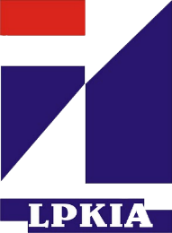
**IMPLEMENTASI ARSITEKTUR MICROSERVICES MENGGUNAKAN RESTFUL WEB SERVICES  
(STUDI KASUS YAYASAN ANAK SHALEH BANDUNG)  
  
IMPLEMENTATION OF ARCHITECTURAL MICROSERVICES USING RESTFUL WEB SERVICES  
(STUDI KASUS YAYASAN ANAK SHALEH BANDUNG)**

**SKRIPSI**

Diajukan untuk memenuhi salah satu persyaratan  
kelulusan program Sarjana Strata-1 (S1)

Disusun oleh

**ALDI ARDIYANTO  
NRP. 161014047**



**SEKOLAH TINGGI MANAJEMEN INFORMATIKA  
D A N I L M U K O M P U T E R L P K I A   
PROGRAM STUDI TEKNIK INFORMATIKA  
BANDUNG  
2020**

# DAFTAR ISI

[DAFTAR ISI 2](#_Toc46440725)

[BAB I PENDAHULUAN 5](#_Toc46440726)

[I.1 Latar Belakang Masalah 5](#_Toc46440727)

[I.2 Identifikasi Permasalahan 5](#_Toc46440728)

[I.3 Ruang Lingkup 5](#_Toc46440729)

[I.4 Tujuan Perancangan 5](#_Toc46440730)

[I.5 Metodologi Penelitian 5](#_Toc46440731)

[I.6 Sistematika Penulisan 5](#_Toc46440732)

[BAB II DASAR TEORI 7](#_Toc46440733)

[II.1 Teori Tentang Permasalahan 7](#_Toc46440734)

[II.1.1 Implementasi 7](#_Toc46440735)

[II.1.2 Microservices 7](#_Toc46440736)

[II.1.3 REST 7](#_Toc46440737)

[II.1.4 Web Services 7](#_Toc46440738)

[II.1.5 JSON 7](#_Toc46440739)

[II.1.6 Apache 7](#_Toc46440740)

[II.1.7 Nodej 7](#_Toc46440741)

[II.1.8 PHP 7](#_Toc46440742)

[II.1.9 Javascript 7](#_Toc46440743)

[II.1.10 Laravel 7](#_Toc46440744)

[II.1.11 Express 7](#_Toc46440745)

[II.1.12 API 7](#_Toc46440746)

[II.1.13 Postman 7](#_Toc46440747)

[II.2 Metodologi Yang Digunakan 7](#_Toc46440748)

[II.2.1 Pengembangan Perangkat Lunak (*Waterfall*) 7](#_Toc46440749)

[II.2.2 Unified Modeling Language (UML) 10](#_Toc46440750)

[II.2.3 Teknik Pengumpulan Data Yang Digunakan 12](#_Toc46440751)

[BAB III ANALISIS DAN PERANCANGAN 15](#_Toc46440752)

[III.1 Aliran Fungsional 15](#_Toc46440753)

[III.1.1 Use Case Diagram 15](#_Toc46440754)

[III.1.2 Use Case Scenario 17](#_Toc46440755)

[III.2 Aliran Kerja 24](#_Toc46440756)

[III.2.1 Activity *Diagram* 24](#_Toc46440757)

[III.2.2 Uraian Workflow 24](#_Toc46440758)

[III.3 Pemodelan Struktur Sistem dan Data 24](#_Toc46440759)

[III.3.1 *Class* Diagram 24](#_Toc46440760)

[III.3.2 *Class* Object *Description* 24](#_Toc46440761)

[III.4 Interaksi Antar Class 24](#_Toc46440762)

[III.4.1 Sequence Diagram 24](#_Toc46440763)

[III.5 Pemodelan Perilaku Sistem 25](#_Toc46440764)

[*III.5.1* *State diagram* 25](#_Toc46440765)

[III.6 Perancangan Endpoint Microservices 25](#_Toc46440766)

[III.6.1 Api Gateway 25](#_Toc46440767)

[III.6.2 Blog Service 25](#_Toc46440768)

[III.6.3 User Service 27](#_Toc46440769)

[III.6.4 Program Service 27](#_Toc46440770)

[III.6.5 Donation Service 28](#_Toc46440771)

[III.7 Pdeudecode 28](#_Toc46440772)

[*III.7.1* Proses *Spesification* 28](#_Toc46440773)

[BAB IV IMPLEMENTASI DAN PENGUJIAN 29](#_Toc46440774)

[IV.1 Implementasi 29](#_Toc46440775)

[IV.1.1 Lingkup dan Batasan 29](#_Toc46440776)

[IV.1.2 Kebutuhan Sumber Daya 29](#_Toc46440777)

[IV.1.3 Imlementasi aplikasi 30](#_Toc46440778)

[IV.2 Pengujian 30](#_Toc46440779)

[IV.2.1 Lingkup dan Lingkungan 30](#_Toc46440780)

[IV.2.2 Kebutuhan Sumber Daya 30](#_Toc46440781)

[IV.2.3 Hasil Pengujian 30](#_Toc46440782)

[BAB V KESIMPULAN DAN SARAN 31](#_Toc46440783)

[V.1 Kesimpulan 31](#_Toc46440784)

[V.2 Saran 31](#_Toc46440785)

[DAFTAR PUSTAKA 32](#_Toc46440786)

[LAMPIRAN 33](#_Toc46440787)

# BAB I PENDAHULUAN

## Latar Belakang Masalah

Sistem informasi pada umumnya dibangun dengan pendekatan monolitik, dimana aplikasi terbungkus dalam satu package besar, dan ketika terjadi perubahan (*requirement changes*) pada salah satu bagian kode program, akan berpengaruh besar terhadap kode program lainnya. Saat ini pendekatan monolitik telah bergeser menjadi pendekatan terdistribusi, dimana aplikasi dibagi menjadi bagian-bagian kecil yang berfungsi spesifik (*high cohesion*) dan tidak bergantung pada komponen program lainnya (*loose coupling*), dengan dikembangkannya service melalui antarmuka API (*application programming interface*)[1].

Arsitektur microservice merupakan alternatif arsitektur yang lebih terukur dan lebih fleksibel. Pada arsitektur microservice, sistem informasi dirancang untuk terdistribusi dan menyediakan layanan secara lebih fokus dan spesifik. Permasalahan besar akan dipecah menjadi beberapa solusi kecil yang disusun dalam satu service, dimana setiap service memiliki tanggung jawabnya sendiri.

Karena sifatnya yang independen atau tidak saling ketergantungan antara modul-modulnya

Yayasan Anak Shaleh memiliki sistem informasi yang didalamnya terdapat beberapa modul yaitu blog, donator, program, donasi, dan pembayaran. Dalam pengembangannya Yayasan Anak Shaleh membutuhkan suatu arsitektur untuk menjadikan modul-modul didalam sistemnya fleksibel terhadap penggunaan teknologi dan tidak saling ketergantungan antara modul. Oleh karena itu, implementasi Arsitektur Microservices dengan menggunakan RESTful Web Services layak diterapkan dalam menghadapi sistem informasi yang ada di Yayasan Anak Shaleh.

## Identifikasi Permasalahan

Berdasarkan latar belakang yang telah diuraikan diatas, maka di dapatkan indentifikasi masalah sebagai berikut:

1. Sistem informasi yang dimiliki oleh Yayasan Anak Shaleh tidak fleksibel terhadap penggunaan teknologi.
2. Komponen-komponen pada sistem informasi yang ada di Yayasan Anak Shaleh masih saling ketergantungan antara satu dengan yang lainnya.

## Ruang Lingkup

Ruang lingkup pada penelitian ini adalah sebagai berikut:

1. Penerepan fokus pada implementasi *microservices* menggunakan RESTful web services.
2. Service-service yang dirancang diperuntukan untuk modul-modul yang ada pada website Yayasan Anak Shaleh.

## Tujuan Perancangan

Tujuan dari penelitian ini adalah menerapkan arsitektur m*icroservices* dengan menggunakan RESTful web *services* sebagai arsitektur komunikasi antar servicenya sehingga lebih fleksibel terhadap penggunaan teknologi.

## Metodologi Penelitian

## Sistematika Penulisan

Untuk melakukan penganalisaan dan pemecahan persoalan secara lebih terinci dan mempermudah dalam penalaran, maka penulisan dan pembahasan disusun menurut sistematika sebagai berikut:

**BAB I PENDAHULUAN**

Bab ini digunakan untuk mendefinisikan persoalan, ruang lingkup dan perencanaan kegiatan atau proyek yang dilakukan. Bab ini berisi latar belakang, indentifikasi persoalan, lingkup dan batasan, tujuan, dan sistematika penulisan.

**BAB II LANDASAN TEORI**

Bab ini digunakan untuk mendefinisikan teori tentang permasalahan yang dibahas dalam penelitian ini, diantaranya adalah pengertian perangkat lunak, microservices, web services, RESTful, Application Programming Interface dan Api Gateway yang digunakan untuk mendukung penelitian ini.

**BAB III ANALISIS DAN PEMBAHASAN**

Bab ini digunakan untuk mendefinisikan semua kebutuhan dan meletakan dasar-dasar untuk proses perancangan perangkat lunak. Bab ini juga menjabarkan tentang pengembangan arsitektur microservices untuk memecahkan suatu persoalan.

**BAB IV IMPLEMENTASI DAN PENGUJIAN**

Bab ini berisi mengenai kegiatan dalam membangun atau mewujudkan perancangan secara nyata.

**BAB V KESIMPULAN DAN SARAN**

Bab ini berisi kesimpulan dan saran untuk kajian yang dapat dituliskan.

# BAB II DASAR TEORI

## Teori Tentang Permasalahan

### Implementasi

### Microservices

### REST

### Web Services

### JSON

### Apache

### Nodejs

### PHP

### Javascript

### Laravel

### Express

### API

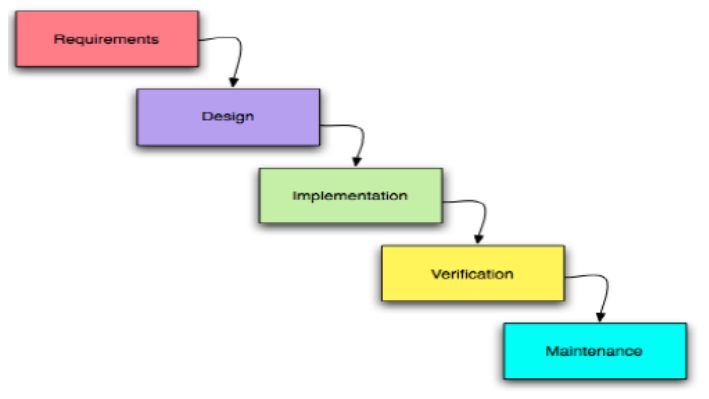
### Postman

## Metodologi Yang Digunakan

Sub bab ini menjelaskan tentang landasan teoritis dari metodologi yang digunakan yakni model *waterfall* yang digunakan dalam perancangan sistem. Pada sub bab ini juga dijabarkan tools atau alat yang digunakan untuk tahapan perancangan yaitu menggunakan SDLC.

### Pengembangan Perangkat Lunak (*Waterfall*)

Menurut Imam Fahrurrozi dan Azhari, Model *waterfall* adalah proses pengembangan perangkat lunak tradisional yang umum digunakan dalam proyek-proyek perangkat lunak yang paling pembangunan. Ini adalah model sekuensial, sehingga penyelesaian satu set kegiatan menyebabkan dimulainyaaktivitas berikutnya. Hal ini disebut *waterfall* karena proses mengalir "secara sistematis dari satu tahap ke tahap lainnya dalam mode ke bawah ". Membentuk kerangka kerja untuk pengembangan perangkat lunak. Beberapa varian dari model ada, setiap label yang berbeda menggunakan untuk setiap tahap. Secara umum, bagaimanapun, model ini dianggap memiliki enam tahap yang berbeda seperti yang ditunjukkan pada Gambar yaitu: analisis Kebutuhan, desain, implementasi, verifikasi, instalasi dan pemeliharaan.



Gambar *Waterfall* Model

1. Kebutuhan berbasis Pengujian

Ini adalah langkah pertama dan paling penting dari model *waterfall*. Ini melibatkan pengumpulan informasi mengenai solusi akhir dari kebutuhan pelanggan pelanggan dan pemahaman. Ini melibatkan definisi yang jelas tentang tujuan pelanggan, harapan terhadap proyek dan masalah produk akhir diharapkan untuk memecahkan. "Analisis meliputi pemahaman konteks bisnis pelanggan dan kendala, fungsi produk harus melakukan, tingkat kinerja itu harus mematuhi dan sistem eksternal itu harus sesuai dengan".

Elisitasi persyaratan adalah proses mengumpulkan informasi dari para pemangku kepentingan dari sistem. Beberapa teknik digunakan untuk elisitasi adalah pelanggan wawancara, prototyping cepat, kasus penggunaan dan brainstorming. Kasus penggunaan umumnya persyaratan fungsional dari sistem.

1. Desain

Langkah ini dimulai dengan menggunakan informasi yang ditangkap di SRS. Ini dapat dianggap sebagai memberikan solusi untuk masalah di lingkup menggunakan sumber daya yang tersedia. Tahap ini terdiri dari bagaimana perangkat lunak akan dibangun, dengan kata lain perencanaan solusi perangkat lunak. Para pemangku kepentingan yang terlibat dalam modul ini adalah para perancang sistem. Desain perangkat lunak mungkin mencakup desain sistem dan desain komponen. "Tahap desain melibatkan mendefinisikan perangkat keras dan perangkat lunak arsitektur, menentukan kinerja dan parameter keamanan, merancang kontainer penyimpanan data dan kendala, memilih IDE dan bahasa pemrograman, dan menunjukkan strategi untuk menghadapi masalah-masalah seperti penanganan eksepsi, pengelolaan sumber daya dan konektivitas antarmuka". Di sinilah desain pengguna sistem juga diakui tergantung pada kemudahan akses dan navigasi. Output dari tahap ini adalah satu atau lebih software desain deskripsi (SDD) yang berfungsi sebagai masukan untuk tahap berikutnya. Pada 9 Maret 2009, Google merilis Android versi 1.1. Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan email.

1. Implementasi

Masukan dari fase ini adalah SDD sistem. "Di sinilah perkembangan aktual sistem terjadi sesuai dengan spesifikasi desain. Langkah ini dilakukan oleh pengembang, desainer interface dan stakeholder lainnya dengan menggunakan alat seperti compiler, debugger, penerjemah dan editor media. Output dari langkah ini adalah komponen produk satu atau lebih yang dibangun berdasarkan standar yang telah ditetapkan coding dan perbaikan, pengujian dan terintegrasi untuk memenuhi kebutuhan arsitektur sistem ".

1. Pengujian: Verifikasi dan Validasi

Pada fase ini kedua komponen individu dan solusi terintegrasi yang diverifikasi untuk melihat itu adalah bug gratis dan memenuhi spesifikasi kebutuhan perangkat lunak. Tester adalah stakeholder yang terlibat dalam fase model. Uji kasus ditulis untuk mengevaluasi apakah sistem sepenuhnya atau sebagian memenuhi persyaratan sistem.

Pengujian dapat dikategorikan ke dalam unit testing (dilakukan pada modul tertentu kode), sistem pengujian (untuk melihat bagaimana sistem bereaksi ketika semua modul yang terintegrasi) dan penerimaan pengujian (dilakukan dengan atau nama pelanggan untuk melihat apakah semua kebutuhan pelanggan puas). Cacat yang ditemukan pada tahap ini diberikan sebagai umpan balik kepada para pengembang yang pada gilirannya memperbaiki masalah. Ini adalah tahap di mana produk dikembangkan didokumentasikan. Buku pedoman ini diterbitkan pada fase ini.

1. Maintenance

Instalasi Fase ini terjadi setelah akhir-produk telah diuji dan disetujui oleh pelanggan. "Langkah ini melibatkan penyusunan sistem atau produk untuk instalasi dan penggunaan di lokasi pelanggan". Pengiriman produk dilakukan melalui internet atau melalui metode fisik. Sejumlah revisi biasanya ditandai samping diserahkan untuk memfasilitasi update atau perubahan pada tahap berikutnya .

Pemeliharaan Ini adalah tahap akhir dari model waterfall dan terjadi setelah instalasi sistem produk / di lokasi pelanggan. "Tahap ini melibatkan membuat modifikasi pada sistem atau komponen individu untuk mengubah atribut atau meningkatkan kinerja sistem". Modifikasi yang muncul karena perubahan permintaan dipicu oleh pelanggan atau cacat yang ditemukan saat menggunakan sistem secara real time. Nomor revisi diperbarui dalam setiap rilis pemeliharaan .

### Unified Modeling Language (UML)

Menurut Prastuti Sulistyorini, *Unified Modelling Language* (UML) adalah sebuah “bahasa” yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML dapat dibuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun.

1. *Class Diagram*

*Class Diagram* bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi serta relasi.

1. *Object Diagram*

*Object Diagram* bersifat statis. Diagram ini memperlihatkan objek-objek serta relasi antar objek. Diagram objek memperlihatkan instansiasi statis dari segala sesuatu yang dijumpai pada diagram kelas.

1. *Use case Diagram*

Diagram ini bersifat statis. Diagram ini memperlihatkan himpunan *use case* dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku dari suatu sistem yang dibutuhkan serta diharapkan pengguna.

1. *Sequence Diagram* (Diagram urutan)

Diagram ini bersifat dinamis. *Sequence Diagram* merupakan diagram interaksi yang menekankan pada pengiriman pesan (*message*) dalam suatu waktu tertentu.

1. *Collaboration Diagram*

Diagram ini bersifat dinamis. Diagram kolaborasi adalah diagram interaksi yang menekankan organisasi struktural dari objek – objek yang menerima serta mengirim pesan (*message*).

1. *Statechart Diagram*

Diagram ini bersifat dinamis. Diagram ini memperlihatkan state – state pada sistem, memuat state, transisi, event, serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka, kelas, kolaborasi dan terutama penting pada pemodelan sistem – sistem yang reaktif.

1. *Activity Diagram*

Diagram ini bersifat dinamis. Diagram ini adalah tipe khusus dari diagram state yang memperlihatkan aliran dari suatu aktifitas ke aktifitas lainnya dari suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi – fungsi dalam suatu sistem dan memberi tekanan pada aliran kendali antar objek.

1. *Component Diagram*

Diagram ini bersifat statis. Diagram ini memperlihatkan organisasi serta kebergantungan pada komponen – komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan ke dalam satu atau lebih kelas- kelas, antarmuka – antarmuka serta kolaborasi – kolaborasi.

1. *Deployment Diagram*

Diagram ini bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (saat *run time*). Dengan ini memuat simpul – simpul (*node*) beserta komponen – komponen yang ada di dalamnya. *Deployment diagram* berhubungan erat dengan diagram kompoen dimana deployment diagram memuat satu atau lebih komponen – komponen. Diagram ini sangat berguna saat aplikasi berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computing*)

### Teknik Pengumpulan Data Yang Digunakan

1. **Wawancara**

Menurut Rachmawati, Wawancara pada penelitian kualitatif memiliki sedikit perbedaan dibandingkan dengan wawancara lainnya seperti wawancara pada penerimaan pegawai baru, penerimaan mahasiswa baru, atau bahkan pada penelitian kuantitatif. Wawancara pada penelitian kualitatif merupakan pembicaraan yang mempunyai tujuan dan didahului beberapa pertanyaan informal.

Wawancara penelitian lebih dari sekedar percakapan dan berkisar dari informal ke formal. Walaupun semua percakapan mempunyai aturan peralihan tertentu atau kendali oleh satu atau partisipan lainnya, aturan pada wawancara penelitian lebih ketat. Tidak seperti pada percakapan biasa, wawancara penelitian ditujukan untuk mendapatkan informasi dari satu sisi saja, oleh karena itu hubungan asimetris harus tampak. Peneliti cenderung mengarahkan wawancara pada penemuan perasaan, persepsi, dan pemikiran partisipan.

Uraian berikut akan menggambarkan jenis wawancara, jenis pertanyaan, lama waktu wawancara, dan prosedur melakukan wawancara pada penelitian kualitatif. Penjelasan tentang pengumpulan data merupakan hal yang penting karena akan menuntun pembaca memahami proses penelitian secara tepat.

Jenis wawancara :

1. Wawancara tidak berstruktur tidak berstandard, informal, atau berfokus

Dimulai dari pertanyaan umum dalam area yang luas pada penelitian. Wawancara ini biasanya diikuti oleh suatu kata kunci, agenda atau daftar topik yang akan dicakup dalam wawancara. Namun tidak ada pertanyaan yang ditetapkan sebelumnya kecuali dalam wawancara yang awal sekali.

Jenis wawancara ini bersifat fleksibel dan peneliti dapat mengikuti minat dan pemikiran partisipan. Pewawancara dengan bebas menanyakan berbagai pertanyaan kepada partisipan dalam urutan manapun bergantung pada jawaban. Hal ini dapat ditindaklanjuti, tetapi peneliti juga mempunyai agenda sendiri yaitu tujuan penelitian yang dimiliki dalam pikirannya dan isyu tertentu yang akan digali.

Wawancara jenis ini terutama cocok bila peneliti mewawancarai partisipan lebih dari satu kali. Wawancara ini menghasilkan data yang terkaya, tetapi juga memiliki *dross rate* tertinggi, terutama apabila pewawancaranya tidak berpengalaman. *Dross rate* adalah jumlah materi atau informasi yang tidak berguna dalam penelitian.

1. Wawancara Semi Berstruktur

Wawancara ini dimulai dari isu yang dicakup dalam pedoman wawancara. Pedoman wawancara bukanlah jadwal seperti dalam penelitian kuantitatif. Sekuensi pertanyaan tidaklah sama pada tiap partisipan bergantung pada proses wawancara dan jawaban tiap individu. Namun pedoman wawancara menjamin peneliti dapat mengumpulkan jenis data yang sama dari partisipan. Peneliti dapat menghemat waktu melalui cara ini. *Dross rate* lebih rendah daripada wawancara tidak berstruktur. Peneliti dapat mengembangkan pertanyaan dan memutuskan sendiri mana isu yang dimunculkan.

1. Wawancara berstruktur atau berstandard.

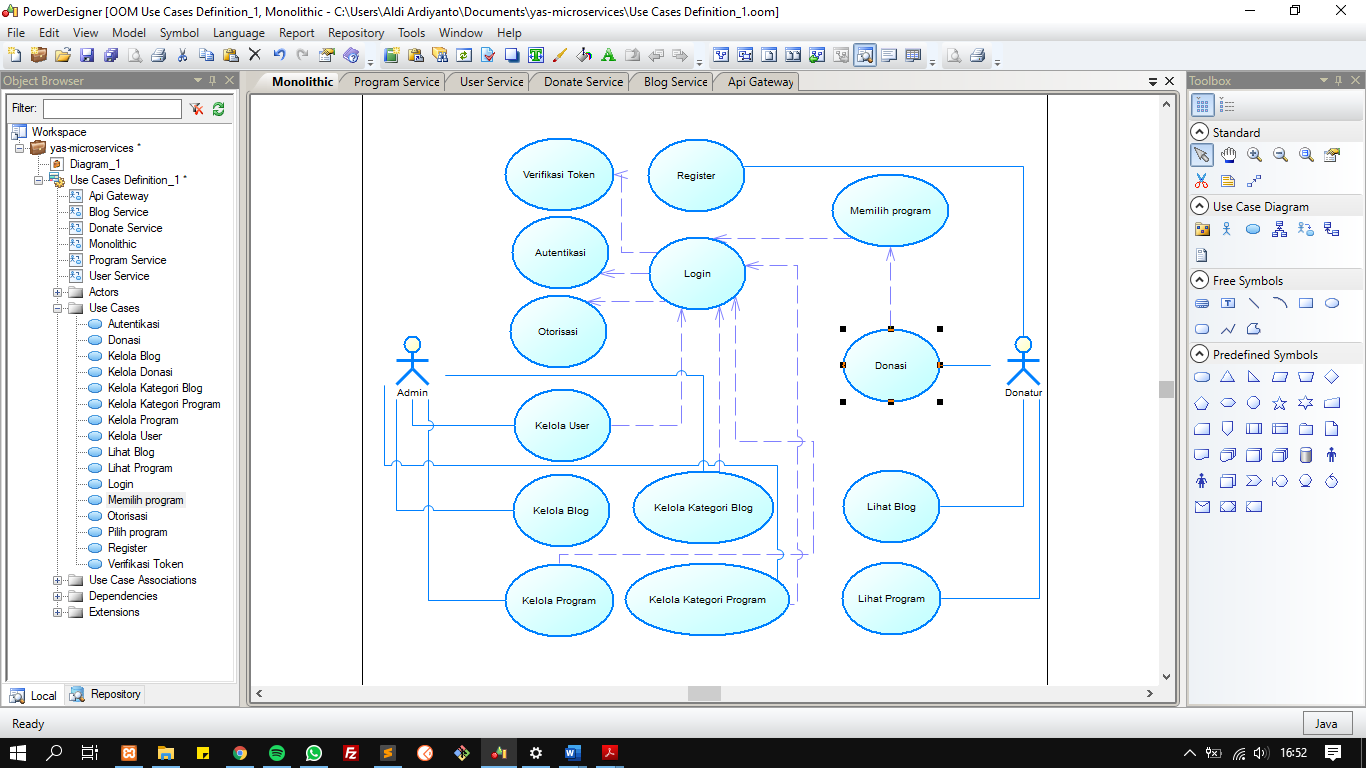
Peneliti kualitatif jarang menggunakan jenis wawancara ini. Beberapa keterbatasan pada wawancara jenis ini membuat data yang diperoleh tidak kaya. Jadwal wawancara berisi sejumlah pertanyaan yang telah direncanakan sebelumnya. Tiap partisipan ditanyakan pertanyaan yang sama dengan urutan yang sama pula. Jenis wawancara ini menyerupai kuesioner survei tertulis. Wawancara ini menghemat waktu dan membatasi efek pewawancara bila sejumlah pewawancara yang berbeda terlibat dalam penelitian. Analisis data tampak lebih mudah sebagaimana jawaban yang dapat ditemukan dengan cepat. Umumnya, pengetahuan statistik penting dan berguna untuk menganalisis jenis wawancara ini. Namun jenis wawancara ini mengarahkan respon partisipan dan oleh karena itu tidak tepat digunakan pada pendekatan kualitatif. Wawancara berstruktur bisa berisi pertanyaan terbuka, namun peneliti harus diingatkan terhadap hal ini sebagai isu metodologis yang akan mengacaukan dan akan jadi menyulitkan analisisnya. Peneliti kualitatif menggunakan pertanyaan yang berstruktur ini hanya untuk mendapatkan data *sosio- demografik*, seperti usia, lamanya kondisi yang dialami, lamanya pengalaman, pekerjaan, kualifikasi, dsb. Kadang komite etik menanyakan jadwal wawancara yang ditentukan sebelumya sehingga mereka dapat menemukan alur penelitian yang sebenarnya. Pada kasus ini, pedoman wawancara semi berstruktur lebih dianjurkan.

# BAB III ANALISIS DAN PERANCANGAN

## Aliran Fungsional

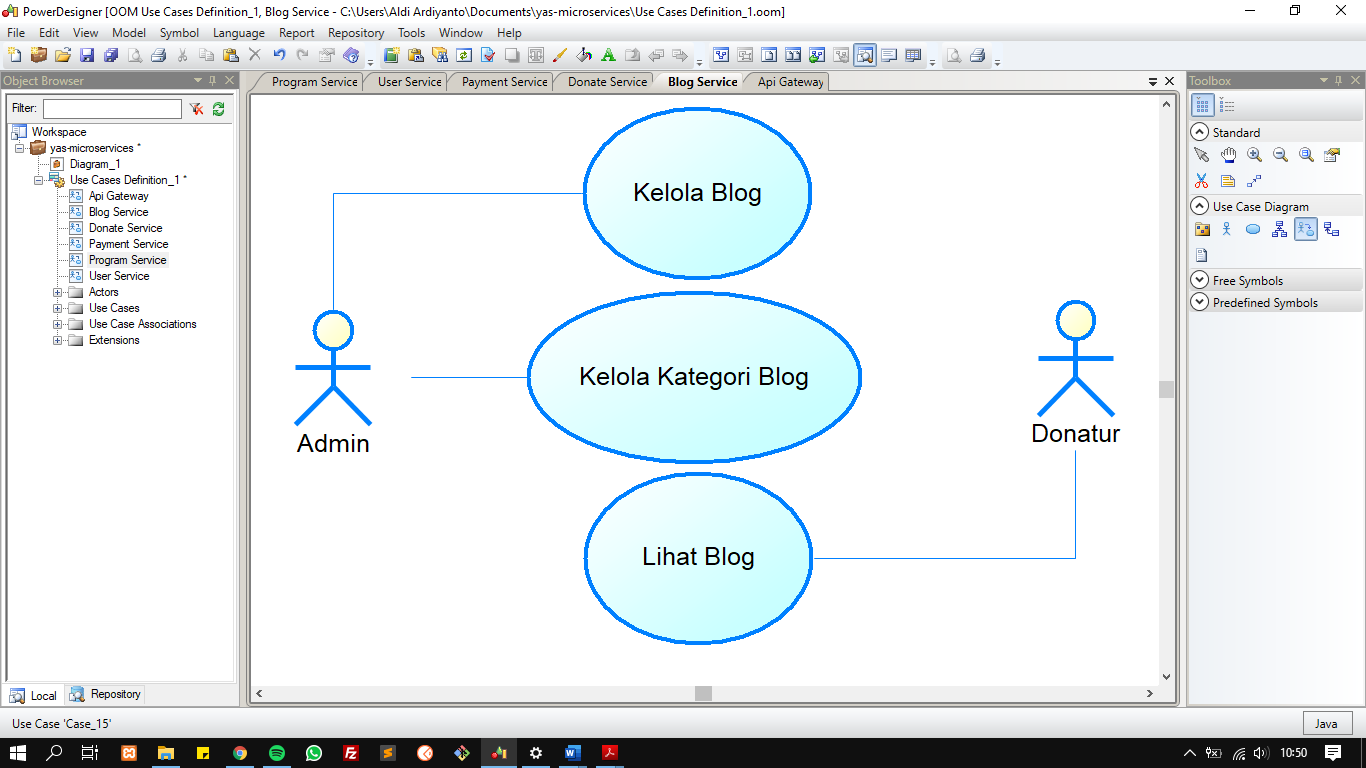
### Use Case Diagram

1. Use Case Diagram Arsitektur Monolotik



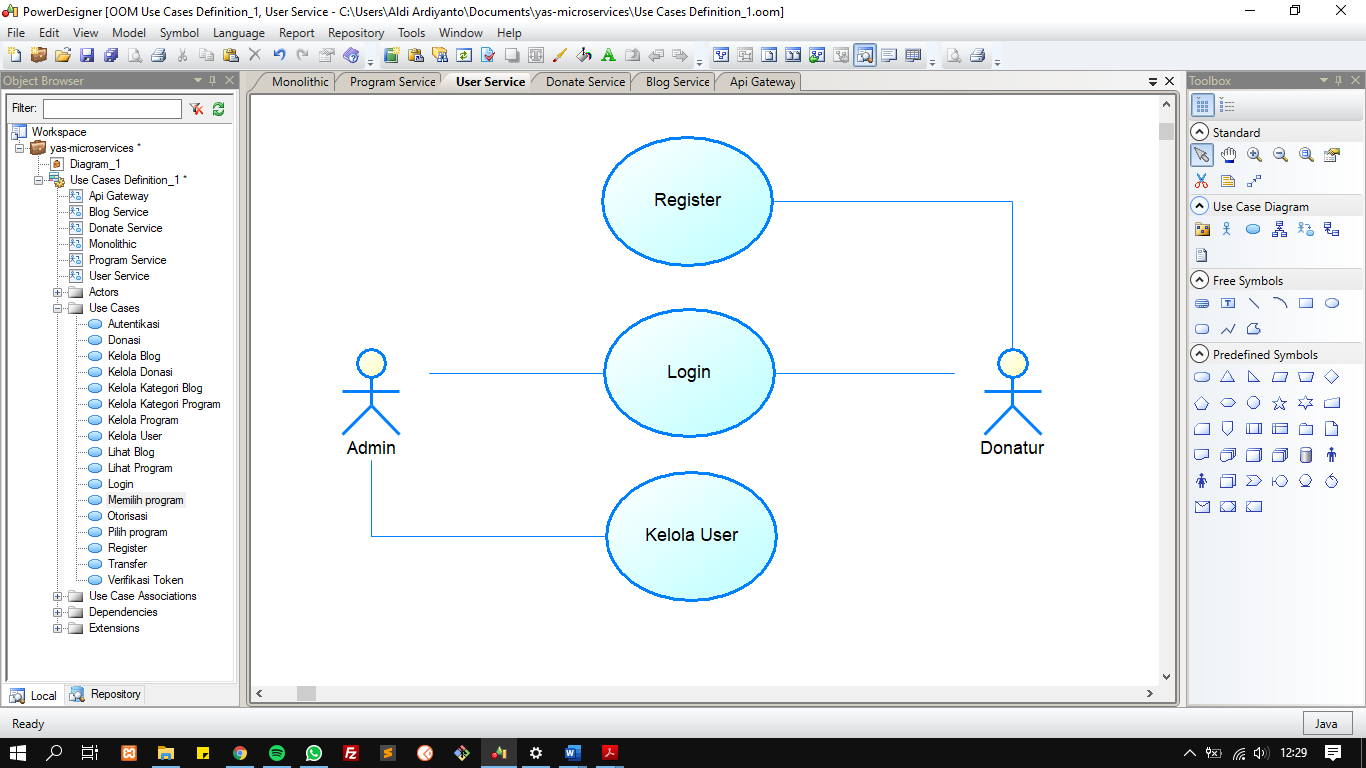
Use Case Diagram Arsitektur Monolitik

1. Use Case Diagram Blog Service



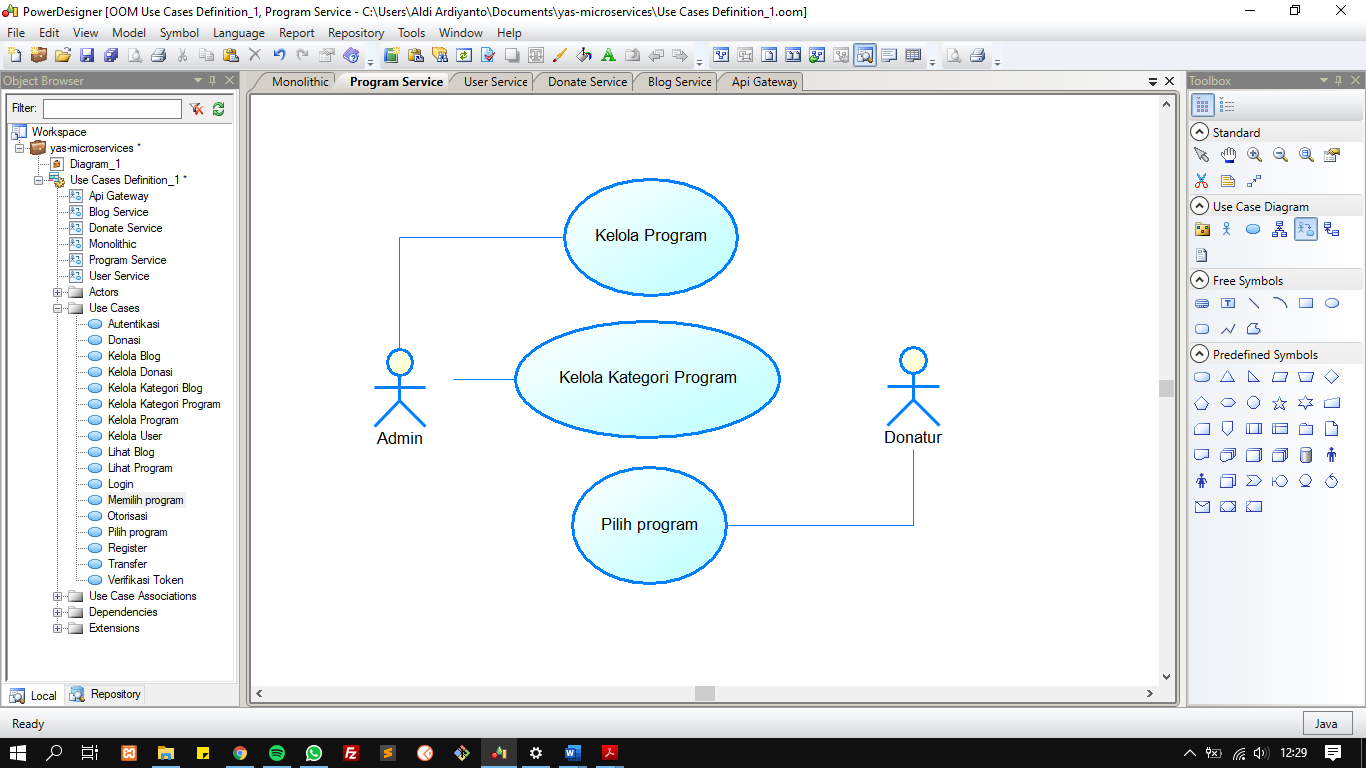
Use Case Diagram Blog Service

1. Use Case Diagram User Service



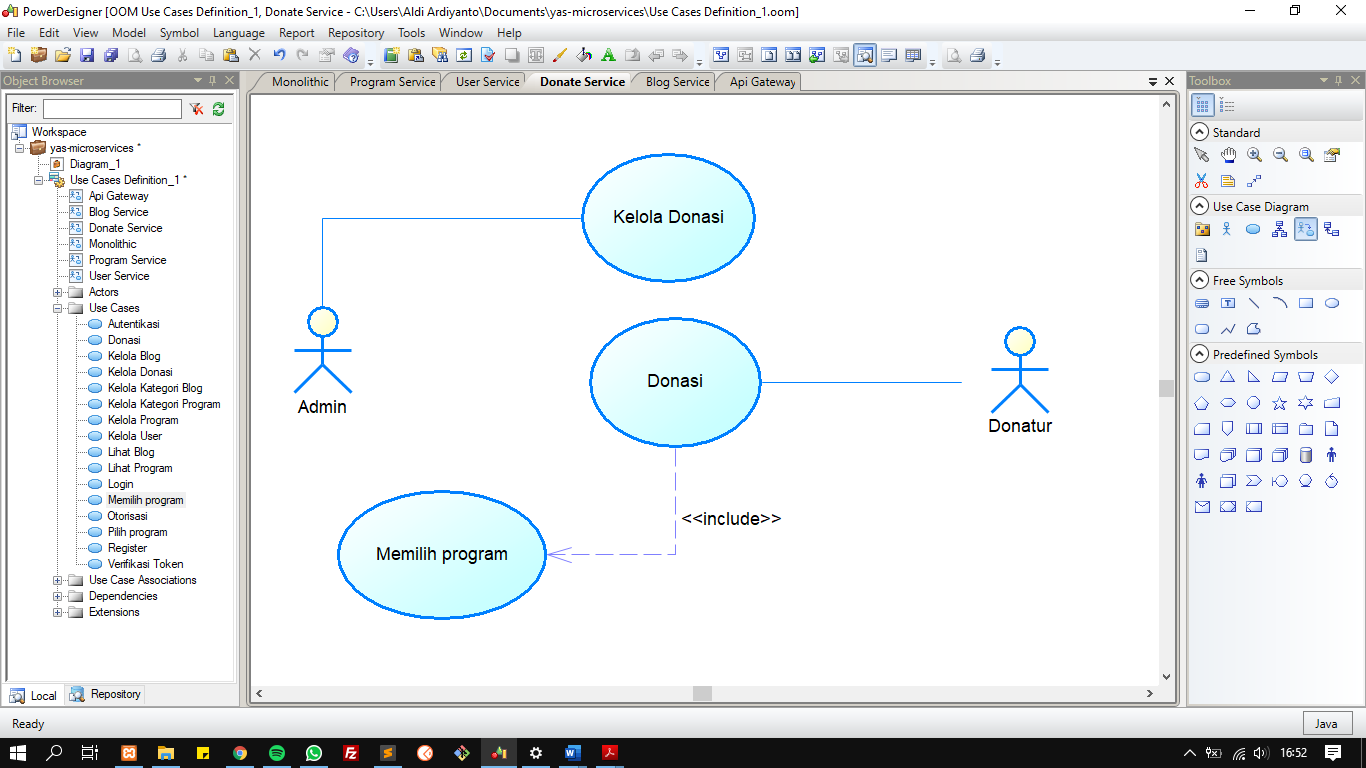
Use Case Diagram User Service

1. Use Case Diagram Program Service



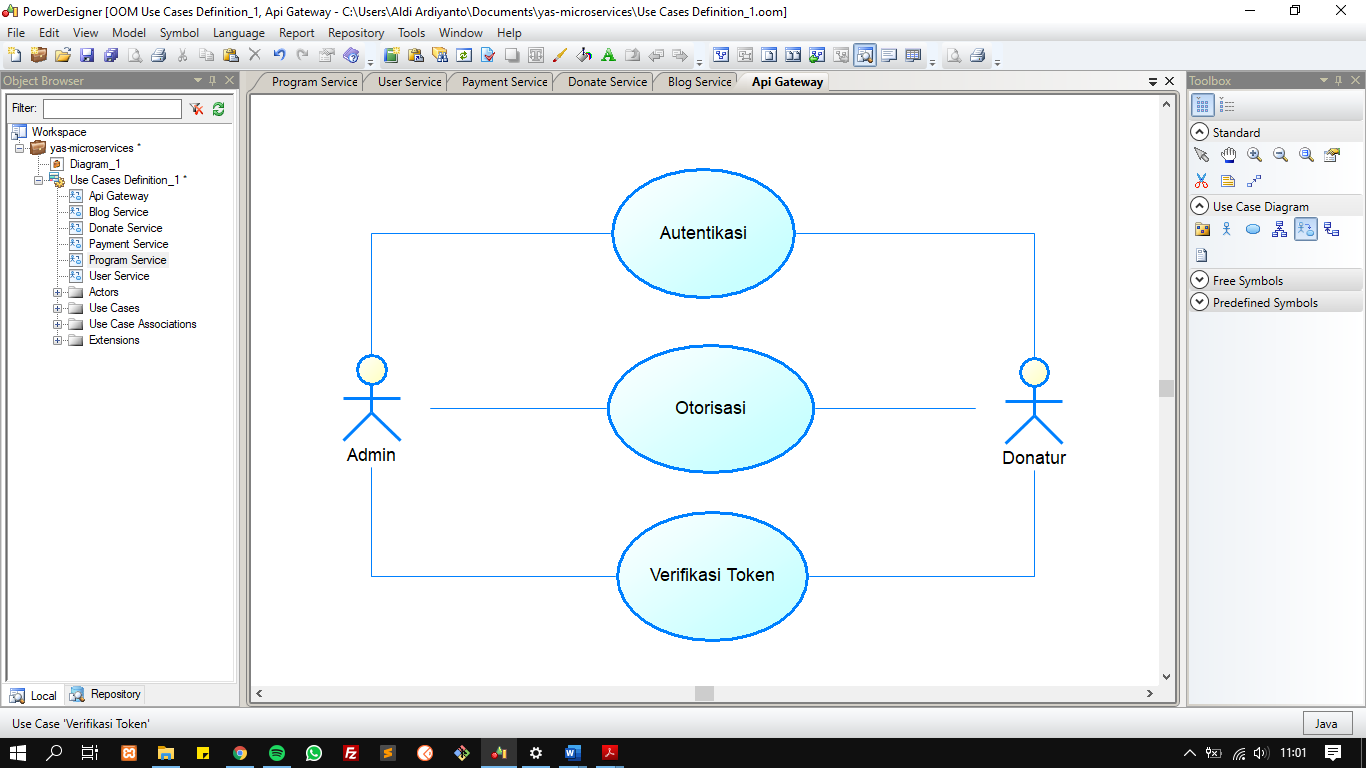
Use Case Diagram Program Service

1. Use Case Diagram Donation Service



Use Case Diagram Donation Service

1. Use Case Diagram Api Gateway



Use Case Diagram Api Gateway

### Use Case Scenario

#### Use Case Scenario Api Gateway

**Tabel** *Use Case Scenario Otentikasi*

Nama Use Case : Otentikasi

Fungsi : Melakukan otentikasi

Pre-Condition : Belum melakukan otentikasi

Post-Condition : Sudah melakukan otentikasi dan mendapatkan token

Aktor : Admin & Donatur

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
| 1. Melakukan permintaan autentikasi ke api gateway | 1. Meneruskan permintaan ke user service |
|  | 1. User service melakukan validasi |
|  | 1. User service memberikan respon sukses autentikasi ke api gateway |
|  | 1. Api gateway memberikan respon sukses autentikasi |
| Skenario Alternatif | |
| 1. Melakukan permintaan autentikasi ke api gateway | 1. Meneruskan permintaan ke user service |
|  | 1. User service melakukan validasi |
|  | 1. User service memberikan respon gagal autentikasi ke api gateway |
|  | 1. Api gateway memberikan respon gagal autentikasi |

**Tabel** *Use Case Scenario Otorisasi*

Nama Use Case : Otorisasi

Fungsi : Melakukan otorisasi terhadap service

Pre-Condition : Belum melakukan autentikasi

Post-Condition : Sudah melakukan autentikasi dan mendapatkan token

Aktor : Admin & Donatur

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
| 1. Melakukan permintaan otorisasi ke api gateway | 1. Api gateway meneruskan permintaan service tujuan |
|  | 1. Service tujuan melakukan validasi |
|  | 1. Service tujuan memberikan respon sukses otorisasi ke api gateway |
|  | 1. Api gateway memberikan respon sukses otorisasi |
| Skenario Alternatif | |
| 1. Melakukan permintaan otorisasi ke api gateway | 1. Api gateway meneruskan permintaan service tujuan |
|  | 1. Service tujuan melakukan validasi |
|  | 1. Service tujuan memberikan respon gagal otorisasi kea pi gateway |
|  | 1. Api gateway memberikan respon gagal otorisasi |

**Tabel** *Use Case Scenario Token*

Nama Use Case : Token

Fungsi : Melakukan permintaan token untuk otentikasi

Pre-Condition : Belum mempunyai token untuk otentikasi

Post-Condition : Sudah mempunyai token untuk otentikasi

Aktor : Admin & Donatur

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
| 1. Melakukan permintaan token ke api gateway | 1. Api gateway meneruskan permintaan ke user service |
|  | 1. User service melakukan validasi |
|  | 1. Service tujuan memberikan respon sukses ke api gateway |
|  | 1. Api gateway memberikan respon sukses dan token |
| Skenario Alternatif | |
| 1. Melakukan permintaan token ke api gateway | 1. Api gateway meneruskan permintaan ke user service |
|  | 1. User service melakukan validasi |
|  | 1. Service tujuan memberikan respon gagal ke api gateway |
|  | 1. Api gateway memberikan respon gagal |

#### Use Case Scenario Blog Service

**Tabel** *Use Case Kelola Blog*

Nama Use Case : Kelola Blog

Fungsi :

Pre-Condition :

Post-Condition :

Aktor : Admin

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  |  |
| Skenario Alternatif | |
|  |  |

**Tabel** *Use Case Kelola Blog*

Nama Use Case : Kelola Kategori Blog

Fungsi :

Pre-Condition :

Post-Condition :

Aktor : Admin

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  |  |
| Skenario Alternatif | |
|  |  |

**Tabel** *Use Case Lihat Blog*

Nama Use Case : Lihat Blog

Fungsi :

Pre-Condition :

Post-Condition :

Aktor : Donatur

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  |  |
| Skenario Alternatif | |
|  |  |

#### Use Case Scenario User Service

**Tabel** *Use Case Register*

Nama Use Case : Register

Fungsi :

Pre-Condition :

Post-Condition :

Aktor : Donatur

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  |  |
| Skenario Alternatif | |
|  |  |

**Tabel** *Use Case Login*

Nama Use Case : Login

Fungsi :

Pre-Condition :

Post-Condition :

Aktor : Admin dan Donatur

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  |  |
| Skenario Alternatif | |
|  |  |

**Tabel** *Use Case Kelola User*

Nama Use Case : Kelola User

Fungsi :

Pre-Condition :

Post-Condition :

Aktor : Admin

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  |  |
| Skenario Alternatif | |
|  |  |

#### Use Case Scenario Program Service

**Tabel** *Use Case Kelola Program*

Nama Use Case : Kelola Program

Fungsi :

Pre-Condition :

Post-Condition :

Aktor : Admin

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  |  |
| Skenario Alternatif | |
|  |  |

**Tabel** *Use Case Kelola Kategori Program*

Nama Use Case : Kelola Kategori Program

Fungsi :

Pre-Condition :

Post-Condition :

Aktor : Admin

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  |  |
| Skenario Alternatif | |
|  |  |

#### Use Case Scenario Donate Service

**Tabel** *Use Case Kelola Donasi*

Nama Use Case : Kelola Donasi

Fungsi :

Pre-Condition :

Post-Condition :

Aktor : Admin

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  |  |
| Skenario Alternatif | |
|  |  |

**Tabel** *Use Case Donasi*

Nama Use Case : Donasi

Fungsi :

Pre-Condition :

Post-Condition :

Aktor : Donatur

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  |  |
| Skenario Alternatif | |
|  |  |

## Aliran Kerja

### Activity *Diagram*

### Uraian Workflow

## Pemodelan Struktur Sistem dan Data

### *Class* Diagram

### *Class* Object *Description*

## Interaksi Antar Class

### Sequence Diagram

## Pemodelan Perilaku Sistem

### *State Diagram*

## Perancangan Endpoint Microservices

### *Api Gateway*

|  |  |  |  |
| --- | --- | --- | --- |
| **Fungsi** | **Method** | **URL** | **Parameter** |
| Menjembatani ke service blog dan menyediakan seluruh data blog | GET | api-gateway.com/blog | Token |
| Menjembatani ke service blog dan menyediakan data blog berdasarkan id | GET | api-gateway.com/blog/$id | Token, id |
| Menjembatani ke service blog dan mengirim data blog | POST | api-gateway/blog | Token, id, title, slug, description. photo |
| Menjembatani ke service blog dan menghapus data blog berdasarkan id | DELETE | api-gateway/blog/$id | Token, id |

### *Blog Service*

|  |  |  |  |
| --- | --- | --- | --- |
| **Fungsi** | **Method** | **URL** | **Parameter** |
| Menyediakan seluruh data blog | GET | service-blog.com/blog | Token |
| Menyediakan data blog berdasarkan blog\_id | GET | service-blog.com/blog/$id | Token, id |
| Mengirim data blog | POST | service-blog.com/blog | Token, title, slug, description. photo |
| Memperbaharui data blog | PUT | service-blog.com/blog/$id | Token, id |
| Menghapus data blog berdasarkan id | DELETE | service-blog.com /blog/$id | Token, id |
| Menyediakan semua data blog-category | GET | service-blog.com/blog-category | token |
| Menyediakan data blog\_category berdasarkan blog\_category\_id | GET | service-blog.com/blog-category/$ | Token, blog\_category\_id |
| Mengirim data blog\_category | POST | service-blog.com/blog-category | Token |
| Memperbaharui data blog\_category berdasarkan blog\_category\_id | PUT | service-blog.com/blog-category/$id | Token, blog\_category\_id |
| Menghapus data blog\_category berdasarkan blog\_category\_id | DELETE | service-blog.com/blog-category/$id | Token, blog\_category\_id |

### *User Service*

|  |  |  |  |
| --- | --- | --- | --- |
| **Fungsi** | **Method** | **URL** | **Parameter** |
| Mengirimkan data user | POST | user-service.com/register | name, email, password, role |
| Mengirimkan data user | POST | user-service.com/login | email, password |
| Menyediakan data user | GET | user-service.com/user | Token, id |
| Menyediakan data user berdasarkan id | GET | user-service.com/user/$id | Token, id |
| Memperbaharui data user berdasarkan id | PUT | user-service.com/user/$id | Token, id |
| Menghapus data user berdasarkan id | DELETE | user-service.com/user/$id | Token, id |

### *Program Service*

|  |  |  |  |
| --- | --- | --- | --- |
| **Fungsi** | **Method** | **URL** | **Parameter** |
| Menyediakan semua data program | GET | program-service.com/program | Token |
| Menyediakan data program berdasarkan program\_id | GET | program-service.com/program/$id | Token, id |
| Mengirimkan data program | POST | program-service.com/program | Token |
| Memperbaharui data program berdasarkan program\_id | PUT | program-service.com/program/$id | Token, program\_id |
| Menghapus data program berdasarkan program\_id | DELETE | program-service.com/program | Token, program\_id |
| Menyediakan semua data program\_category | GET | program-service.com/program-category | Token |
| Menyediakan data program\_category berdasarkan program\_category\_id | GET | program-service.com/program-category/$id | Token, program\_category\_id |
| Mengirimkan data program\_category | POST | program-service.com/program-category | Token |
| Memperbaharui data program\_category berdasarkan program\_category\_id | PUT | program-service.com/program-category/$id | Token, program\_category\_id |
| Menghapus data program\_category | DELETE | program-service.com/program-category/$id | Token, program\_category\_id |

### Donation Service

|  |  |  |  |
| --- | --- | --- | --- |
| **Fungsi** | **Method** | **URL** | **Parameter** |
| Menyediakan semua data donasi | GET | Donation-service.com/donation | Token |
| Menyediakan data donasi berdasarkan donation\_id | GET | Donation-service.com/donation/$id | Token, donation\_id, program\_id |
| Menghapus data donasi berdasarkan donation\_id | DELETE | Donation-service.com/donation/$id | Token, donation\_id |
| Menyediakan semua data pembayaran | GET | Donation-service/payment | Token |
| Menyediakan data pembayaran berdasarkan payment\_id | GET | Donation-service/payment/$id | Token, payment\_id |
| Mengirimkan data pembayaran | POST | Donation-service/payment | Token, payment\_id, program\_id |

## Pseudecode

### Proses *Spesification*

# BAB IV IMPLEMENTASI DAN PENGUJIAN

## Implementasi

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

### Lingkup dan Batasan

Lingkup dan batasan dari perangkat lunak yang telah penulis buat adalah sebagai berikut :

1. Perangkat lunak dibuat dengan menggunakan arsitektur *microservices*.

### Kebutuhan Sumber Daya

Untuk menggunakan perangkat lunak ini dibutuhkan spesifikasi yang mampu mendukung pengopersiannya, beberapa komponen yang dibutuhkan adalah sebagai berikut :

1. Kebutuhan Minimum Hardware

Untuk menjalankan perangkat lunak yang telah dibuat, dibutuhkan beberapa spesifikasi kebutuhan hardware sebagai berikut :

1. Processor : Quadcore 3,0 GHz
2. RAM : 8 GB
3. SSD : 256 GB
4. VGA : 2 GB
5. Monitor : 22’ inch FHD
6. Keyboard : Wireless Keyboard
7. Mouse : Wireless Mouse
8. Kebutuhan Minimum Software

Perangkat Lunak yang telah dibuat juga membutuhkan software yang mendukung dalam proses pembuatan maupun dalam penggunaan perangkat lunak tersebut.

Kebutuhan akan software atau perangkat lunak untuk mengembangkan aplikasi ini adalah sebagai berikut :

1. Sistem operasi : Windows 10 32/64 Bit
2. Web browser : Google Chrome
3. Pemodelan : Power Designer
4. Apache :
5. Composer :
6. Laravel :
7. Node Package Manager :
8. Nodejs :
9. Expressjs :
10. DBMS : MySQL
11. Tools Management MySQL : phpMyAdmin
12. Text Editor : Sublime Text 3
13. Command Line : Git Bash
14. REST Tester : Postman
15. Spesifikasi Brainware

Spesifikasi kebutuhan brainware untuk mengembangkan aplikasi ini yaitu :

1. Memahami dasar arsitektur *microservices*
2. Memahami teknologi di sisi server

### Imlementasi Aplikasi

## Pengujian

### Lingkup dan Lingkungan

### Kebutuhan Sumber Daya

### Hasil Pengujian

# BAB V KESIMPULAN DAN SARAN

## Kesimpulan

Dari penelitian dan implementasi arsitektur microservices maka dapat ditarik kesimpulan sebagai berikut:

1. Arsitektur microservices dapat mengimplementasikan berbagai teknologi di dalamnya.

## Saran

Setelah melakukan implementasi, penulis memberikan saran yang ditujukan untuk tahap pengembangan ke depannya. Berikut poin-poin saran yang disampaikan:

1. Penggunaan autodiscovery pada service untuk mengantisipasi terjadinya service tidak bisa diakses.
2. Penerapan message broker sebagai komunikasi antar service.

# DAFTAR PUSTAKA

[1] S. Newman, *Building Microservices @ Squarespace*. O’Reilly Media, 2015.

# LAMPIRAN