDataApi, createDataCommentApi, getDataCommentApi
3	NewsController	getDatatableApi, getDataApi, getDataDetailDataApi, createDataApi, putDataApi, deleteDataApi
4	OAuthController	wrapExchange, token, bearer, getTokenDirectPassword, createClientApi
5	ReminderController	patchLicenseDataApi, patchServiceDataApi,
6	UserController	getDatatableApi, getDataApi, getDataDetailDataApi, createDataApi, putDataApi, deleteDataApi, patchVerificationDataApi, patchTokenDataApi, getUserDataApi, getLocationUserApi, ptchLocationDataApi, patchPrivacyLocationUserDataApi
7	WorkerController	notificationWorkerQueue, notificationUserWorkerQueue
8	WorkshopController	getDatatableApi, getDataApi, getDataDetailDataApi, createDataApi, createDataRatingApi, putDataApi, deleteDataApi, createDataCommentApi, getDataCommentApi

4. Simpulan

Kesimpulan dari penelitian ini adalah bahwa *web service* telah diimplementasikan pada sistem informasi Bandung N-Max Community, dan berdasarkan pengujian telah dapat mengintegrasikan aplikasi web CMS pengelolaan data yang digunakan oleh pengurus komunitas dan aplikasi mobile yang digunakan oleh member komunitas.

Ucapan Terimakasih

Ucapan terimakasih disampaikan kepada Jurusan Teknik Informatika dan Fakultas Teknik Universitas Pasundan Bandung baik dalam bentuk dana, fasilitas dan peralatan yang telah banyak membantu bagi keberhasilan dan kelancaran kegiatan penelitian.

Daftar Pustaka

[1] G.M. Tere, B.T. Jadhav, R.R. Mudholkar. *Dynamic Invocation of Web Services*. Advances in Computational Research. 2012; Vol. 4, Issue 1, pp.-78-82.
[2] F. Felhi, J. Akaichi. *Adaptation of Web Services to the Context Based on Workflow*. International Journal of Web & Semantic Technology (IJWesT). 2012; Vol. 3, No.4.
[3] A. N. Khan, S. Asghar, S. Fong. *Framework of integrated Semantic Web Services and Ontology Development for Telecommunication Industry*. Journal of Emerging Technologies in Web Intelligence. 2011; Vol 3, No 2, pp. 110-119.
[4] C. Strimbei. *Smart Data Web Services*. Informatica Economica Journal. 2012; Vol. 16, No 4.
[5] S. Faerul. *Pembangunan Aplikasi Sistem Informasi Bandung N-Max Community.* Universitas Pasundan Bandung. 2017.
[6] Kreger, Heather. *Web services Conceptual Architecture*. IBM Software Group. 2001.
[7] The Stencil Group. *Defining Web services*. www.perfectXML.com/Xanalysis/TSG/TSG_DefiningWebServices.pdf. diakses 20 Desember 2017
[8] W3C Working Group. *Web services Architecture.* http://www.w3.org/TR/ws-arch/. Diakses 24 Desember 2017
[9] Octavian D., Marian P. *Web Services in Mobile Applications.* Informatica Economica Journal. 2013; Vol 17, No. 2.
[10] J. Sandoval. *RESTful Java Web service*. Packt Publishing Ltd. 2009.
[11] T. Hunter II. *Consumer-Centric API Design*. https://thomashunter.name/consumer-centric-api-design/. Diakses 24 Desember 2017.
[12] JSON. *Intoducing JSON*. www.json.org. Diakses 20 Desember 2017.
[13] OASIS. *Web service Implementation Methodology*. OASIS Public Review Draft 1.0. 2005.
[14] H. Endang. *Web Service untuk E-Voting*. Universitas Pasundan Bandung 2015.