TUGAS AKHIR I

<< Judul Skripsi >>

Diajukan untuk memenuhi salah satu syarat mengerjakan dan menempuh ujian tugas akhir 2

Disusun Oleh:

Nama	: Muhammad Arji' Sharofuddin
NIM	: A11.2014.08073
Program Studi	: Teknik Informatika

FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO
SEMARANG
20...

PERSETUJUAN SKRIPSI

Nama	: Muhammad Arji' Sharofuddin
NIM	: A11.2014.08073
Program Studi	: Teknik Informatika
Fakultas	: Ilmu Komputer
Judul Tugas Akhir	: << Judul Skripsi >>

Tugas Akhir ini telah diperiksa dan disetujui,

Semarang, …….

Menyetujui: Pembimbing	Mengetahui: Dekan Fakultas Ilmu Komputer
Fahri Firdausillah S.Kom, M.CS	**Dr. Drs. Abdul Syukur MM**

PENGESAHAN DEWAN PENGUJI

Nama	: Muhammad Arji' Sharofuddin
NIM	: A11.2014.08073
Program Studi	: Teknik Informatika
Fakultas	: Ilmu Komputer
Judul Tugas Akhir	: << Judul Skripsi >>

Tugas akhir ini telah diujikan dan dipertahankan dihadapan Dewan Penguji pada Sidang tugas akhir tanggal …..... Menurut pandangan kami, tugas akhir ini memadai dari segi kualitas maupun kuantitas untuk tujuan penganugrahan gelar Sarjana Komputer (S.Kom.)

Semarang, …….

Dewan Penguji:

Nama Dosen Penguji 1
Anggota

Nama Dosen Penguji 2
Anggota

Nama Ketua Dosen Penguji
Ketua Penguji

HALAMAN RINGKASAN

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

ARTI LAMBANG, SINGKATAN, DAN ISTILAH

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dewasa ini pengembangan aplikasi web dibangun dengan komposisi dari berbagai komponen terpisah memanfaatkan teknologi web 2.0 untuk mengkombinasikan data dari *web service* yang dikembangkan secara terpisah baik secara internal maupun dari pihak lain [1]–[3]. Komposisi berbagai komponen ini berarti integrasi dengan web API dalam satu aplikasi. Pengembangan aplikasi seperti ini dinilai memiliki hambatan dalam hal heterogenitas semantic dari web API dan berkembangnya API tersebut [4].

Beberapa hal yang perlu dipertimbangkan dalam hal ini antara lain yang pertama yaitu sumber data yang berbeda kemungkinan memiliki cara pengambilan data (*fetching*) atau pemanipulasian data sendiri-sendiri. Maka integrasi ini membutuhkan cara yang terstandar dalam pengambilan data-data. Kemudian sumber data dari service memiliki berbagai model/entitas, yang ada juga yang saling berhubungan. Ini memungkinkan terjadi kesalahan dalam merepresentasikan data. Maka diperlukan mekanisme untuk memetakan model-model dari service yang sesuai dengan sistem.

Teknologi yang sudah ada seperti WADL (Web Application Description Language)[5] yang dikembangkan untuk *RESTful web service* mampu menyajikan representasi terstruktur dan semantic dari API web service. Namun dalam pengembangan aplikasi, API yang sudah mampu ter-representasikan strukturnya ini masih terdapat kekurangan dalam hal fleksibilitas ketika API berkembang semakin kompleks.

Arsitektur *web service* dalam REST berupa *endpoint-endpoint* yang masing-masing darinya mewakili sebuah model sumber data yang dapat diakses dengan HTTP request ke endpoint yang diinginkan. Pendekatan seperti ini mengakibatkan

terjadinya percakapan bolak-balik antara client dengan server untuk meminta data yang di mana data tersebut dari server tersedia dalam endpoint yang berbeda, disebut juga *under fetching [6]*.

Kombinasi dari semantic dan fleksibilitas, pada tahun 2016 muncul spesifikasi baru untuk web service, yaitu GraphQL. GraphQL di-*release* oleh Facebook sebagai cara baru dalam mengakses data pada aplikasi *client-server* dengan menggunakan bahasa *query* yang intuitif dan fleksibel [7]. Spesifikasi ini menjadi dasar untuk framework dan komunitas membangun *tools, library,* dan pengimplementasian dalam pengembangan web API, dimana data yang tersedia dari API disajikan dalam bentuk skema dan memberi kemudahan akses data dengan query yang memiliki sintaks mirip dengan JSON.

Dari penjelasan yang penulis paparkan di atas, maka penulis tertarik untuk melakukan penelitian mengenai “Implementasi GraphQL pada Sistem Pengaduan Masyarakat”.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang penulis sampaikan, maka dapat dirumuskan permasalahannya adalah:

1. Bagaimana implementasi GraphQL untuk membangun aplikasi dengan *backend API* dan *frontend* terpisah.
2. Apakah GraphQL dapat mengatasi *underf etching dan over fetching.*
3. Bagaimana membangun *Web Service* API yang semantic

1.3 Batasan Masalah

1. Metode *Web Service* penulis menggunakan GraphQL dengan hasil data berformat JSON.
2. Aplikasi yang penulis kembangkan merupakan *server-side* (beroperasi di bagian server) sebagai *web service* untuk aplikasi mobile dan web.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah dijelaskan sebelumnya, tujuan dari penelitian ini adalah

1. Membangun aplikasi dengan *backend API* dan *frontend* terpisah menggunakan GraphQL
2. Membangun *web service* dengan memanfaatkan GraphQL untuk mencegah *under fetching dan over fetching.*
3. Membangun *web service* berbasis GraphQL untuk API yang semantic

1.5 Manfaat Penelitian

Adapun manfaat untuk pengembang adalah:

1. Mendapatkan alternatif solusi untuk cara pengambilan data ke server yang lebih fleksibel
2. Mendapatkan alternatif solusi untuk pembuatan *Web Service* API terdokumentasi

BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Studi

Penelitian pertama dari GraphQL adalah “An Initial Analysis of Facebook’s GraphQL Language” oleh Olaf Hartig dan Jorge Pérez. Penelitian ini bertujuan untuk memahami bahasa query dari GraphQL. Adapun GraphQL ini memiliki bentuk format query yang menyerupai JSON (Javascript Object Notation). Dari analisa penelitian ini menyatakan bahwa GraphQL memiliki konsep baru dalam mengakses data [7].

Penelitian “Efficient Data Communication Between a Web Client and a Cloud Environment” oleh Kit Gustavsson dan Erik Stenlund menjelaskan perbedaan arsitektur antara REST dan GraphQL [8]. Berdasarkan hasil penelitian ini, dalam pengembangan harus terlebih dahulu membuat model keputusan untuk mementukan teknik mana yang akan digunakan. Adapun efek ketika pengembang memutuskan untuk menggunakan GraphQL, pengembangan bergantung dengan *external dependencies* GraphQL. Sedangkan untuk REST, pengembang tidak harus bergantung dengan *external dependencies*. Di sisi lain dalam hal performa, dari penelitian ini menunjukkan GraphQL dapat mengurangi beban dari server maupun client.

Penelitian “Experiences on Migrating RESTful Web Services to GraphQL” menyajikan laporan berisi pengalaman Maximilian Vogel dan timnya dalam memigrasikan aplikasi *smart home* dari yang sebelumnya menggunakan REST API menjadi GraphQL. Hasil dari penelitian ini penyatakan GraphQL dalam konseptual memiliki keunggulan dalam struktur sumber dayanya yang berupa *object graph* dengan satu endpoint URI daripada banyak endpoint URI [9].

Penelitian selanjutnya adalah “Toward Automatic Semantic API Descriptions to Support Services Composition” oleh Marco Cremaschi dan Flavio De Paoli. Tujuannya adalah untuk memperkaya format *OpenAPI Specification* yang populer dengan anotasi semantic, dan menambahkan fungsionalitas dari anotasi semantic dan ke editor terkait (swagger editor). Pada pendahuluan penelitian ini menyatakan bahwa kebanyakan pendekatan yang dilakukan dalam mengintegrasikan Web Service mengalami masalah untuk membuat API berkomunikasi satu sama lain karena kurangnya kecocokan semantik antara data input dan output. Yang seperti ini membutuhkan keahlian khusus hanya untuk men-*deliver web service* yang semantic. Penelitian ini memanfaatkan *tools* Swagger Editor yang tersedia dengan *Open API Specification* untuk membuat *Table Interpretation* sehingga *value* dan properti API dapat “dipahami” oleh komputer. Tabel ini berformat relational lengkap dengan deskripsi API dengan tipe data, nama entitas dan relasi [10].

No	Nama	Tahun	Judul	Metode	Hasil
1	Maximilian Vogel	2018	Experiences on Migrating RESTful Web Services to GraphQL	GraphQL	Hasil dari penelitian ini penyatakan GraphQL dalam konseptual memiliki keunggulan dalam struktur sumber dayanya yang berupa *object graph* dengan satu endpoint URI daripada banyak endpoint URI
2	Marco	2017	Toward Automatic	REST	Penelitian ini

	Cremaschi dan Flavio De Paoli		Semantic API Descriptions to Support Services Composition		memanfaatkan *tools* Swagger Editor yang tersedia dengan *Open API Specification* untuk membuat *Table Interpretation* sehingga *value* dan properti API dapat "dipahami" oleh komputer. Tabel ini berformat relational lengkap dengan deskripsi API dengan tipe data, nama entitas dan relasi
3	Olaf Hartig dan Jorge Pérez	2017	An Initial Analysis of Facebook's GraphQL Language	GraphQL	Penelitian ini menghasilkan sebuah teknologi baru untuk mengakses data pada Web API menggunakan kueri berbentuk Graph.
4	Kit Gustavsson dan Erik Stenlund	2016	Efficient Data Communication Between a Web Client and a Cloud Environment	REST dan GraphQL	Pada penelitian ini menghasilkan, jika pengembang lebih memilih menggunakan GraphQL, pengembang harus bergantung pada external dependencies dengan pertimbangan performa yang lebih cepat ketimbang

					REST API. Tetapi jika pengembang lebih memilih menggunakan REST API pengembang bisa tidak bergantung pada external dependencies.

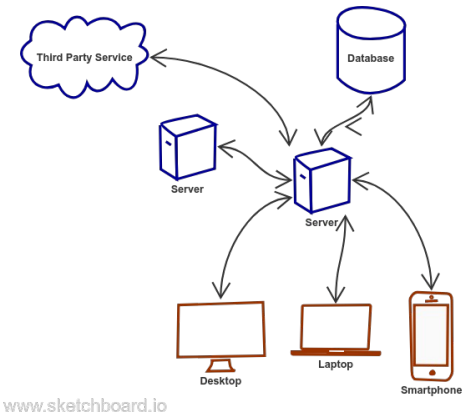
2.2 Tinjauan Pustaka

2.2.1 Aplikasi Modern

Perkembangan teknologi informasi modern ini, pembuatan sebuah aplikasi merupakan sebuah kesatuan sistem yang dibangun dengan berbagai sumber atau aplikasi yang berberda. Hal ini kemudian menjadi tuntutan dimana sebuah sistem harus bisa melakukan integrasi dengan sistem yang lain [11]. Berbagai aplikasi yang ada ini tentunya dibuat dengan teknologi yang beragam, mulai dari bahasa pemrograman hingga platform yang berbeda.

Sebagai contoh aplikasi *market place* tersedia aplikasi *mobile* untuk pengguna di dua platform platform yang berbeda yaitu android dan iOS. Ada juga aplikasi untuk mitra perusahaan *market place* tersebut. Semua aplikasi ini tentunya terhubung ke *database*, dimana untuk itu aplikasi yang berjalan di *smartphone* ini memerlukan aplikasi lain yang berjalan di sisi server.

Secara arsitektur, bagian-bagian dari sistem ini digolongkan menjadi backend dan frontend.



Gambar 2.1 Backend dan Frontend

Backend dan frontend di sini yang dimaksud adalah secara teknologi. Teknologi *backend* adalah segala macam teknologi yang berjalan di sisi server. Sebagai contoh untuk terhubung ke database, untuk mengirim email massal, untuk data analisis dan kebutuhan lain yang berjalan di sisi server. Teknologi backend seringnya berurusan dengan infrastruktur, seperti *scaling* database, integrasi dengan service lain, dan sebagainya. Adapun frontend adalah sebaliknya, yaitu berjalan di sisi client, dalam hal ini yang dimaksud adalah di *web* browser atau perangkat *mobile* yang digunakan oleh pengguna. Untuk aplikasi frontend sendiri bertanggung jawab terhadap optimasi di sisi *User Experience* [12] dan antarmuka pengguna.

Aplikasi frontend ini tidak terhubung langsung ke database karena berjalan di perangkat yang digunakan pengguna. Jadi untuk mengakses data yang dari database bisa memanfaatkan *web service* yang disediakan di aplikasi yang berjalan di server.

2.2.2 Single Page Application

Single Page Application adalah aplikasi yang berjalan di *web browser* dan tidak memuat ulang halaman ketika digunakan. Perubahan tampilan tidak memuat halaman melainkan mengubah halaman yang sedang ditampilkan. Perubahan tampilan ini ditangani secara terprogram. Javascript merupakan salah satu yang banyak digunakan untuk membuat web Single Page Application. Cara kerja seperti ini membuat web terasa seperti aplikasi desktop. Contoh web yang Single Page Application antara lain gmail dan goole maps dari Google.

2.2.3 Web Service

Web service merupakan standar pendistribusian layanan melalui internet [13]. Dalam penggunaan *web service*, client tidak perlu mengetahui tentang web service tersebut sebelum client benar-benar menggunakannya. Web service merupakan platform yang independen dan bersifat *loosely-coupled*. Karena sifatnya ini web service dapat diakses dengan mudah melalui berbagai macam platform, misalkan pada *smartphone [14]* dan service lain untuk pertukaran data antar service.

Karena *web service* ini akan digunakan untuk aplikasi saling berkomunikasi, dokumentasi untuk cara dan deskripsi *web service* ini sangatlah diperlukan.

2.2.4 Semantic Web Service

Dalam dokumentasi API *web service* menyajikan deskripsi, dan adanya arti dalam deskripsi-deskripsi inilah yang disebut semantic [10]. Misalnya adalah WSDL 2.0 (*Web Services Description Language*) [15]. WSDL menggunakan format XML untuk mendeskripsikan fungsionalitas *web service* secara abstrak

dan juga rincian seperti bagaimana dan dimana fungsi-fungsi yang tersedia.WSDL mendukung untuk SOAP dan REST, meskipun untuk REST jarang digunakan. Kemudian ada WADL (*Web Application Description Language*)[5] yang mampu menyajikan representasi terstruktur dan semantic dari API web service berbasis REST. WADL menggunakan format XML dan juga *machine-readable* yang secara eksplisit memang ditujukan untuk layanan API. WADL juga diusulkan untuk standardisasi, namun tidak ada tindak lanjut.

Yang lebih baru, lebih *human-readable* memperkenalkan format metadata dan juga tersedia tools editor untuk memudahkan pengembang dalam membuat dokumentasi REST API. Yang populer misal OpenAPI Specification (OAS)[16] atau dikenal juga sebagai Swagger. Swagger menggunakan format berbasis YAML atau bisa juga dengan JSON. Format deskripsi ini sintaksis, yang berarti bahwa sedikit dukungan untuk bisa mengotomatisasi operasi seperti pengenalan model data, komposisi dan juga verifikasi atas model data dari *web service* yang tersedia. Sehingga untuk membuat deskripsi atau dokumentasi API membutuhkan pekerjaan manual diluar pembuatan API *web service.*

2.2.5 GraphQL

Pada tahun 2016 GraphQL di-*release* oleh Facebook sebagai cara baru dalam mengakses data pada aplikasi *client-server* dengan menggunakan bahasa *query* yang intuitif dan fleksibel [7]. GraphQL menawarkan cara baru dalam membangun *web service*. *Web service* API dibuat dalam bentuk *schema* [17] yang merepresentasikan model data yang tersedia yang bisa diakses oleh

client. Schema ini menyediakan deskripsi API yang lengkap dan mudah dimengerti. Berikut contoh *schema* graphQL.

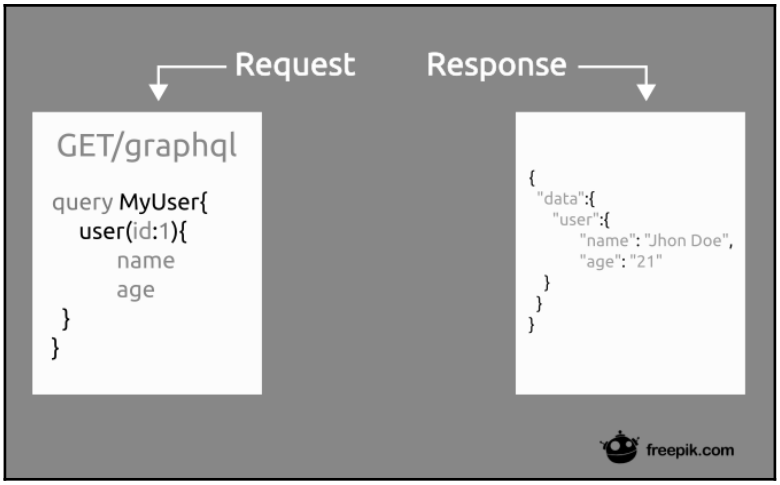
```
type User {
  name: String!
  username: String!
}
```

Contoh di atas merupakan *schema* User yang memiliki dua field bersifat *required* yang ditandai dengan tanda ! dan bertipe string.

GraphQL *schema* juga bisa berelasi. Contoh *schema* User di atas bisa berelasi dengan *schema* Document di bawah ini

```
type Document {
  title: String!
  content: String!
  author: User!
}
```

Cara mengakses API menggunakan query dengan format menyerupai JSON dan akan menerima data sesuai dengan yang diminta dengan query tersebut.



Contoh lebih nyata ada pada Github API versi 4 dimana sebelumnya yakni pada versi 3 Github menggunakan REST untuk api versi 3-nya, kemudian memilih untuk menggunakan GraphQL untuk API versi 4 karena dinilai lebih fleksibel dan lebih *scalable*[18].

2.3 Kerangka Pemikiran

Kerangka pemikiran dalam penelitian ini dapat digambarkan sebagai berikut.

Permasalahan
Bagaimana membangun *Web Service* yang semantic dan dapat digunakan secara efektif

Pendekatan
Pembuatan aplikasi backend menggunakan GraphQL untuk *web service* API

yang semantic

Pengembangan
Sisi Server : API berbasis GraphQL dengan NodeJS Sisi *Client* : web berupa *Single Page Application* dengan bahasa pemrograman javascript.

Pengujian
Menggunakan *unit test* untuk fungsi yang berjalan mandiri dan *integration test* untuk fungsi yang berhubungan dengan aplikasi lain

Hasil
Aplikasi backend dengan *web service* GraphQL

BAB III
METODE PENELITIAN

3.1 Instrumen Penelitian

Dalam melakukan penelitian yang dikerjakan ini diperlukan berbagai macam perangkat yang digunakan, yaitu:

1.5.1 Kebutuhan Software

Software atau perangkat lunak yang digunakan untuk mendukung penelitian ini ialah:

1. Sistem Operasi yang digunakan adalah Manjaro Linux 64 bit.
2. Visual Studio Code versi 1.35 untuk menulis kode program
3. NodeJS sebagai runtime untuk menjalankan kode program berbasis javascript beserta *tools-tools* yang diperlukan.
4. Postgres sebagai penyimpanan data yang diperlukan.
5. *Nginx* sebagai mesin untuk menjalankan aplikasi.

1.5.2 Kebutuhan Hardware

Hardware atau perangkat keras yang digunakan untuk mendukung penelitian ini ialah:

1. Komputer yang digunakan Laptop Thinkpad L440.
2. Prosesor Intel Core i5 generasi 4 vPro.
3. Kapasistas RAM 12GB.
4. Penyimpanan SSD Samsung EVO 850 250GB.

3.2 Jenis Dan Sumber Data

Penulis telah mengumpulkan beberapa jenis data sebagai acuan penelitian, data tersebut adalah data sekunder, yaitu data yang dijadikan landasan teori dan penunjang atau pelengkap data primer yang ada. Data sekunder didapatkan dari studi literatur dan dokumen penelitian terkait sebelumnya. Studi literatur digunakan untuk mencari penelitian *web service* yang menggunakan graphQL.

3.3 Teknik Pengumpulan Data

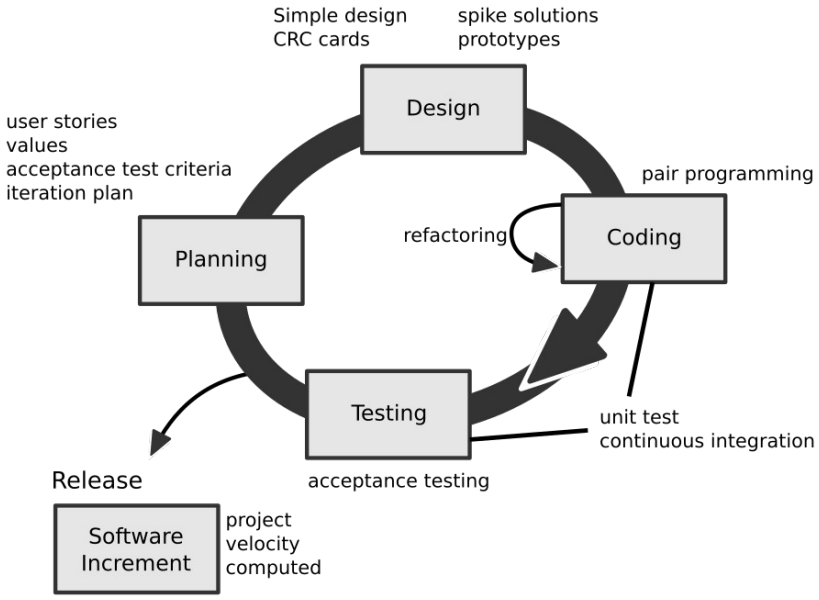
Metode yang digunakan untuk mengumpulkan data dalam melakukan penelitian ini adalah studi pustaka. Studi pustaka digunakan dengan cara mencari dan memahami teori-teori yang ada pada literatur terkait dengan penelitian yang dilakukan, salah satunya adalah penelitian "Toward Automatic Semantic API Descriptions to Support Services Composition". Penelitian ini membuat *web service* semantic dengan memanfaatkan swagger dengan API berbasis REST.

3.4 Metode Pengembangan Sistem

Untuk mengembangkan suatu sistem dengan baik dibutuhkan sebuah metode pengembangan yang sesuai dengan kebutuhan. Berdasarkan tinjauan studi yang penulis lakukan, penulis memutuskan untuk menggunakan metode Pemrograman Ekstrim (Extreme Programming) sebagai metode pengembangan *web service* berbasis GraphQL untuk Sistem Pengaduan Masyarakat dan Informasi Publik.

Extreme Programming merupakan metode pengembangan yang banyak digunakan dalam pengembangan aplikasi secara cepat, yang menyederhanakan tahapan-tahapan dalan proses pengembangan menjadi

lebih adaptif. Metode ini memiliki empat kegiatan utama yaitu perencanaan, perancangan, pengkodean, dan pengujian



Gambar 3.1: Extreme Programming

1.5.1 Planning / Perencanaan

Tahap perencanaan dimulai dengan mengumpulkan berbagai kebutuhan untuk pengembangan perangkat lunak dan menentukan output yang diharapkan. Berikut kebutuhan-kebutuhannya :

1. Sistem yang dikembangkan menghasilkan *web service* API yang dapat diakses.
2. *Web Service* API dapat diakses dari beragam aplikasi *client-side* yang berbeda (web dan *mobile*).
3. Pengambilan dan pengoperasian data dari aplikasi *client* menggunakan query GraphQL.

4. Data yang diterima dari *web service* API berformat JSON

1.5.2 Perancangan / Design

Tahap perancangan dilakukan untuk membuat alur perangkat lunak akan bekerja sesuai dengan kebutuhan pada tahap perencanaan. Perancangan dalam pemrograman ekstrim memiliki prinsip “penyederhanaan” atau *simplicity*. Rancangan yang sederhana akan memakan waktu pengembangan yang lebih sinkat dibanding dengan yang rumit. Pemrograman ekstrim merekomendasikan untuk membuat prototype yang operasional untuk kemudian dievaluasi. Solusi ini disebut dengan *spike.*

1.5.3 Pengkodean

Pengkodean adalah penulisan kode program untuk membangun aplikasi sesuai dengan rancangan. Untuk ini penulis menggunakan bahasa pemrograman javascript dengan nodeJS sebagai runtime untuk menjalankan kode program javascript di server.

1.5.4 Pengujian

Tahap pengujian ini berfungsi untuk menguji apakah kode program yang telah ditulis bisa bekerja sesuai yang diharapkan pada tahap perencanaan. Pegujian yang dilakukan adalah unit test dan *integration test.*

3.5 Metode Pengujian

Metode pengujian yang digunakan penulis dalam penelitian ini adalah black-box testing. Black-box testing merupakan pengujian sistem berdasarkan fungsionalitas dari spesifikasi kebutuhan tanpa melakukan pengecekan kode. Black-box testing melakukan pengujian berdasarkan sudut pandang penggunaan[19]

Kemudian penulis juga menggunakan white-box testing. White-box testing merupakan pengujian yang berbasis path. Pengguaan metode barbasis path ini memungkinkan untuk menentukan tingkat kompeksitas dalam level komponen, fungsi dan logic[20]. Dalam extreme programming, white-box tesing dapat menggunakan unit test, yang bisa dilakukan menggunakan beragam *framework* yang sudah ada hingga pengujiannya dapat diotomatisasi.

DAFTAR PUSTAKA

[1] N. Vesyropoulos, C. K. Georgiadis, dan P. Katsaros, “Ensuring business and service requirements in enterprise mashups,” *Inf. Syst. E-Bus. Manag.*, vol. 16, no. 1, hlm. 205–242, Feb 2018.

[2] G. Ghiani, F. Paternò, L. D. Spano, dan G. Pintori, “An environment for End-User Development of Web mashups,” *Int. J. Hum.-Comput. Stud.*, vol. 87, hlm. 38–64, Mar 2016.

[3] V. Hoyer dan M. Fischer, “Market Overview of Enterprise Mashup Tools,” dalam *Service-Oriented Computing – ICSOC 2008*, 2008, hlm. 708–721.

[4] D. Bianchini, V. De Antonellis, dan M. Melchiori, “Semantics-Enabled Web API Organization and Recommendation,” dalam *Advances in Conceptual Modeling. Recent Developments and New Directions*, 2011, hlm. 34–43.

[5] M. J. Hadley, “Web Application Description Language (WADL),” Jan 2009.

[6] K. Soames dan J. Lind, *Detecting Cycles in GraphQL Schemas*. 2019.

[7] O. Hartig dan J. Pérez, “An initial analysis of Facebook’s GraphQL language,” dalam *AMW 2017 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017.*, 2017, vol. 1912.

[8] E. Stenlund dan K. Gustavsson, “Efficent data communication between a webclient and a cloud environment,” 2016.

[9] M. Vogel, S. Weber, dan C. Zirpins, “Experiences on Migrating RESTful Web Services to GraphQL,” dalam *Service-Oriented Computing – ICSOC 2017 Workshops*, 2018, hlm. 283–295.

[10] M. Cremaschi dan F. De Paoli, “Toward Automatic Semantic API Descriptions to Support Services Composition,” dalam *Service-Oriented and Cloud Computing*, 2017, hlm. 159–167.

[11] P. R. ANDY, “Implementasi Webservice untuk Sinkronisasi Data Trankskrip Nilai di Tata Usaha Fakultas Universitas Dian Nuswantoro,” *SkripsiFakultas Ilmu Komput.*, 2016.

[12] C. Lallemand, G. Gronier, dan V. Koenig, “User experience: A concept without consensus? Exploring practitioners’ perspectives through an international survey,” *Comput. Hum. Behav.*, vol. 43, hlm. 35–48, Feb 2015.

[13] Snehal Mumbaika dan Puja Padiya, “Web services based on soap and rest principles,” *Int. J. Sci. Res. Publ. IJSRP*, vol. 3, no. 5, Mei 2013.

[14] F. N. Rofiq dan A. Susanto, “Implementasi RESTful Web Service untuk Sistem Penghitungan Suara Secara Cepat pada Pilkada,” *J. Eksplora Inform.*, vol. 6, no. 2, hlm. 159–168, Mar 2017.

[15] R. Chinnici, J. J Moreau, A. Ryman, dan S. Weerawarana, “Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language,” *W3C Work. Draft*, vol. 26, Jan 2004.

[16] “Home,” *OpenAPI Initiative*. [Daring]. Tersedia pada: https://www.openapis.org/. [Diakses: 15-Jul-2019].
[17] “GraphQL: A query language for APIs.” [Daring]. Tersedia pada: http://graphql.org/. [Diakses: 15-Jul-2019].
[18] “The GitHub GraphQL API,” *The GitHub Blog*, 14-Sep-2016. .
[19] S. Nidhra, “Black Box and White Box Testing Techniques - A Literature Review,” *Int. J. Embed. Syst. Appl.*, vol. 2, hlm. 29–50, Jun 2012.
[20] “(PDF) A Comparative Study of White Box, Black Box and Grey Box Testing Techniques.” [Daring]. Tersedia pada: https://www.researchgate.net/publication/270554162_A_Comparative_Study_of_White_Box_Black_Box_and_Grey_Box_Testing_Techniques. [Diakses: 23-Sep-2019].