TUGAS AKHIR – KI1502

IMPLEMENTASI APLIKASI MANAJEMEN API BERBASIS PROTOKOL REST MENGGUNAKAN PLATFORM WSO2

RARAS ANGGITA
NRP 05111440000046

Dosen Pembimbing
Rizky Januar Akbar, S.Kom., M.Eng.
Wijayanti Nurul Khotimah, S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI1502

IMPLEMENTASI APLIKASI MANAJEMEN API BERBASIS PROTOKOL REST MENGGUNAKAN PLATFORM WSO2

RARAS ANGGITA
NRP 05111440000046

Dosen Pembimbing
Rizky Januar Akbar, S.Kom., M.Eng.
Wijayanti Nurul Khotimah, S.Kom., M.Sc.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESIS – KI1502

IMPLEMENTATION OF API MANAGEMENT APPLICATION BASED ON REST PROTOCOL USING WSO2 PLATFORM

RARAS ANGGITA
NRP 05111440000046

Supervisors
Rizky Januar Akbar, S.Kom., M.Eng.
Wijayanti Nurul Khotimah, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

IMPLEMENTASI APLIKASI MANAJEMEN API BERBASIS PROTOKOL REST MENGGUNAKAN PLATFORM WSO2

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh
RARAS ANGGITA
NRP : 05111440000046

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Rizky Januar Akbar, S.Kom., M.Eng. .....
   NIP: 19870103 201404 1 001 (Pembimbing 1)

2. Wijayanti Nurul K., S.Kom., M.Sc. .....
   NIP: 19860312 201212 2 004 (Pembimbing 2)

SURABAYA
JULI 2018

[Halaman ini sengaja dikosongkan]

IMPLEMENTASI APLIKASI MANAJEMEN API BERBASIS PROTOKOL REST MENGGUNAKAN PLATFORM WSO2

Nama Mahasiswa	**:**	**Raras Anggita**
NRP	**:**	**05111440000046**
Jurusan	**:**	**Departemen Informatika FTIK-ITS**
Dosen Pembimbing 1	**:**	**Rizky Januar Akbar, S.Kom., M.Eng.**
Dosen Pembimbing 2	**:**	**Wijayanti Nurul Khotimah, S.Kom., M.Sc.**

ABSTRAK

Institut Teknologi Sepuluh Nopember Surabaya tentunya mempunyai berbagai sistem yang menunjang jalannya aspek pendidikan, manajemen, dan aspek lainnya. Antar satu sistem dengan sistem yang lain pasti membutuhkan pertukaran data. Namun, sistem-sistem di ITS saat ini masih menggunakan sistem pertukaran data terdistribusi. Pengggunaan sistem terdistribusi ini lebih menyulitkan dalam pertukaran data.

Untuk mengatasi hal tersebut, pada tugas akhir ini dibuat aplikasi manajemen API dengan menggunakan platform WSO2. Manajemen API ini merupakan solusi untuk mendesain, menerbitkan API, dan mengelola semua aspek API serta untuk mengatur sekuriti dan juga analisis.

Dalam pembuatan API dengan menggunakan web service REST diperlukan suatu standar dalam penamaan resource. Hal ini untuk memudahkan pengelola kini dan yang akan datang sehingga tidak memunculkan kerancuan. Selain itu, untuk memberikan proses sekuriti yang sama antar sistemnya, maka dibangun sebuah sistem sekuriti yang mengatur proses otentikasi, otorisasi, dan juga pembatasan akses berdasarkan IP. Untuk mengkustomisasi halaman antarmuka website WSO2 API Store, digunakan kerangka kerja Jaggery dalam pengembangan temanya.

Pegujian pada sistem keamanan yang dilakukan telah berhasil menampilkan status respon yang sesuai, untuk otentikasi, otorisasi, dan juga untuk pembatasan IP. Pengujian terhadap aplikasi manajemen API yang dilakukan pada sistem e-kelas telah berhasil dilakukan. Evaluasi terhadap halaman antarmuka pengguna awal menghasilkan nilai sebesar 6,28 dan antarmuka hasil kustomisasi menghasilkan nilai yang lebih tinggi, yaitu 8,34.

Kata kunci: Manajemen API, WSO2 API Manajemen, Otentikasi, Otorisasi

IMPLEMENTATION OF API MANAGEMENT APPLICATION BASED ON REST PROTOCOL USING WSO2 PLATFORM

Name	**: Raras Anggita**
NRP	**: 05111440000046**
Major	**: Informatics Department FTIK-ITS**
Supervisor I	**: Rizky Januar Akbar, S.Kom., M.Eng.**
Supervisor II	**: Wijayanti Nurul Khotimah, S.Kom., M.Sc.**

ABSTRACT

Institut Teknologi Sepuluh Nopember Surabaya (ITS) has several systems to support the aspect of education, management, and other aspects that be held there. Between one system and another system requires data exchange. Currently, the data exchange between the system still use distributed data system. The use of this system gives a lot of disadvantages in exchange data.

To overcome this, in this thesis will be handled by using WSO2 API Management platform.This API management is a solution to create, publish, and manage all aspects of the API, also handling security and analysis the usages of the API.

In API's development using REST based web service it is required a resource naming standard. It is necessary for the current and future developer so as not to create confusion. In addition, to provide the same security process between systems, then built a security system that regulates the authentication process, authorization, and also access restrictions based on IP. To customize the WSO2 API Store website interface, customization is uses Jaggery framework in its theme development.

The tests on the security system performed have successfully displayed the appropriate response status, for authentication, authorization, and also for IP restrictions. The test on management API application in e-kelas system is successfull. Evaluation of the initial user interface yields a final value of 6.28 and the customized interface yields a higher value of 8.34.

Keyword: ***Management API, WSO2 API Management, Authentication, Authorization***

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan yang Maha Esa, karena atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul

IMPLEMENTASI APLIKASI MANAJEMEN API BERBASIS PROTOKOL REST MENGGUNAKAN PLATFORM WSO2

Melalui lembar ini, penulis ingin menyampaikan ucapan terimakasih dan penghormatan yang sebesar-besarnya kepada:

1. Tuhan yang Maha Kuasa atas segala kasih, karunia, dan rahmat-Nya yang telah diberikan selama ini.
2. Ayah, Ibu, dan ketiga kakak penulis yang tiada henti-hentinya mencurahkan kasih sayang, perhatian dan doa kepada penulis selama ini.
3. Bapak Rizky Januar Akbar, S.Kom., M.Eng. selaku dosen pembimbing I yang selalu memberikan motivasi dan membimbing penulis selama pengerjaan tugas akhir.
4. Ibu Wijayanti Nurul Khotimah, S.Kom., M.Sc. selaku dosen pembimbing II yang telah memberikan nasihat, arahan, dan bantuan sehingga penulis dapat menyelesaikan tugas akhir ini.
5. Bapak Dr.Eng Darlis Herumurti, S.Kom.,M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, Bapak Dr. Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator TA, dan segenap dosen Teknik Informatika yang telah banyak memberikan ilmu kepada penulis.
6. Bebet, Nafia, Mila, Rani yang telah membantu dan menjadi tempat keluh kesah, penghibur, dan pemberi semangat selama ini.
7. Teman-teman seperjuangan tugas akhir di Laboratorium RPL, Sabila, Winda, Nurul, Rara, Elva,

Nia, Steven, William, Farhan, Valdy, Faishal, Aldo, Dewangga, Fito yang telah membantu, menemani, dan menghibur selama pengerjaan tugas akhir ini.

8. Mbak Fitria yang telah banyak membantu dalam proses pengerjaan serta memberikan masukan kepada penulis.
9. Teman-teman angkatan 2014 yang telah memberikan semangat dan motivasi, serta berbagi ilmu selama penulis berkuliah di Informatika ITS.
10. Serta pihak lain yang namanya tidak dapat penulis sebutkan satu-persatu.

Penulis menyadari sepenuhnya bahwa tugas akhir ini masih memiliki kekurangan. Oleh karena itu, dengan tangan terbuka, penulis menerima segala saran dan kritik dari pembaca untuk perbaikan ke depannya.

Surabaya, Juni 2018

Raras Anggita

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SEMU

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

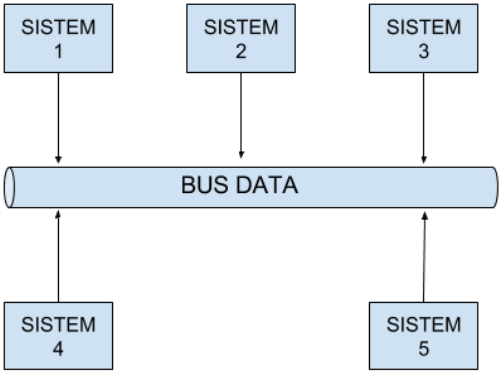
Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1 Latar Belakang

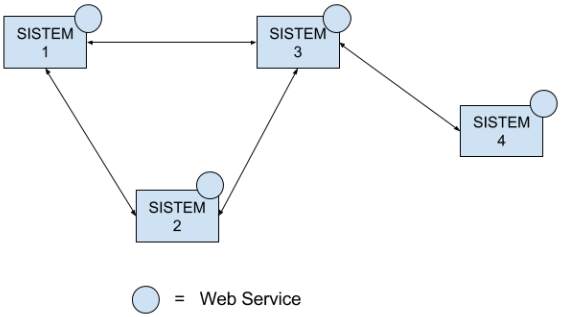
Institut Teknologi Sepuluh Nopember Surabaya merupakan perguruan tinggi di bidang sains dan teknologi dan juga salah satu perguruan tinggi terbesar di Indonesia. Untuk menunjang jalannya aspek pendidikan, manajemen, dan aspek lainnya, tentunya ITS mempunyai berbagai sistem yang mendukung. Sistem-sistem tersebut, antara lain e-learning (share.its.ac.id) untuk sarana pembelajaran daring terbuka dan terpadu, e-office (esurat.its.ac.id) merupakan sistem informasi e-surat di ITS, dan *archive* (arsip.its.ac.id) yang menyediakan informasi mengenai UPT Kearsipan ITS. Selain itu, ITS juga menyediakan webmail ITS (webmail.its.ac.id), *licensed software* yang menyediakan beberapa *software* yang berlisensi resmi, dan *lecturer profile* (personal.its.ac.id) yang menyediakan berbagai informasi terkait dosen-dosen ITS.

Antar satu sistem dengan sistem yang lain pasti membutuhkan pertukaran data. Terdapat 2 tipe pertukaran data, yaitu terpusat dan federasi (terdistribusi). Pada pertukaran data terpusat, sebuah server yang menyediakan informasi di dalam basis data hanya terdiri dari satu database saja, sebagai pusat dan juga sebagai komputer induk bagi seluruh sistem dan juga jaringan yang ada. Hal ini dapat di lihat pada Gambar 1.1. Sedangkan sistem-sistem di ITS masih menggunakan sistem terdistribusi. Hal ini lebih menyulitkan dalam pertukaran data. Apabila suatu sistem akan meminta data dari sistem lain maka sistem peminta akan langsung berhubungan dengan sistem pemberi, seperti yang terlihat pada Gambar 1.2. Selain itu, setiap sistem pada pertukaran data terdistribusi mempunyai *provider service*

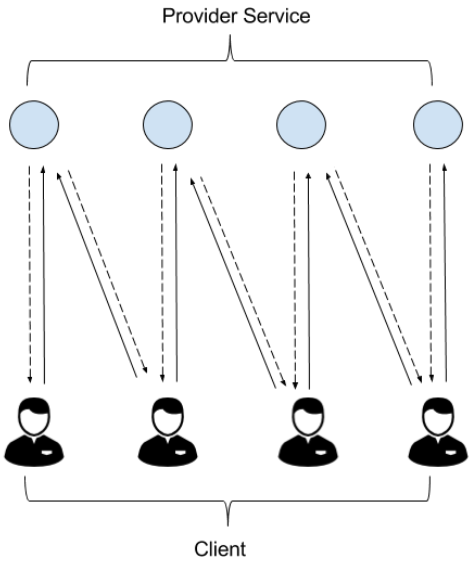
sendiri. Pada Gambar 1.3 dijelaskan apabila klien akan mengakses salah satu sistem tersebut, maka klien akan secara langsung mengakses ke *provider service* sistem tersebut. Setiap sistem juga mempunyai sekuriti (otentikasi dan otorisasi) dan standar penamaan *resource* yang berbeda antara satu sistem dan sistem lain.



Gambar 1.1 Pertukaran Data Terpusat



Gambar 1.2 Pertukaran Data Terdistribusi



Gambar 1.3 Alur kerja sistem di ITS

Untuk mengatasi masalah tersebut, diusulkan sebuah solusi berupa manajemen API dengan menggunakan platform WSO2. Penggunaan manajemen API akan membuat layanan lebih efektif, karena lebih mudah untuk dilakukan manajemen dan pengaturannya. Sehingga nantinya setiap sistem yang dimiliki ITS mempunyai sistem sekuriti (otentikasi dan otorisasi) yang sama, serta standar penamaan *resource*, dan *versioning* yang sama.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana standar penamaan *resource* pada REST *service*?
2. Bagaimana mekanisme implementasi otentikasi OAuth2?

3. Bagaimana mekanisme implementasi otorisasi menggunakan Json Web Token (JWT)?
4. Bagaimana mekanisme implementasi pembatasan akses berdasarkan IP?
5. Bagaimana cara kustomisasi *user interface* manajemen API berplatform WSO2?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, yaitu:

1. Sistem ini khusus untuk ruang lingkup ITS.
2. Platform yang digunakan pada tugas akhir ini adalah WSO2 dengan versi 2.0.0.
3. Sistem keamanan dan *user store* dibangun dengan menggunakan bahasa Java dan Eclipse IDE.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah:

1. Mengetahui standar penamaan *resource* pada REST service.
2. Mengetahui mekanisme implementasi otentikasi OAuth2.
3. Mengetahui mekanisme implementasi otorisasi menggunakan Json Web Token (JWT).
4. Mengetahui mekanisme implementasi pembatasan akses berdasarkan IP.
5. Mengetahui cara kustomisasi *user interface* manajemen API berplatform WSO2.

1.5 Manfaat

Manfaat yang diharapkan dengan adanya tugas akhir ini adalah agar sistem-sistem yang ada lebih terstruktur. Selain itu, dengan adanya manajemen API lebih memudahkan pengaturan sekuriti, versioning, dan juga dokumentasi terhadap sistem-sistem di ITS karena hanya mengaturnya sekali.

1.6 Metodologi Pembuatan Tugas Akhir

Adapun beberapa tahap dalam proses pengerjaan tugas akhir ini, yaitu sebagai berikut:

1. Penyusunan proposal tugas akhir

Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat dari hasil pembuatan tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Sub bab metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

2. Studi literatur

Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan aplikasi yaitu mengenai REST *web service*, manajemen API, OAuth2 yang akan digunakan untuk otentikasi, JWT yang akan digunakan untuk otorisasi, dan juga kustomisasi halaman antar muka WSO2 dengan menggunakan Jaggery.

3. Analisis dan desain perangkat lunak

Tahap ini meliputi perumusan membahas tentang analisis permasalahan, deskripsi umum sistem, perancangan WSO2 API manajemen, dan perancangan keamanan yang termasuk proses otentikasi, otorisasi, dan pembatasan akses dengan IP.

4. Implementasi

Aplikasi ini diimplementasikan dengan menggunakan kakas bantu:

1. Bahasa pemrograman yang digunakan adalah Java.

2. Postman, kakas bantu untuk menguji integrasi *web service* dengan sistem yang sudah dibangun.
3. Eclipse sebagai *text editor* dalam pengerjaan *web service*.
4. Sublime sebagai *text editor* dalam pegerjaan kustomisasi halaman antarmuka.

5. Pengujian dan evaluasi

Pengujian dan evaluasi dari sistem ini akan dilakukan dengan melakukan *request* yang dilakukan oleh beberapa partisipan. Selain itu akan disebarkan kuisioner kepada 10 partisipan untuk mengetahui ketertarikan partisipan terhadap halaman antarmuka yang baru.

6. Penyusunan buku tugas akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini. Pada tahap ini juga disertakan hasil dari implementasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir ini secara garis besar antara lain:

1. Pendahuluan
   a. Latar Belakang
   b. Rumusan Masalah
   c. Batasan Masalah
   d. Tujuan
   e. Manfaat
   f. Metodologi Pembuatan Tugas Akhir
   g. Sistematika Penulisan
2. Dasar Teori
3. Analisis dan Perancangan Sistem
4. Implementasi
5. Pengujian dan Evaluasi
6. Kesimpulan dan Saran
7. Daftar Pustaka

1.7 Sistematika Penulisan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

Bab II Dasar Teori

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini akan membahas tentang analisis permasalahan, deskripsi umum sistem, perancangan WSO2 AM, dan perancangan keamanan yang termasuk proses otentikasi, otorisasi, dan pembatasan akses dengan IP.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan dan implementasi fitur-fitur penunjang aplikasi termasuk implementasi sistem dan implementasi antarmuka pengguna.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian dan evaluasi dari sistem ini akan dilakukan dengan melakukan *request* yang dilakukan oleh beberapa partisipan. Selain itu akan disebarkan kuisioner kepada 10 partisipan untuk mengetahui ketertarikan partisipan terhadap halaman antarmuka yang baru.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan pada tugas akhir ini. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar kode yang ada pada aplikasi ini dan juga hasil dari kuisioner pada pengujian.

BAB II
DASAR TEORI

2.1 REST *Web Service*

REST (*REpresentational State Transfer*) merupakan standar arsitektur komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan berbasis web. Umumnya menggunakan HTTP (*Hypertext Transfer Protocol*) sebagai protocol untuk komunikasi data. REST pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000. Pada arsitektur REST, REST server menyediakan resources (sumber daya/data) dan REST client mengakses dan menampilkan resource tersebut untuk penggunaan selanjutnya. Setiap resource diidentifikasi oleh URIs (*Universal Resource Identifiers*) atau global ID. Resource tersebut direpresentasikan dalam bentuk format teks, JSON atau XML. Pada umumnya formatnya menggunakan JSON dan XML [1].

Penggunaan metode-metode HTTP dalam REST adalah sebagai berikut [1]:

Tabel 2.1 Metode HTTP dan Penggunaannya dalam REST

Metode	Deskripsi
GET	Mendapatkan (*read*) sebuah sumber daya (*resource*) yang diidentifikasi dengan URI (*Uniform Resource Identifier*)
POST	Mengirimkan sumber daya (*resource*) ke server. Digunakan untuk membuat (*create*) sumber daya baru
PUT	Mengirimkan sumber daya (*resource*) ke server. Digunakan untuk memasukkan (*insert*) atau memperbarui (*update*) sumber daya yang tersimpan.
DELETE	Menghapus (*delete*) sumber daya (*resource*) yang diidentifikasi dengan URI

Metode	Deskripsi
HEAD	Mendapatkan metadata (*response header*) dari sumber daya (*resource*) yang diidentifikasi dengan URI.

Komponen dari HTTP Response adalah:

- Status/Response Code, mengindikasikan status *server* terhadap *resource* yang diminta. Misal: 404, artinya *resource* tidak ditemukan dan 200 response OK.
- HTTP Version, menunjukkan versi dari HTTP yang digunakan, contoh HTTP v1.1.
- Response Header, berisi metadata untuk HTTP Response. Contoh, type server, panjang content, tipe content, waktu response, dll.
- Response Body, konten dari data yang diberikan.

2.2 Manajemen API

Manajemen API adalah proses penerbitan, pendokumentasian dan pengawasan antarmuka pemrograman aplikasi (API) di lingkungan yang aman dan terukur. Tujuan manajemen API adalah mengizinkan organisasi yang menerbitkan API untuk memantau siklus hidup antarmuka dan memastikan kebutuhan pengembang dan aplikasi yang menggunakan API terpenuhi [2].

Kebutuhan manajemen API mungkin berbeda dari organisasi ke organisasi, tetapi manajemen API itu sendiri mencakup beberapa fungsi dasar, termasuk keamanan, *monitoring*, dan kontrol versi. Agar dapat digunakan dengan benar, API memerlukan dokumentasi yang kuat, peningkatan tingkat keamanan, pengujian komprehensif, dan keandalan yang tinggi.

2.3 WSO2 API Manager

WSO2 API Manager adalah solusi lengkap untuk membuat, menerbitkan (*publish*), dan mengelola semua aspek API dan siklus

hidupnya [3]. WSO2 AM terdiri dari beberapa komponen utama, yaitu sebagai berikut:

1. **API Publisher**
   Menyediakan antarmuka pengguna *web* kolaboratif pengguna akhir untuk penyedia API untuk menerbitkan API, berbagi dokumentasi, menyediakan kunci API, dan mengumpulkan umpan balik tentang fitur, kualitas, dan penggunaan API. API Publisher didukung oleh Jaggery, WSO2 *Governance Registry* dan produk WSO2 *Identity Server*. API *Publisher* dapat diakses melalui URL https://localhost:9443/publisher.
2. **API *Store***
   Menyediakan antarmuka pengguna kolaboratif bagi pengguna akhir (*enduser*) untuk mendaftarkan diri, menemukan fungsionalitas API, berlangganan API, dan berinteraksi dengan API Publisher. API Store didukung oleh Jaggery, WSO2 *Governance Registry* dan produk WSO2 *Identity Server*. API *Store* dapat diakses melalui URL https://localhost:9443/store.
3. **API *Gateway***
   API *Gateway* mengamankan, melindungi, mengelola, dan menskalakan panggilan API. API *Gateway* adalah proksi API sederhana yang memotong permintaan API dan menerapkan kebijakan seperti pemeriksaan keamanan. API *Gateway* menangani validasi sekuriti. Setelah validasi, maka akan melewati panggilan layanan web ke *back-end* yang sebenarnya. Jika panggilan layanan adalah panggilan permintaan token, API Gateway akan meneruskannya langsung ke *server* API *Key Manager* untuk menanganinya. API *Gateway* sendiri dapat diakses melalui URL https://localhost:9443/carbon.
4. **API *Key Manager***
   API *Key Manager* menangani semua keamanan. Ketika API *Gateway* menerima panggilan API, maka API *Gateway* akan menghubungi layanan API *Key Manager* untuk memverifikasi validitas token dan melakukan pemeriksaan keamanan. Ketika

API *Gateway* menerima panggilan untuk login, itu langsung meneruskan panggilan ke *server Key Manager*.

2.4 OAuth2

OAuth2 adalah kepanjangan dari *open authorization* dimana OAuth2 banyak digunakan dikalangan developer sebagai proses authorization sebuah aplikasi. Dengan menggunakan protokol ini, maka aplikasi pihak ketiga dapat mengakses data dari aplikasi yang telah dibangun [5]. Di dalam OAuth2 terdapat beberapa grant type yang memungkinkan pengguna untuk menerima *access token* dalam beberapa cara, antara lain:

1. **Grant Type Authorization Code**
   Grant type ini menggunakan *generate code* yang berasal dari authorization server untuk ditukarkan dengan sebuah token, dimana token ini digunakan untuk mengakses sebuah resource/API [4].
2. **Grant Type Resource Owner Password Credentials**
   Pada *grant type* ini, menggunakan *username* dan *password* langsung dari *owner* sebagai bagian dari *request*. *Grant type* ini biasanya digunakan jika aplikasi yang dibangun merupakan aplikasi pribadi atau pembuat aplikasi berasal dari perusahaan yang sama.
3. **Grant Type Client Credentials**
   Grant type ini biasanya digunakan oleh aplikasi-aplikasi yang telah menjalin kerja sama dengan suatu perusahaan. Masing-masing aplikasi pihak ketiga akan mendapatkan *client id* dan *client secret*, dengan menggunakan *client id* dan *client secret*, dan menukarnya dengan *access token*.
4. **Grant Type Implicit Client**
   Grant type ini biasanya digunakan untuk *client* yang tidak dapat menyimpan *client secret* dengan aman, contohnya seperti Angular JS, Vue JS, React JS dan lain sebagainya. Pada grant type ini, biasanya *authorization server* tidak akan memberikan *refresh token* dikarenakan penyimpanan *refresh token* yang tidak aman.

5. **Grant Type JWT Bearer**
   Dengan *grant type* ini pengguna dapat mengajukan JWT (JSON Web Token) dalam *request* ke token *endpoint*. *Access* token (tanpa *refresh token*) akan diberikan secara langsung sebagai balasan.

2.5 Json *Web* Token (JWT)

JSON *Web* Token (JWT) adalah standar terbuka (RFC 7519) yang mendefinisikan cara yang kompak dan mandiri untuk mentransmisikan informasi antar pihak sebagai objek JSON dengan aman. Informasi ini bisa diverifikasi dan dipercaya karena sudah ditandatangani secara digital. JWT dapat ditandatangani dengan menggunakan rahasia (dengan algoritma HMAC) atau pasangan kunci publik / swasta menggunakan RSA [5].

JWT terdiri dari tiga bagian yang dipisahkan oleh titik (.), yaitu

1. **Header**
   Header biasanya terdiri dari dua bagian: jenis token, yaitu JWT, dan algoritma *hashing* yang digunakan, seperti HMAC SHA256 atau RSA.
2. **Payload**
   Payload berisi klaim. Klaim adalah pernyataan tentang entitas (biasanya, pengguna) dan metadata tambahan. Terdapat tiga jenis klaim, yaitu
   a. ***Reserved claims***
      Ini adalah serangkaian klaim yang telah ditetapkan sebelumnya yang tidak wajib namun disarankan, untuk menyediakan satu set klaim interoperabel yang berguna. Beberapa diantaranya adalah: iss (*issuer*), exp (*expiration time*), sub (subjek), aud (*audience*), dan lain-lain.
   b. ***Public claim*s**
      Ini dapat didefinisikan sesuka hati oleh mereka yang menggunakan JWT. Tapi untuk menghindari tabrakan mereka harus didefinisikan di IANA JSON Web Token

Registry atau didefinisikan sebagai URI yang berisi namespace tabrakan.

c. ***Private claims***

Ini adalah klaim khusus yang dibuat untuk berbagi informasi antar pihak yang setuju untuk menggunakannya.

3. ***Signature***

Signature digunakan untuk memverifikasi bahwa pengirim JWT adalah yang dikatakannya dan untuk memastikan bahwa pesan tersebut tidak berubah sepanjang jalan.

2.6 Apache Maven

Maven merupakan alat bantu pengeloalaan dependensi proyek pada Java.

2.6.1 POM

POM merupakan singkatan dari "Project Object Model". POM merupakan representasi XML dari proyek Maven yang dibuat dalam file bernama pom.xml. POM berisi semua informasi yang diperlukan tentang proyek, serta konfigurasi dependensi dan plugin yang akan digunakan selama proses pengembangan [6]. Penambahan dependensi dapat dilakukan dengan menambah `<dependency></dependency>` untuk tiap pustaka dependensi ke dalam `<dependencies></dependencies>`. Dependensi-dependensi yang diperlukan dapat ditemukan pada web Maven Repository.

2.6.2 Dependensi Maven

Dependensi merupakan pustaka Java yang berbentuk berkas .jar yang dapat biasanya diperlukan pengembang dalam mengembangkan perangkat lunak.

2.6.2.1 Dependensi Transitif

Dependensi transitif merupakan fitur baru pada Maven 2.0. Fitur ini memungkinkan pengembang untu tidak menemukan dan

menentukan *library* spesifik yang dibutuhkan oleh dependensi proyek akan disertakan secara otomatis.

1. Dependensi *Mediation*
   Dependensi ini menentukan versi dependensi apa yang akan digunakan ketika beberapa versi artefak ditemui. Saat ini, Maven 2.0 hanya mendukung menggunakan "definisi terdekat" yang berarti bahwa akan menggunakan versi ketergantungan terdekat ke proyek dalam pohon dependensi.
2. Dependensi *Management*
   Dependensi ini memungkinkan pengembang untuk secara langsung menentukan versi artefak yang akan digunakan ketika mereka ditemui dalam dependensi transitif atau dalam dependensi di mana tidak ada versi yang telah ditentukan.
3. Dependensi *Scope*
   Dependensi ini memungkinkan untuk hanya menyertakan dependensi yang sesuai untuk tahap pembuatan saat ini.
4. Dependensi E*xcluded*
   Pengembang mendefinisikan dependensi apa saja yang tidak diperlukan oleh dependensi di dalam proyek.
5. Dependensi *Optional*
   Merupakan dependensi yang secara *default* tidak disertakan.

2.6.2.2 Cakupan Dependensi (*Dependency Scope*)

Fitur ini berguna untuk mengatur transitifitas dari sebuah dependensi. Terdapat 6 *scope* yang tersedia [7], yaitu

1. *Compile*
   Scope default pada maven jika elemen *scope* tidak didefinisikan dalam sebuah dependensi. Dependensi dengan *scope* ini tersedia di dalam *classpath* proyek. *Scope* ini secara *default* menyertakan dependensi transitif.
2. *Provided*

Scope ini mirip dengan *scope compile* namun tidak menyertakan dependensi ke dalam *classpath* proyek.

3. *Runtime*
   Scope ini menandakan bahwa dependensi diperlukan saat proses eksekusi aplikasi bukan saat proses kompilasi.
4. *Test*
   Scope ini menandakan bahwa dependensi diperlukan saat proses *testing* aplikasi bukan saat proses kompilasi atau eksekusi.
5. *System*
   Scope ini mirip dengan *scope provided* namun pengembang diharuskan menyediakan pustaka .jar secara eksplisit.
6. *Import*
   Merupakan *scope* yang digunakan untuk menyertakan dependensi yang bertipe pom. Dependensi ini hanya bisa digunakan didalam elemen <*dependencyManagement*>.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Pada Bab 3 ini akan dijelaskan mengenai analisis dan perancangan perangkat lunak untuk mencapai tujuan dari tugas akhir. Perancangan ini meliputi analisis permasalahan, deskripsi umum sistem, perancangan WSO2 AM, perancangan keamanan, dan perancangan *user store*.

3.1 Analisis Metode Secara Umum

Tahap analisis dibagi menjadi beberapa bagian, antara lain analisis permasalahan, deskripsi umum sistem, dan spesifikasi kebutuhan perangkat lunak.

3.1.1 Analisis Permasalahan

Sebagai salah satu perguruan tinggi ternama, tentunya ITS mempunyai banyak sistem untuk menunjang jalannya berbagai aktivitas. Dengan banyaknya sistem yang dimiliki ITS diperlukan suatu aplikasi yang berguna untuk memanajemen API, mulai dari proses merancang, menerbitkan (*publish*), mendokumentasikan, dan siklus hidupnya API dari tiap sistem tersebut.

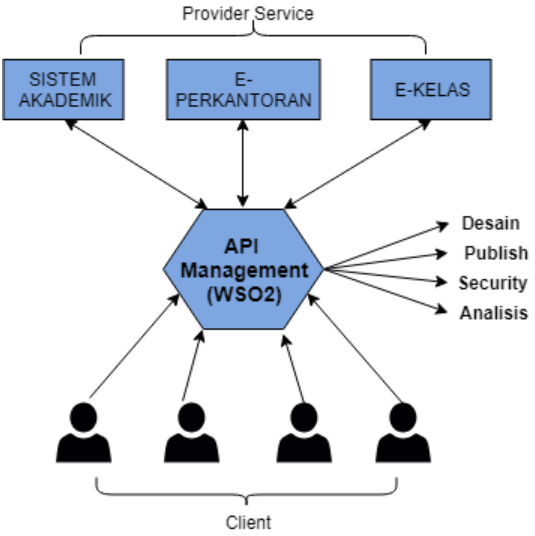
Mengingat dengan banyaknya sistem tersebut, dalam pembuatan API itu sendiri diperlukan suatu standar dalam penulisan *resource*. Namun, pada protokol REST tidak terdapat standar baku pada sistem penamaan *resource*. Padahal standarisasi ini sangat diperlukan. Standarisasi ini juga berfungsi agar sistem-sistem di ITS mempunyai standar penamaan *resource* yang baku dan sama antar satu sistem dengan sistem lainya. Penulisan resource yang konsisten juga akan membantu pengembang, baik pengembang sekarang maupun pengembang yang akan datang. Standarisasi ini meliputi *filtering*, *sorting*, *searching*, *paging*, dan juga *versioning*.

Pada sistem ini juga diperlukan sebuah standar sistem keamanan yang mengatur jalannya proses otentikasi apakah username dan password sudah cocok dengan basis data, otorisasi apakah pengguna yang sudah diidentifikasi (diotentikasi) tersebut

mempunyai kewenangan untuk mengakses sumber daya yang diminta. Selain itu, akan diberikan pembatasan akses berdasarkan dari IP *request*.

3.1.2 Deskripsi Umum Sistem

Pada tugas akhir ini akan dibuat sebuah sistem manajemen API dengan berbasis protokol REST yang dapat dilihat pada Gambar 3.1. Pada sistem ini menggunakan WSO2 API *Manajement* yang nantinya akan disingkat menjadi WSO2 AM. Dalam manajemen API ini, pengembang dapat mempublish dan membuat API, mengatur sekuriti, dan juga melakukan analisa penggunaan API.



Gambar 3.1

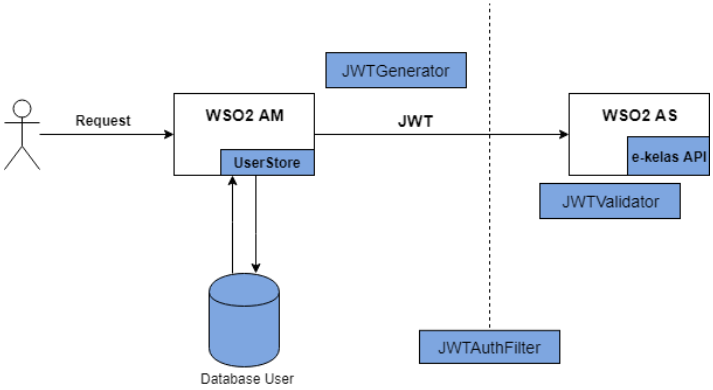
Manfaat lain yang didapatkan dengan adanya aplikasi manajemen API ini, pengembang juga dapat dengan mudah mengelola siklus hidup dari API ini, mengurangi waktu yang dihabiskan untuk membuat API dan memberikan layanan jauh lebih cepat. Selain itu, pengembang dapat mengatur dan menggabungkan layanan ke dalam satu API untuk membangun solusi. Perbandingan antara sistem yang lama dan baru setelah

diimplementasikan manajemen API ini dapat dilihat pada Tabel 3.1.

Tabel 3.1 Perbandingan Sistem Baru dan Sistem yang Dikembangkan

Parameter	Sistem Lama	Sistem yang Dikembangkan
Sekuriti	Mempunyai sistem sekuriti yang terpisah-pisah, dimana satu sistem mempunyai sistem sekuritinya sendiri.	Mempunyai sistem sekuriti yang sama untuk tiap sistem, hanya perlu menambahkan dependensi pada file xml.
Web servis	Web servis pada sistem yang lama berada secara terpisah.	Web servis terdapat pada aplikasi manajemen API.

Alur sistem keamanan ini dapat dilihat pada Gambar 3.2. Ketika terdapat *request* dari pengguna maka akan masuk dalam WSO2 AM yang di dalamnya terdapat *user store*. *User store* merupakan basis data yang berisi informasi tentang pengguna dan peran (*role*) pengguna disimpan, termasuk nama login, password, nama depan, nama belakang, dan alamat e-mail. Kemudian akan mengecek apakah pengguna memasukkan *username* dan *password* yang benar. Setelah berhasil masuk, maka akan menghasilkan (generate) JWT, dan masuk ke JWTAuthFilter untuk melakukan validasi dengan memanggil JWTValidator dan mengetahui apakah pengguna mempunyai hak akses atau tidak.



Gambar 3.2 Alur pada Sistem Keamanan

Pada tugas akhir kali ini, halaman antarmuka manajemen API WSO2 yang akan kustomisasi adalah antarmuka pada WSO2 API Store. Kustomisasi yang dilakukan ini menggunakan kerangka kerja Jaggery sesuai dengan yang digunakan pada WSO2.

3.2 Perancangan

Pada subbab ini dijelaskan mengenai tahapan perancangan sistem. Perancangan sistem ini dibagi menjadi beberapa bagian yang meliputi perancangan WSO2 AM, perancangan basis data, perancangan keamanan, dan perancangan *user store*.

3.2.1 Lingkungan Perancangan Perangkat Lunak

Spesifikasi perangkat keras serta perangkat lunak yang digunakan dalam tahap perancangan perangkat lunak tugas akhir ini seperti dijelaskan pada Tabel 3.2.

Tabel 3.2 Lingkungan Perancangan Perangkat Lunak

Perangkat Keras	Komputer	Dell Inspiron 14 3000 Series
	Prosesor	Intel® Core™ i5-3317U CPU @ 1.70GHz (1.70GHz)
	Memori Primer	4 GB

	Memori Sekunder	500 GB
Perangkat Lunak	Sistem Operasi	Windows 10 Pro 64-bit
	Perangkat Lunak	Java Development Kit 1.8, Sybase PowerDesigner 16.5, Microsoft Word 2013, Eclipse, Sublime Text 3, MySQL, MS SQL Server, Postman

3.2.2 Perancangan WSO2 AM

Pada WSO2 AM nantinya terdapat beberapa komponen utama yang dapat diakses. Namun, untuk dapat mengakses sistem tersebut perlu dilakukan beberapa konfigurasi terlebih dahulu.

3.2.2.1 Instalasi WSO2 AM

Langkah-langkah yang perlu dilakukan untuk melakukan instalasi WSO2 AM pada sistem operasi Windows adalah sebagai berikut:

Instalasi OpenJDK

1. Unduh JDK 1.8
2. Lakukan penginstalan
3. Atur JAVA_HOME, dengan cara buka Control Panel
4. Pilih Edit Environment for Your Account
5. Tambahkan Variable baru
6. Variable name : JAVA_HOME
7. Variable value : `c:/Program Files/Java/jdk1.8.0_121` (atau sesuai dengan path instalasi Java Development Kit)

Mengunduh Produk WSO2 API Manager 2.1.0

1. Melalui web *browser*, **http://wso2.com/products/api-manager**
2. Klik **Download**.

3. Masukkan email di form dan klik **Download**.

Ekstraksi File

1. Membuat direktori baru
2. Ekstraksi file wso2am-2.1.0.zip

Pengaturan Home Carbon

1. Buka **Control Panel**
2. Pilih **Edit Environment for Your Account**
3. Tambahkan Variable baru
   Variable name : CARBON_HOME
   Variable value : sesuai dengan direktori wso2 berada

Installasi WSO2

1. Buka **Command Prompt**
2. Masuk ke direktori **CARBON_HOME/bin**

```
cd CARBON_HOME/bin
```

3. Jalankan **wso2server.bat**

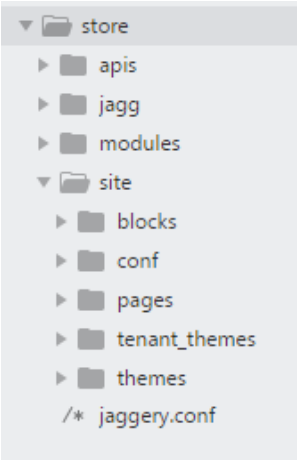
```
wso2server.bat
```

4. Setelah selesai run, buka browser dan buka URL https://localhost:9443/store untuk membuka "**store**", https://localhost:9443/publisher untuk membuka "**publisher**", dan https://localhost:9443/carbon untuk membuka "**carbon**".

3.2.2.2 Struktur Direktori JaggeryApps

Pada WSO2 API Store ini terdapat beberapa direktori dan juga file paten yang nantinya akan digunakan dalam proses kustomisasi *user interface*. Pada direktori Store ini, terdapat beberapa direktori yaitu, apis, jagg, modules, dan site. Di dalam direktori site sendiri terdapat direktori lain yaitu blocks, conf,

pages, tenant_themes, themes, dan terdapat file jaggery.conf. Struktur diretori ini dapat dilihat pada Gambar 3.3.

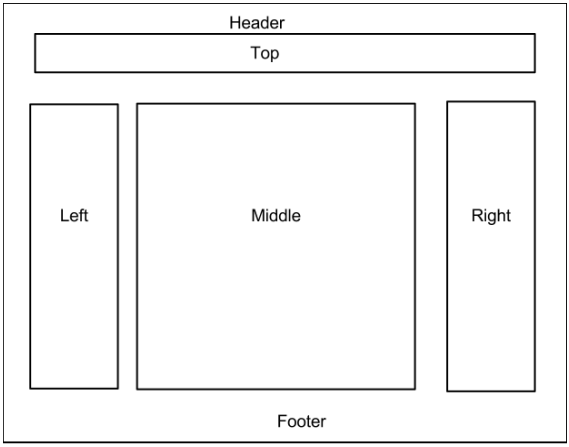


Gambar 3.3 Struktur Direktori JaggeryApps

1. **jaggery.conf**
   File **jaggery.conf** berfungsi untuk menentukan kofigurasi aplikasi. File ini digunakan untuk menentukan pemetaan URL untuk tiap halaman, menentukan halaman awal (*welcome page*), dan *security constrain*. Pemetaan URL ini berfungsi apabila terdapat permintaan ke aplikasi, maka aplikasi ini akan mencari URL yang dipetakan dalam **jaggery.conf**. Apabila cocok, maka file tersebut akan dipanggil.
2. **Direktori "site"**
   Di dalam direktori site terdapat beberapa direktori, yaitu
   a. **themes**
      Sistem ini mempunyai beberapa tema bawaan yang dapat digunakan oleh pengguna. Pengguna juga dapat menambahkan tema yang disesuaikan dengan aplikasinya. Setiap tema terdiri dari beberapa direktori, yaitu
      - **css**

Berisi file yang berfungsi untuk mengatur gaya tampilan atau CSS (*Cascading Style Sheet*) halaman website.

- **images**
  Berisi file gambar yang akan digunakan untuk halaman website.
- **Js**
  Berisi file JavaScript yang digunakan untuk halaman website.
- **template**
  Pada direktori **template** berisi tampilan aplikasi web. Tampilan tata telak pada halaman website telah dipisahkan menjadi beberapa **block**. Tata letak yang telah ditentukan pada halaman website API-Store dapat dilihat pada Gambar 3.4.



Gambar 3.4 Tata Letak Tampilan Antarmuka

Pada setiap **template** harus terdiri dari 2 file yaitu:

1. **template.jag**
   File ini berisi kode HTML. Template untuk halaman utama terdapat dalam direktori

templates/pages/base/template.jag. Pada halaman utama ini berisi style umum. Dalam suatu template, pengguna dapat menyertakan template lain. Hal tersebut dapat dilakukan dengan menggunakan kode berikut

```
<%jagg.includeBlock("TemplatePath",
{"pagePath":section,"tenant":""}); %>
```

Contoh penulisan file template ini dapat dilihat pada

```
1 <% jagg.template( "api", function(inputs, outputs,
jagg) { %>
2
3 <% });
4 %>
```

Kode Sumber 3.1 Contoh Kode pada template.jag

Tabel 3.3 Penjelasan Kode Sumber 3.1

No. Baris	Kegunaan
1	Output yang ditampilkan dari template dan input untuk template
2	Berisi semua HTML yang akan digunakan dalam website.

2. initializer.jag

File ini digunakan untuk menginisialisasi template yang menggunakan JavaScript atau CSS. Berikut adalah contoh kode initializer

```
1 <% jagg.initializer( "templatePath", {
2    preInitialize:function() {
3          jagg.addHeaderJS("templatePath", "key",
"jsFilePath");
4          jagg.addHeaderCSS("templatePath", "key",
"CSSFilePath");
5          }
6    });
7 %>
```

Kode Sumber 3.2 Contoh Kode initializer.jag

Tabel 3.4 Penjelasan Kode Sumber 3.2

No. Baris	Kegunaan
1	templatePath, merupakan lokasi dimana file template berada
2	jsFilePath, merupakan lokasi dimana file JavaScript berada
3	CSSFilePath, merupakan lokasi dimana file CSS berada

b. tenant_themes

Pada direktori berisi direktori ini berisi tema yang diunggah oleh *tenant* melalui sistem carbon.

c. Blocks

Direktori ini dapat dianggap sebagai bagian dari *controller*. Aktivitas user pada halaman web (**template**) akan dikirim ke **block** terkait dan kemudian akan menangani HTTP *request* yang masuk.

Setiap template memiliki **block** yang sesuai. Penanganan HTTP *request* yang masuk dilakukan dalam file .jag yang terdapat pada direktori **'blocks'** untuk tiap komponen. **Block** akan menerima request dan menerjemahkannya menjadi tindakan yang harus dilakukan oleh direktori **'modules'** yang terkait dengan file jag. Kemudian dengan respon dari **'modules'**, block memilih template yang sesuai untuk memberikan respon sebagai input pada template.

```
1 <%
 2     jagg.block("api", {
 3         initializer:function (data) {
 4
 5         },
 6         getInputs:function () {
 7
 8         },
 9         getOutputs:function(inputs) {
10
```

```
11          }
12      });
13  %>
```

Kode Sumber 3.3 Contoh Kode pada Block

Tabel 3.5 Penjelasan Kode Sumber 3.3

No. Baris	Kegunaan
3	Fungsi initializer untuk mendefinisikan sebuah **block**
7	Fungsi ini bersifat opsional dan digunakan untuk menentukan input
10	Melakukan beberapa operasi dari input **block**.

d. conf

Direktori ini berisi konfigurasi terkait dengan antarmuka web. Di dalam direktori ini terdapat file **site.json** berisi pengaturan untuk tema *default*.

e. pages

Direktori ini berisi halaman utama yang akan diakses oleh aplikasi web, atau inisialisasi lokasi awal yang akan muncul pada browser. Setiap halaman dibangun menggunakan blok (tata letak pada halaman web). Suatu blok memiliki nama, input, dan output. Untuk blok input dan output juga dapat berisi dari blok lain. Pada halaman ini, juga menangani rendering halaman web, dengan memanggil fungsi **render()** yang didefinisikan dalam file **jagg.jag** yang terdapat di lokasi "**jagg/jagg.jag**".

```
1  jagg.render({
 2 "name":"page/base",
 3     "inputs":{
 4         "title":"API Store Listing",
 5         "pagePath":"/site/pages/list-apis.jag",
 6         "body":[
 7             {
 8                 "name":"layout/custom",
 9                 "inputs":{
10                     "title":null,
11                     "middle":{
12                         "name":"api/tenant-stores-
listing"
13                     }
14                 }
15             }
16         ]
17     }
18 });}
```

Kode Sumber 3.4 Contoh Kode pada Page

Tabel 3.6 Penjelasan Kode Sumber 3.4

No. Baris	Kegunaan
4	Menentukan judul (*title*) halaman website
5	Lokasi **'page'**
9	Menentukan **"input"** apa saja yang akan digunakan di dalam *body*
12	'template' yang akan digunakan untuk bagian tengah

3. Direktori "modules"

Direktori ini bisa dianggap sebagai 'model' aplikasi. Module menangani segala bentuk keadaan aplikasi. Ketika sebuah **block** meminta suatu tindakan dari module tertentu dari

direktori ini, maka module akan memanggil fungsi yang sesuai.

4. **Direktori "jagg"**
   Direktori ini berisi file jagg.jag yang menangani semua fungsionalitas untuk mengendalikan dan mengelola interaksi antara modules, block, dan templates.

Tabel 3.7 *Method* pada jagg.jag

Method	**Kegunaan**
jagg.render(obj)	Untuk render konten HTML untuk halaman website tertentu.
jagg.getThemePath()	Untuk mengambil tema yang telah didefinisikan di **site/conf/site.json**
jagg.getUserTheme()	Untuk mengambil tema yang telah ditentukan oleh pengguna.
jagg.getAbsoluteUrl(path)	Untuk mengambil lokasi URL
jagg.addHeaderJS(template, key, js)	Untuk mengimpor file JavaScript ke template header yang telah ditentukan.

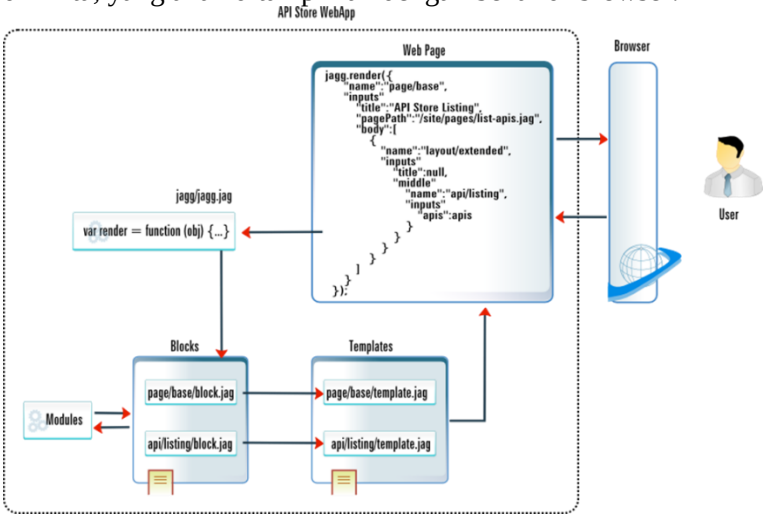
3.2.2.3 Alur *Request Front End* pada Browser

Pada Gambar 3.5 di bawah ini menjelaskan alur pemrosesan *request* pada browser yang terjadi di dalam aplikasi API Store. Hal yang sama juga terjadi pada aplikasi API Publisher. Ketika seorang pengguna mengakses aplikasi API Store/Publisher melalui browser, permintaan itu pertama-tama akan mendapatkan halaman yang relevan yang terletak di direktori "**site/pages**" di dalam API Store/API Publisher.

Kemudian akan dipanggil fungsi **jagg.render()** pada file **jagg.jag** yang terdapat pada direktori "**jagg**". Fungsi render() ini nantinya akan mencari konfigurasi blok sesuai dengan inputan yang tertera pada **pages**. Dan tiap blok yag dibutuhkan tadi akan meminta *action* yang relevan.

Pada contoh Gambar 3.5, akan dimuat halama API Store Listing. Pada file pages berisi blok nama base page, layout body, dan sub layout. Jadi di dalam base page, terdapat "layout/extend"

yang didalamnya akan memuat sub layout “api/listing” pada bagian tengah dari halaman. Setelah dungsi **render()** dipanggil, kemudian akan diperiksa apakah terdapat blok (site/blocks) dan juga template (site/themes/templates) yang relevan. Jika terdapat blok dan juga template yag relevan maka template yang cocok dengan nama blok yang tepat, dipanggil dengan meneruskan objek input dan output yang disebutkan di atas. Template akan menghasilkan konten HTML dan pengguna akan dapat melihat halaman web yang diminta, yang akan ditampilkan dengan benar di browser.

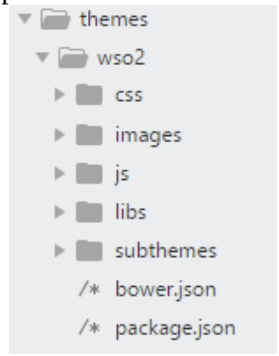


Gambar 3.5 Alur *Request Front End* pada Browser

3.2.2.4 Peracangan Kustomisasi Tema API Store

Sebuah tema terdiri dari elemen-elemen *user interface* contohnya adalah logo, gambar, warna *background*, dan lain lain. WSO API Store mempunyai sebuah tema bawaan. Pengguna dapat membuat sebuah tema baru atau menyesuaikan tema yang sudah ada. Tema bawaan API Store bernama **wso2** yang berada dalam <**API-M_HOME>/repository/deployment/server/jaggeryapps/store/site/themes**.

Cara termudah untuk membuat sebuah tema baru adalah menyalin file dari tema yang ada ke folder yang dinamai sesuai dengan nama tema baru dan kemudian melakukan modifikasi yang diinginkan ke file di dalamnya. Semua tema memiliki struktur folder yang sama seperti pada Gambar 3.6.



Gambar 3.6 Struktur Folder Tema API Store

Pengguna dapat menambahkan tema baru sebagai tema utama atau subtema.

1. **Tema Utama**
   Terdapat pada folder <**API-M_HOME>/repository/deployment /server/jaggeryapps/store/site/themes**.
2. **Sub Tema**
   Terdapat pada folder <**API-M_HOME>/repository/deployment /server/jaggeryapps/store/site/themes/<main-theme-directory>/subtheme**. Sebuah sub tema disimpan di dalam tema utama. Sub tema ini hanya berisi file yang berbeda dari tema utama. Semua file yang ditambahkan dalam sub tema akan menimpa file yang sesuai dalam tema utama. File lainnya akan mengikuti sesuai dengan file utamanya.

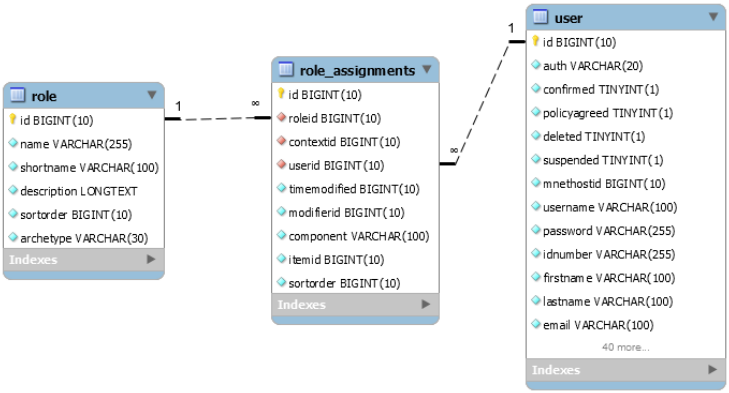
3.2.3 Perancangan Basis Data

Perancangan basis data pada sistem ini menggunakan 2 basis data. Basis data yang pertama merupakan basis data moodle dan basis data OIDC.

3.2.3.1 Basis Data Moodle

Basis data ini merupakan basis data bawaan dari moodle. Pada sistem ini terdapat beberapa tabel yang digunakan seperti pada Gambar 3.7.

1. **Tabel Role Assignments**
   Tabel ini menyimpan relasi antara pengguna (*user*) dan hak aksesnya (*role*).
2. **Tabel User**
   Tabel user ini merupakan tabel yang menyimpan rincian infomasi dari pengguna.
3. **Tabel Role**
   Tabel *role* adalah tabel yang digunakan untuk menyimpan data macam-macam hak akses untuk masuk ke dalam sistem. Terdapat beberapa role yang ada pada tabel ini, yaitu: *manager*, *coursecreator*, *teacher*, *student*, dan guest.



Gambar 3.7 Model Data Fisik Basis Data Moodle

3.2.3.2 Basis Data OIDC

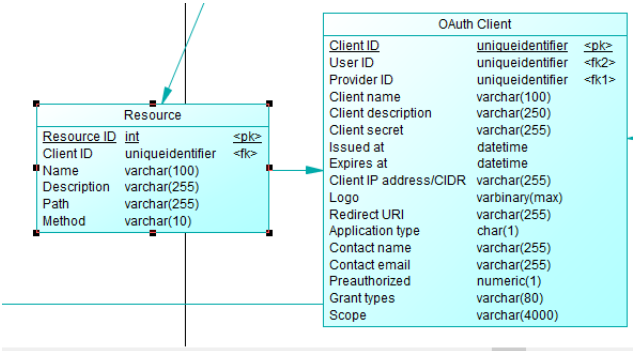
Pada basis data ini, hanya 2 tabel seperti pada Gambar 3.8 yang digunakan dalam sistem ini, yaitu

1. **Tabel Resource**

Tabel resource ini merupakan table yang menyimpan data resource pada tiap API. Tabel ini memiliki relasi dengan tabel OAuth Client. Tabel Resource dapat dilihat pada diagram OAuth2 pada Gambar 3.8.

2. **Tabel OAuth Client**

Tabel OAuth Client ini merupakan tabel yang menyimpan daftar IP yang mempunyai hak akses, yaitu pada kolom Client IP address/CIDR. Pada kolom tersebut nantinya terdapat 3 jenis penulisan IP, yaitu: IP tunggal, multipel IP, dan subnet. IP tunggal akan berisi 1 IP saja, sedangkan multipel IP akan berisi beberapa IP dan antar IP akan dipisahkan menggunakan tanda koma (,). Untuk subnet, maka akan berisi sebuah subnet.



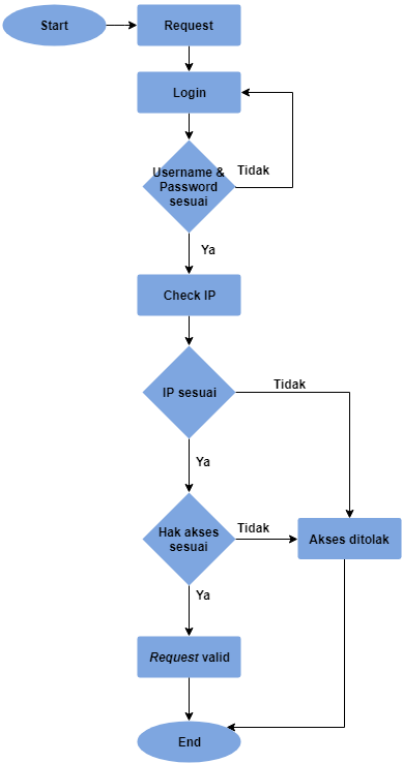
Gambar 3.8 Model Data Fisik Basis Data OIDC

3.2.4 Perancangan Keamanan

Keamanan pada sistem ini meliputi proses pengecekan IP request, otentikasi, dan otorisasi yang didukung oleh kerangka kerja *Spring Security*. Alur proses dari sistem sekuriti ini dapat dilihat pada Gambar 3.9.

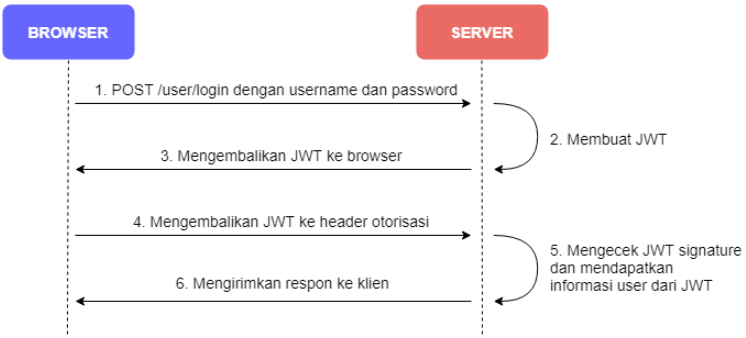
Ketika pengguna melakukan *request* maka akan terlebih dahulu dicek apakah proses otentikasi yang dilakukan oleh *user store* sudah benar atau belum. Kemudian akan dicek IP *request* dari pengguna, apakah IP tersebut mempunyai hak atau tidak. Apabila IP tersebut sesuai, maka akan dilakukan pengecekan peran (*role*)

dari pengguna apabila mempunyai hak akses maka *request* tersebut valid.



Gambar 3.9 Diagram Alur Proses Sekuritas

Otorisasi pada sistem ini menggunakan JWT. Setiap kali pengguna ingin mengakses *route* atau *resource* yang dilindungi, pengguna harus mengirim JWT. *Route* yang dilindungi server akan memeriksa JWT yang valid di header otorisasi, dan jika ada, pengguna akan diizinkan untuk mengakses *resource* yang dilindungi. Mekasisme proses otorisasi menggunakan JWT akan dijelaskan pada Gambar 3.10.



Gambar 3.10 Proses Jalannya Otorisasi

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi dari perancangan sistem sesuai dengan perancangan yang telah dibuat. Bahasa pemrograman yang digunakan untuk implementasi sistem adalah bahasa pemrograman Java.

5.1 Lingkungan Implementasi

Lingkungan implementasi sistem yang digunakan untuk mengembangkan tugas akhir memiliki spesifikasi perangkat keras dan perangkat lunak seperti ditampilkan pada Tabel 5.1.

Tabel 5.1 Lingkungan Implementasi Perangkat Lunak

Perangkat Keras	Komputer	Dell Inspiron 14 3000 Series
	Prosesor	Intel® Core™ i5-3317U CPU @ 1.70GHz (1.70GHz)
	Memori Primer	4 GB
	Memori Sekunder	500 GB
Perangkat Lunak	Sistem Operasi	*Windows* 10 Pro 64-bit
	Perangkat Lunak	Java Development Kit 1.8, Sybase PowerDesigner 16.5, Microsoft Word 2013, Eclipse, Sublime Text 3, MySQL, MS SQL Server, Postman

5.2 Standarisasi Penamaan Resource pada REST Service

Dalam pembuatan *endpoint* dibutuhkan sebuah standar yang konsisten. Standarisasi ini dibutuhkan agar mudah dipahami oleh para pengembang, baik pengembang yang sekarang maupun pengembang yang berikutnya.

5.2.1 Standarisasi Penamaan dan Struktur Endpoint

Terdapat beberapa aturan dalam penulisan endpoint

a. Menggunakan Kata Benda untuk Merepresentasikan *Resource*

   Dalam penamaan endpoint, URL hanya boleh berisi kata benda, bukan kata kerja. Dengan kata lain, harus mengacu ke suatu benda/hal bukan pada tindakan. Hal tersebut karna kata benda itu punya properti, sedangkan kata kerja tidak. Dalam penulisannya huruf kecil lebih dipilih karena lebih nyaman dibaca oleh pengguna dan lebih kosisten. Sedangkan aktifitas yang dilakukan oleh *endpoint* ditunjukkan oleh HTTP *methods* dari *endpoint* tersebut.

b. Menggunakan Garis Miring (/)

   Karakter garis miring (/) digunakan untuk memisahkan antar kata benda. Pada akhir karakter dalam penulisan *resource* tidak perlu menambahkan garis miring.

c. Menggunakan Tanda Hubung (-)

   Tanda hubung ini digunakan untuk memudahkan pengguna dalam membaca dan menafsirkan alamat *resource*. Penggunaan garis bawah (_) tidak disarankan. Hal tersebut karena, garis bawah dapat tak terlihat pada beberapa *browser*.

d. Tidak Mencantumkan ekstensi file

   Penambahan ekstensi file tidak menambahkan keuntungan apapun dan malah akan menambahkan panjang dari *resource*.

e. HTTP *Method*

   Terdapat beberapa HTTP *method* yang penting, yaitu

   1. GET

      GET *method* melakukan request data dari *resource*.

2. POST
   POST *method* melakukan *request* pada server untuk membuat (*create*) sebuah *resource* di basis data.
3. PUT
   PUT *method* melakukan *request* pada server untuk mengupdate *resource* atau membuat (create) resource apabila resource tersebut belum tersedia.
4. DELETE
   DELETE *method* melakukan *request* bahwa resource harus dihapus dari basis data.

Berikut merupakan contoh endpoint:

- **GET mahasiswa/5114100046** mendapatkan detail mengenai mahasiswa dengan NRP 5114100046.
- **GET mahasiswa/5114100046/nilai-mahasiswa** mendapatkan detail nilai mahasiswa dengan NRP 5114100046.

5.2.2 Respon Kode Status HTTP

Ketika klien mengajukan *request* ke server melalui API, klien harus mengetahui umpan balik (*feedback*), apakah *request* tersebut gagal atau permintaan salah. Kode status HTTP sekelompok kode standar yang memiliki berbagai penjelasan dalam berbagai skenario. Terdapat beberapa katerogi kode status HTTP, misalnya 200 apabila sukses, seri 400 apabila terjadi masalah pada sisi klien & seri 500 untuk masalah pada server. Keterangan mengenai respon kode status HTTP terdapat pada Tabel 5.2.

Tabel 5.2 Respon Kode Status HTTP

Kategori	Kode Status	Keterangan
2xx	200 Ok	Respon HTTP standar yang mewakili

Kategori	Kode Status	Keterangan
(Success Category)		kesuksesan untuk GET, PUT, atau POST
	201 Created	Respon HTTP yang menunjukan bahwa data berhasil ditambahkan
	204 No Content	Menunjukkan bahwa permintaan berhasil diproses, tetapi tidak ada hasil yang ditampilkan
3xx (Redirection Category)	304 Not Modified	Menunjukkan bahwa klien memiliki respon pada *cache* sehingga tidak perlu lagi mentransfer data yang sama
4xx (Client Error Category)	400 Bad Request	Menunjukkan bahwa permintaan dari klien tidak diproses, karena server tidak dapat memahami apa yang diminta oleh klien
	401 Unauthorized	Menunjukkan bahwa *request* membutuhkan otentikasi dari pengguna
	403 Forbidden	Menunjukkan bahwa server mengetahui *request* yang diinginkan, tetapi ditolak untuk diotorisasi
	404 Not Found	Menunujukkan bahwa request yang diminta tidak ditemukan / tidak tersedia.
	410 Gone	Menunjukkan bahwa *resource* yang diminta tidak lagi tersedia dan

Kategori	Kode Status	Keterangan
		dengan sengaja telah dipindahkan.
5xx (Server Error Category)	500 Internal Server Error	Menunjukkan bahwa *request* tersebut valid, tetapi server
	503 Service Unavailable	Menunjukkan bahwa server sedang down atau tidak dapat memproses *request*. Atau server sedang di *maintenance*

5.2.3 *Searching, Sorting, Filtering, Pagination*, dan Pembatasan *Field*

- ***Searching***

  Searching digunakan untuk merpresentasikan hasil yang diinginkan berdasarkan hasil yang diinputkan. Misalnya menggunakan ElasticSearch atau dengan menggunakan *query* 'LIKE'. Berikut adalah contoh *endpoint* untuk *seacrhing*:

  GET /name?search=Digital Mckinsey

- ***Sorting***

  Pada *sorting* juga terdapat parameter, dan nantinya hasil akan diurutkan sesuai parameter tersebut. Berikut adalah contoh *endpoint* untuk *sorting*:

  GET /name?sort=name _asc

- ***Filtering***

  Filtering berguna untuk membatasi jumlah *output* yang diminta berdasarkan beberapa parameter yang tersedia. Berikut adalah contoh *endpoint* untuk *filtering*:

  GET /ip?nama=raras

- **Pembatasan *Field***
  Konsumen API tidak selalu ingin mengambil informasi dari suatu *endpoint* secara menyeluruh. API yang dibuat dikegiatan ini, memungkinkan konsumen untuk memilih field-field yang dikembalikan sehingga bisa meminimalisir beban jaringan dan mempercepat penggunaan API. Untuk memfilter *field* output menggunakan parameter 'fields' dan diikuti nama field yang dibutuhkan dipisahkan dengan tanda koma. Dan nama *field* tersebut menggunakan snake_case, dimana semua kata ditulis menggunakan huruf kecil. Ketika nama *field* tersebut lebih dari satu suku kata maka kedua suku kata tersebut dihubungkan dengan tanda hubung under score(_).
  GET /pt/002002?fields=nama, sk_pendirian, status
- **Pagination**
  Paginasi berguna ketika terdapat yang *dataset* terlalu besar, dataset tersebut dapat dibagi menjadi bagian yang lebih kecil, yang membantu dalam meningkatkan kinerja dan lebih mudah untuk menangani respons. Berikut adalah contoh endpoint untuk paging:
  GET /ip?page=23&per_page=100

5.2.4 Versi API

Dalam pengembangan API diperlukan juga suatu standar dalam penulisan versi. Hal tersebut dikarenakan tidak ada suatu versi API yang benar – benar stabil. Seiring berjalannya waktu API tersebut akan terus disempurnakan dan akan meningkatkan versi dari API tersebut. Struktur Versi dari API terdiri Nama API dan diikuti 3 digit angka yang dipisahkan dengan titik. Format dari struktur API yang standar yaitu sebagai berikut:

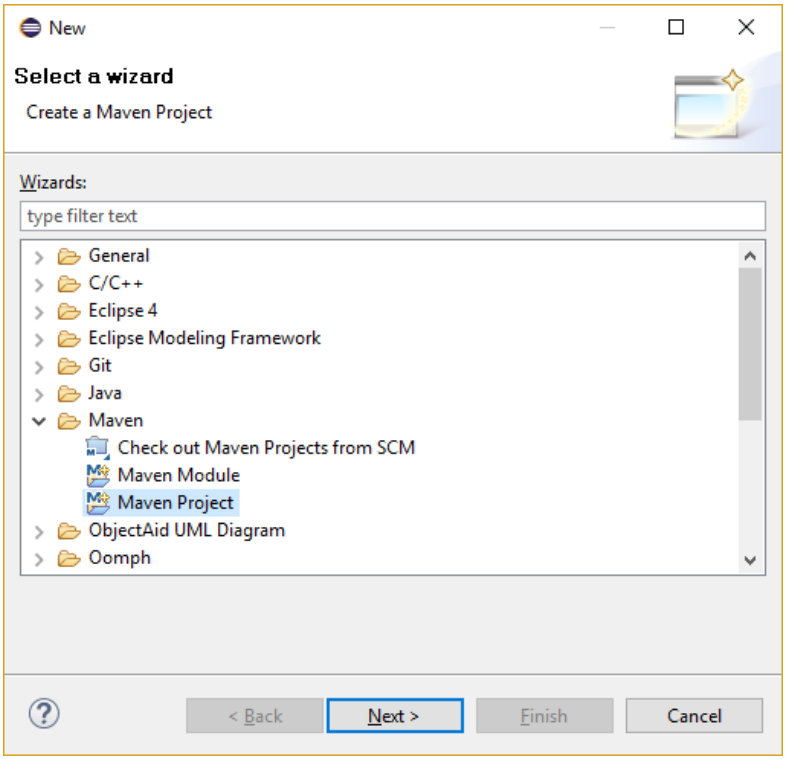
namaAPI.x.y.z

- **namaAPI** adalah nama dari API yang merepresentasikan secara garis besar fungsi dari API tersebut. Misalkan API tentang akademik maka nama yang sesuai dan merepresentasikan tentang akademik yaitu AkademikAPI yang diikuti dengan penomoran versi.
- **x** adalah penomoran versi untuk penambahan atau perubahan fungsi secara besar atau signifikan pada API tersebut.
- **y** adalah penomoran versi untuk perubahan maupun penambahan secara minor dari API tersebut.
- **z** adalah penomoran versi lebih kepada perbaikan fungsi endpoint pada sebuah API.

5.3 Pembuatan Proyek Maven

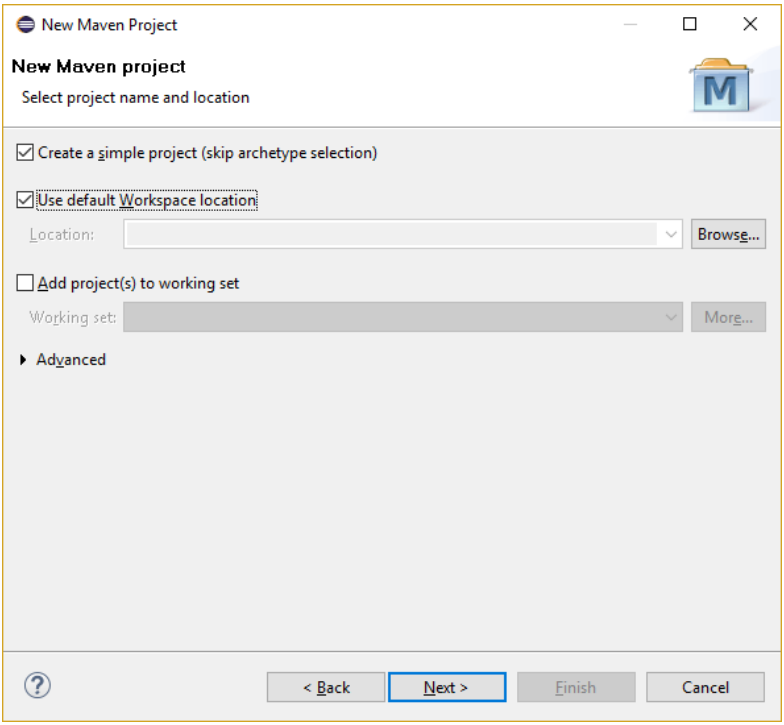
Pada subbab ini kana dijelaskan mengenai langkah-lagkah yang diperlukan dalam pembuata proyek web servis dengan maven *project* yang nantinya akan diperukan dalam implementasi keamanan dam pembuatan *user store*. Langkah-langkah yang dilakukan adalah sebagai berikut:

1. Klik File → New → Project → Maven Project



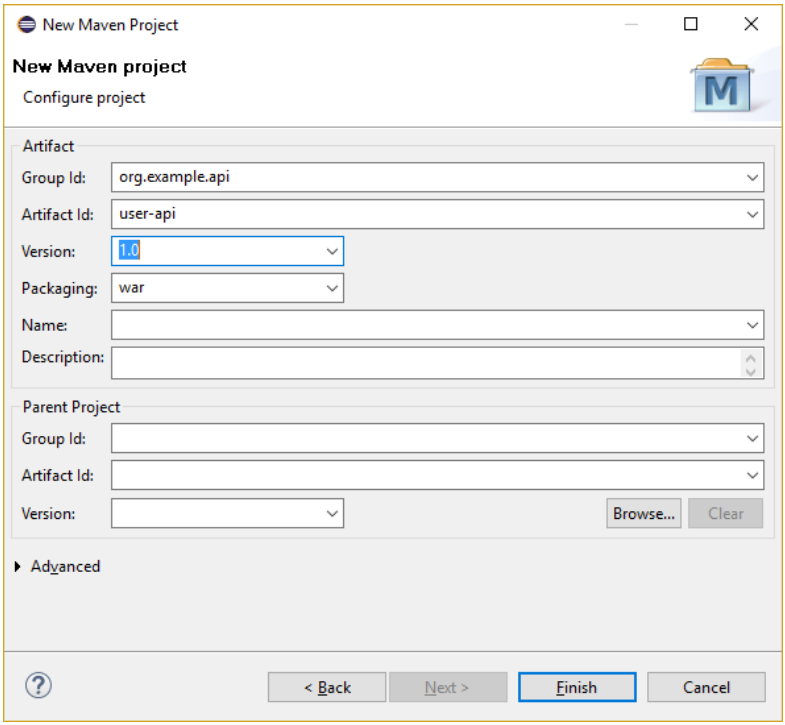
Gambar 5.1 Jendela "*New Project*"

2. Ketika muncul jendela "New Maven Project" seperti pada , pilih "Create a simple project". Tentukan tempat penyimpanan proyek atau gunakan lokasi *default*.



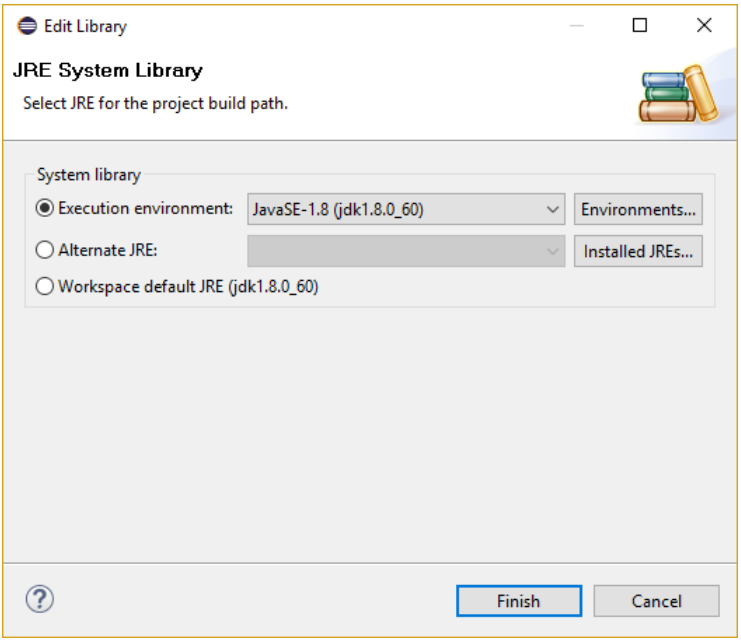
Gambar 5.2 Jendela "*New Maven Project*"

3. Masukkan informasi proyek pada bagian *Artifact*, berupa *Group Id, Artifact Id,* dan *Version.* Pada kolom *packaging* isi sesuai dengan hasil akhir proyek yang diinginkan, ketika packaging kosong maka Maven akan mengasumsikan sesuai *default*, yaitu .jar.



Gambar 5.3 Jendela Informasi *New Project*

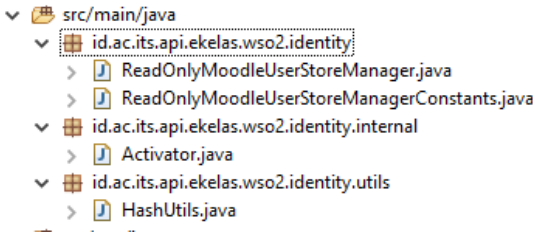
4. Ubah versi java menjadi JDK 1.8. Jika tidak sesuai ubah dengan klik kanan di JRE System Library → Build Path → Configure Build Path. Pada tab Library, pilih JRE System Library, dan klik tombol Edit, pilih versi java yang sesuai di mana pada proyek ini diperlukan jdk 1.8.



Gambar 5.4 Mengubah Versi Java

5.4 Implementasi Otentikasi (*User Store*)

Pada subbab ini akan dibahas mengenai langkah-langkah dalam pembuatan *user store*. Langkah awal yang perlu dilakukan adalah dengan membuat membuat *simple* maven *project* dengan *packaging* .jar. Setelah project berhasil dibuat, perlu ditambahkan beberapa package seperti pada Gambar 5.5.



Gambar 5.5 Struktur *Package* Maven Project User Store

5.4.1 Pom.xml

Pada file POM ini perlu ditambahkan beberapa dependensi dan juga plugin yang berkaitan dengan proyek ini. Tambahkan juga .jar dari WSO2 user store management yng tersimpan dalam Maven repository ke dalam dependensi seperti pada Kode Sumber 5.1.

```
<dependency>
    <groupId>org.wso2.carbon</groupId>
    <artifactId>org.wso2.carbon.user.core</artifactId>
    <version>4.4.11</version>
</dependency>
<dependency>
    <groupId>org.wso2.carbon</groupId>
    <artifactId>org.wso2.carbon.user.api</artifactId>
    <version>4.4.11</version>
</dependency>
<dependency>
     <groupId>org.wso2.carbon</groupId>
     <artifactId>org.wso2.carbon.utils</artifactId>
     <version>4.4.11</version>
</dependency>
```

Kode Sumber 5.1 Dependensi WSO2

Sertakan juga dependensi lain yang dibutuhkan. Dalam sistem ini dependensi lain yang dibutuhkan, yaitu mysql connector yang dependensinya dapat dilihat pada Kode Sumber 5.2. Selain itu, ubah *packaging* yang awalnya jar menjadi *bundle*, <packaging>bundle</packaging>.

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.44</version>
</dependency>
```

Kode Sumber 5.2 Dependensi MySql Connector

Tambahkan maven bundle plugin dan juga tambahkan maven scr plugin seperti pada Kode Sumber 5.3.

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-
plugin</artifactId>
    <version>2.0</version>
    <configuration>
        <source>1.8</source>
        <target>1.8</target>
    </configuration>
</plugin>
<plugin>
    <groupId>org.apache.felix</groupId>
    <artifactId>maven-scr-plugin</artifactId>
    <version>1.0.10</version>
    <executions>
        <execution>
            <id>generate-scr-scrdescriptor</id>
            <goals>
                <goal>scr</goal>
            </goals>
        </execution>
    </executions>
</plugin>
<plugin>
    <groupId>org.apache.felix</groupId>
    <artifactId>maven-bundle-plugin</artifactId>
    <version>1.4.0</version>
    <extensions>true</extensions>
    <configuration>
        <instructions>
            <Bundle-
SymbolicName>${project.artifactId}</Bundle-
SymbolicName>
            <Bundle-
Name>${project.artifactId}</Bundle-Name>
            <Private-
Package>id.ac.its.api.ekelas.wso2.identity.internal
```

```
    ,id.ac.its.api.ekelas.wso2.identity.utils</Private-
    Package>
33.             <Import-Package>
34.                 javax.servlet; version=2.4.0,
35.                 javax.servlet.http; version=2.4.0,

36.                 org.wso2.carbon.base.*,
37.                 org.wso2.carbon.user.core.*,
38.                 *;resolution:=optional
39.             </Import-Package>
40.             <Export-
    Package>!id.ac.its.api.ekelas.wso2.identity.interna
    l,
41.                 id.ac.its.api.ekelas.wso2.identity.
    *
42.             </Export-Package>
43.             <DynamicImport-
    Package>*</DynamicImport-Package>
44.         </instructions>
45.     </configuration>
46. </plugin>
```

Kode Sumber 5.3 Penambahan plugin Maven

5.4.2 ReadOnlyCustomUserStoreManager.java

User store dalam skenario ini berbasis JDBC sehingga dalam penulisannya perlu di *extend* dengan kelas org.wso2.carbon.user.core.jdbc.JDBCUserStoreManager. Dalam kelas ini terdapat beberapa *method*, yaitu

– *public boolean doAuthenticate(String username, Object credential)*
Method ini berfungsi untuk mengembalikan detail tentang apakah nama pengguna dan kata sandi yang diberikan cocok atau tidak.

1	Set isAuthed menjadi *false*
2	TRY Set koneksi basis data (dbConnection) mode commit otomatis menjadi *false* setAutoCommit(**false**) Menyimpan query dalam variabel sqlstmt

	mengambil query, set string 1 dengan username Menjalankan query untuk mengambil semua data user dan hasilnya disimpan pada variable *rs* IF Menyimpan kata sandi dari kolom *password* dalam variabel storedPassword apabila storedPassword tidak kosong dan storedPassword apabila hasilnya sama dengan hasil hash password dari kelas HashUtil Set isAuthed menjadi *true* Menutup semua koneksi
3	Return nilai dari isAuthed

Kode Semu 5.1 doAuthenticate(String username, Object credential)

- ***public boolean*** *doCheckExistingUser(String username)*
  Method ini berfungsi untuk mengembalikan apakah nama pengguna yang disediakan sudah ada di *user store*.

1	Menyimpan query pengguna yang usernamenya sama
2	Apabila `sqlStmt` kosong Maka akan menampilkan `"The sql statement for is user existing null"`
3	Return nilai `isValueExisting`

Kode Semu 5.2 doCheckExistingUser(String username)

- ***public*** *Properties getDefaultUserStoreProperties()*
  Method ini berfungsi untuk mengembalikan properti default dari *user store*. Properti ini digunakan dalam operasi terkait *user store*.
- ***protected boolean*** *isValueExisting*
  Method ini berfungsi untuk mengecek apakah terdapat koneksi ke basis data.
- ***public*** *String[] getRoleListOfUser(String username)*
  Method ini berfungsi untuk mengembalikan list dari user.

5.4.3 ReadOnlyCustomUserStoreManagerConstans.java

Dalam ReadOnlyCustomUserStoreManagerConstans.java pengguna dapat mengatur *mandatory*, *optional*, dan *advanced*. *Mandaroty* ini berfungsi untuk mendefinisikan database yang digunakan, termasuk driver, URL, username database, dan juga password database seperti pada Kode Sumber 5.4. *Advanced* ini berfungsi untuk mendefinisikan query untuk mengambil data pengguna dan juga peran pengguna.

```
1. setMandatoryProperty(JDBCRealmConstants.DRIVER_NAME, "Dr
   iver Name", "com.mysql.jdbc.Driver", "Full qualified dri
   ver name");
2. setMandatoryProperty(JDBCRealmConstants.URL, "Connection
   URL", "jdbc:mysql://localhost:3306/moodle", "URL of the
   user store database");
3. setMandatoryProperty(JDBCRealmConstants.USER_NAME, "User
   Name", "root", "Username for the database");
4. setMandatoryProperty(JDBCRealmConstants.PASSWORD, "Passw
   ord", "root", "Password for the database");
```

Kode Sumber 5.4 *Mandatory* untuk Koneksi Database

Tabel 5.3 Keterangan Kode Sumber 5.4

No. Baris	Kegunaan
1	Menentukan jenis basis data yang digunakan, pada sistem ini menggunakan mysql
2	Menentukan dimana basis data berada dan juga nama basis data, nama basis data ini adalah moodle
3	Menentukan username dari basis data moodle
4	Menentukan kata sandi dari basis data moodle

```
1.  setAdvancedProperty("SelectUserSQL", "Select User S
    QL", "SELECT * FROM user WHERE username=?", "");
2.  setAdvancedProperty("UserFilterSQL", "User Filter S
    QL", "SELECT username FROM user WHERE username LIKE
    " + " ? ORDER BY username", "");
```

```
3.  setAdvancedProperty("IsUserExistingSQL", "Is User E
    xisting SQL", "SELECT username FROM user WHERE " +
    "username=? ", "");
4.  setAdvancedProperty("UserRoleSQL", "User Role SQL",
    "SELECT shortname FROM role R, role_assignments UR,
    user U WHERE U.id = UR.userid AND R.id = UR.roleid
    AND U.username = ?", "");
```

Kode Sumber 5.5 *Advanced* untuk Mengambil data Pengguna dan Peran Pengguna

Tabel 5.4 Keterangan Kode Sumber 5.5

No. Baris	Kegunaan
1	Query untuk mengambil seluruh data pengguna
2	Query untuk memfilter data pengguna berdasarkan username
3	Query untuk mengambil data pengguna yang benar adanya
4	Query untuk mengambil peran dari pengguna

5.4.4 HashUtils.java

HashUtil ini merupakan file java yang di dalamnya berisi fungsi mengenai fungsi hashing password yang digunakan pada sebuah basis data. Fungsi hashing yang digunakan untuk mengenkripsi password pengguna pada basis data moodle ini menggunakan fungsi MD5. Pada HashUtils.java ini terdapat fungsi **activate(ComponentContext ctxt)**. Kemudian akan dibuat MessageDigest untuk MD5 dengan menggunakan **getInstance("MD5")**. Untuk menggunakan MessageDigest ini, perlu mengimport class **java.security.MessageDigest**. Menambahkan byte dari kata sandi (*password*) untuk dapat dipendekkan. Setelah didapatkan b5ayte hashnya dalam bentuk desimal maka harus diubah menjadi bentuk heksadesimal, sehingga hasil akhirnya berupa kata sandi yang ditelah dihash menjadi heksadesimal. HashUtils.java ini akan digunakan pada

ReadOnlyCustomUserStoreManager.java, sehingga nantinya akan diimport ke dalamnya.

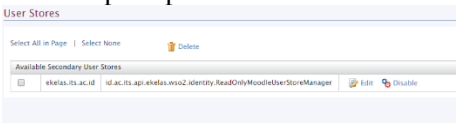
5.4.5 Activator.java

Sesuai dengan namanya, activator.java berguna untuk mengaktivasi. Aktivasi ini dengan mendaftarkan user store yang telah dibuat (*custom*) ke framework OSGI. Pada fungsi tersebut akan diambil data dari **ReadOnlyMoodleUserStoreManager()**.

5.4.6 Build

Setelah semua kode yang diperlukan telah dibuat, langkah-langkah yang dilakukan berikutnya adalah sebagai berikut

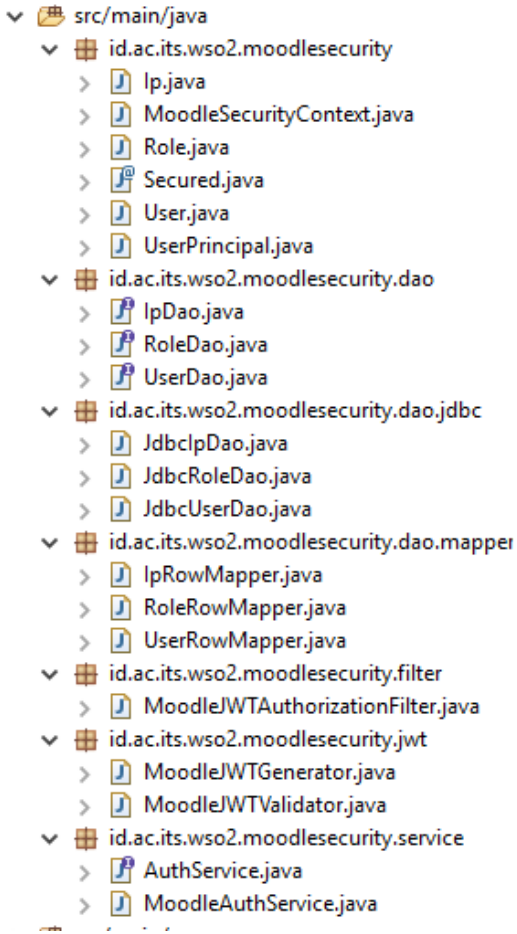
1. Compile proyek dengan cara klik kanan pada proyek → Run As → Maven install maka akan didapatkan **moodle-userstore.jar**.
2. Salin file moodle-userstore.jar tersebut ke dalam folder **APIM_HOME/repository/conf/dropins**.
3. Jalankan WSO2 AM
4. Buka dan masuk ke halaman https://localhost:9443/carbon/admin/login.jsp dengan username dan password admin maka secara otomatis user store yang telah dibuat tadi akan muncul di daftar user store seperti pada Gambar 5.6.



Gambar 5.6 Daftar *User Store*

5.5 Implementasi Otorisasi

Pada subab ini akan dibahas langkah-langkah yang diperlukan untuk membangun sistem keamanan yang meliputi proses pembatasan akses menggunakan IP dan otorisasi. Langkah pertama yang harus dilakukan dalam implementasi ini adalah dengan membuat Simple Maven Project dengan pengaturan *packaging* .jar.



t

Gambar 5.7 Struktur *Package* pada Moodle *Security*

5.5.1 Pom.xml

Pada file POM ini perlu ditambahkan beberapa dependensi dan juga plugin yang berkaitan dengan proyek ini. Sebelum mengidentifikasikan dependensi, perlu ditambahkan driver dari dua jenis basis data (MySQL dan SQLServer) ke dalam repositori Maven lokal. Penambahan itu dapat dilakukan dengan cara

mvn install:install-file –Dfile=sqljdbc42.jar –DgroupId=com.microsoft.sqlserver –D artifactId=sqljdbc –Dversion=4.2 –Dpackaging=jar

groupId, *artifactId*, dan *version* disesuaikan dengan driver yang digunakan pada proyek tersebut dan kemudian ditambahkan pada dependensi.

Sistem ini juga menggunakan Maven *Properties*. *Properties* ini berguna untuk menampung nilai. Nantinya nilai ini akan dapat diakses di mana saja di dalam POM, dengan menggunakan **${X}**, di mana X merupakan *property*. Selain itu, *properties* juga dapat digunakan plugin sebagai default. Terdapat beberapa p*roperty* yang digunakan, yaitu, Microsoft JDBC Driver versi 4.2, Spring Framework versi 3.0.7, JDK versi 1.8, HikariCP versi 2.4.7, Apache CXF versi 2.7.16, JUnit versi 4.12, Data Mapper For Jackson versi 1.9.13, dan MySQL Connector versi 5.1.44. Property yang telah didefinisikan ini nantinya akan dipanggil di dependensi yang berkaitan.

```
1.  <properties>
2.      <sqljdbc.version>4.2</sqljdbc.version>
3.      <spring.version>3.0.7.RELEASE</spring.version>
4.      <jdk.version>1.8</jdk.version>
5.      <hikaricp.version>2.4.7</hikaricp.version>
6.      <cxf.version>2.7.16</cxf.version>
7.      <junit.version>4.12</junit.version>
8.      <jackson.version>1.9.13</jackson.version>
9.      <mysqljdbc.version>5.1.44</mysqljdbc.version>
10. </properties>
```

Kode Sumber 5.6 Properties pada pom.xml Moodle-Security

Dependensi lain yang digunakan pada sistem ini terdapat pada.

```
1.  <dependency>
```

```
2.      <groupId>mysql</groupId>
3.      <artifactId>mysql-connector-java</artifactId>
4.      <version>${mysqljdbc.version}</version>
5.      <scope>runtime</scope>
6.  </dependency>
7.  <dependency>
8.      <groupId>com.zaxxer</groupId>
9.      <artifactId>HikariCP</artifactId>
10.     <version>${hikaricp.version}</version>
11.     <scope>provided</scope>
12. </dependency>
13. <dependency>
14.     <groupId>com.microsoft.sqlserver</groupId>
15.     <artifactId>sqljdbc</artifactId>
16.     <version>${sqljdbc.version}</version>
17.     <scope>runtime</scope>
18. </dependency>
19. <dependency>
20.     <groupId>org.springframework</groupId>
21.     <artifactId>spring-test</artifactId>
22.     <version>${spring.version}</version>
23.     <scope>test</scope>
24. </dependency>
25. <dependency>
26.     <groupId>org.springframework</groupId>
27.     <artifactId>spring-context</artifactId>
28.     <version>${spring.version}</version>
29.     <scope>provided</scope>
30. </dependency>
31. <dependency>
32.     <groupId>org.springframework</groupId>
33.     <artifactId>spring-jdbc</artifactId>
34.     <version>${spring.version}</version>
35.     <scope>provided</scope>
36. </dependency>
37. <dependency>
38.     <groupId>org.wso2.carbon.apimgt</groupId>
39.     <artifactId>org.wso2.carbon.apimgt.keymgt</arti
    factId>
40.     <version>6.0.4</version>
41.     <scope>provided</scope>
42. </dependency>
43. <dependency>
```

```
44.     <groupId>org.slf4j</groupId>
45.     <artifactId>slf4j-api</artifactId>
46.     <version>1.5.10</version>
47. </dependency>
48. <dependency>
49.     <groupId>commons-codec</groupId>
50.     <artifactId>commons-codec</artifactId>
51.     <version>1.10</version>
52. </dependency>
53. <dependency>
54.     <groupId>com.googlecode.json-simple</groupId>
55.     <artifactId>json-simple</artifactId>
56.     <version>1.1.1</version>
57. </dependency>
58. <dependency>
59.     <groupId>junit</groupId>
60.     <artifactId>junit</artifactId>
61.     <version>${junit.version}</version>
62.     <scope>provided</scope>
63. </dependency>
64. <dependency>
65.     <groupId>com.auth0</groupId>
66.     <artifactId>java-jwt</artifactId>
67.     <version>3.1.0</version>
68.     <scope>provided</scope>
69. </dependency>
70. <dependency>
71.     <groupId>commons-net</groupId>
72.     <artifactId>commons-net</artifactId>
73.     <version>3.6</version>
74. </dependency>
```

Kode Sumber 5.7 Dependensi yang Digunakan pada Moodle Security

Tabel 5.5 Penjelasan Kode Sumber 5.7

No. Baris	Kegunaan
1 – 6	Dependensi untuk driver basis data MySQL
6 – 12	Dependensi untuk HikariCP
13 – 18	Dependensi untuk driver basis data SQLServer

No. Baris	Kegunaan
19 – 36	Dependensi untuk framework Spring, yang terdiri dari Spring Test, Spring Context, dan Spring JDBC.
37 – 42	Dependensi untuk implementasi WSO2 Carbon API Management
43 – 47	Dependensi untuk SLF4J API Module dengan versi 1.5.10
48 – 52	Dependensi untuk Apache Commons Codec versi 1.10, berisi encoder dan decoder sederhana untuk berbagai format seperti Base64 dan Hexadecimal
53 – 57	Dependensi untuk JSON.simple dengan versi 1.1.1, yang merupakan toolkit untuk JSON
58 – 63	Dependensi untuk JUnit yang berguna untuk unit testing Java
64 – 69	Dependensi untuk Java JWT dengan versi 3.1.0
70 – 74	Dependensi untuk Apache Commons Net dengan versi 3.6, merupakan library yang berisi kumpulan utilitas jaringan dan implementasi protocol.

5.5.2 Koneksi ke Basis Data

Pada langkah akan dibuat file XML dengan nama **moodle-moodle-db-config.xml** di dalam folder **src/main/resources**. Ini berguna untuk mendefinisikan basis data yang digunakan. Pada sistem ini menggunakan 2 jenis basis data, yang pertama menggunakan MySQL dan yang kedua menggunakan SQL Server. Terdapat 2 cara dalam megkonfgurasi basis data, yang tidak menggunakan HikariCp dan menggunakan HikariCp.

```
1.  <bean id="moodleDataSource"
2.      class="org.springframework.jdbc.datasource.Driver
            ManagerDataSource">
3.      <property name="driverClassName" value="com.mysql
            .jdbc.Driver" />
4.      <property name="url" value="jdbc:mysql://localhos
            t:3306/moodle" />
5.      <property name="username" value="username" />
6.      <property name="password" value="password" />
```

```
7.  </bean>
```

Kode Sumber 5.8 Konfigurasi Basis Data Tanpa Menggunakan HikariCp

Tabel 5.6 Penjelasan Kode Sumber 5.8

No. Baris	Kegunaan
1	Berfungsi untuk menentukan id dari data source
3	Sebagai keterangan bahwa menggunakan driver MySql
4	Menentukan URL dari basis data dan juga nama dari basis data yang digunakan, yaitu `localhost:3306/moodle`
5	Menentukan username dari basis data yang digunakan
6	Menentukan password dari basis data yang digunakan

```
1.  <bean id="userHikariConfig" class="com.zaxxer.hikar
    i.HikariConfig">
2.      <property name="poolName" value="pdptPool" />
3.      <property name="connectionTestQuery" value="SEL
    ECT 1" />
4.      <property name="dataSourceClassName"
5.          value="com.microsoft.sqlserver.jdbc.SQLServ
    erDataSource" />
6.      <property name="maximumPoolSize" value="10" />

7.      <property name="idleTimeout" value="10000" />
8.      <property name="dataSourceProperties">
9.          <props>
10.             <prop key="url">jdbc:sqlserver://10.107
    .1.176:1433;databaseName=sso;</prop>
11.             <prop key="user">username</prop>
12.             <prop key="password">password </prop>
13.         </props>
14.     </property>
15. </bean>
16.
```

```
17. <bean id="ipDataSource" class="com.zaxxer.hikari.Hi
    kariDataSource" destroy-method="close">
18.     <constructor-arg ref="userHikariConfig" />
19. </bean>
```

Kode Sumber 5.9 Konfigurasi Basis Data Menggunakan HikariCp

Tabel 5.7 Penjelasan Kode Sumber 5.9

No. Baris	Kegunaan
1	Berfungsi untuk menentukan id dari pool
2	Berfungsi untuk merepresentasikan nama yang ditentukan oleh pengguna untuk kumpulan koneksi.
3	Query ini akan dijalankan tepat sebelum koneksi diberikan kepada pengguna, yang fungsinya untuk memvalidasi bahwa koneksi ke basis data masih hidup
4	Berfungsi untuk menentukan driver apa yang digunakan pada basis data, dimana pada basis data ini menggunakan driver SqlServer
6	Berfungi untuk mengatur nilai maksimum yang diizinkan untuk menjangkau koneksi, termasuk koneksi yang tidak aktif dan yang sedang digunakan.
7	Berfungsi untuk mengontrol waktu maksimum untuk koneksi dalam keadaan idle.
10	Berfungsi untuk mengarahkan HikariCP menggunakan konfigurasi URL dari basis data dan nama basis data ini, yaitu `10.107.1.176:1433` dengan nama basis data sso
11	Berfungsi menentukan username dari basis data yang digunakan
12	Berfungsi menentukan password dari basis data yang digunakan
17	Berfungsi untuk merepresentasikan id dari data source

5.5.3 Konfigurasi Dependency Injection

Pada pembuatan API, *dependency injection* dilakukan pada berkas XML yang merupakan salah satu cara *dependency*

injection pada kerangka kerja Spring. Pada kerangka kerja Spring, dikenal istilah *bean* yang berisi konfigurasi *metadata.* Sebuah *bean* merujuk ke suatu kelas yang akan menerima injeksi. Pada *bean*, terdapat beberapa properti yang membentuk konfigurasi *bean*. Contohnya adalah *properties* yang digunakan untuk melakukan injeksi dari *bean* melalui *method setter*. Terdapat beberapa berkas XML, di antaranya

5.5.3.1 moodle-data-beans.xml

Berkas ini mempunyai fungsi untuk menginjeksikan dataSource yang telah didefinisikan sebelumnya (db-config) ke JdbcIpDao, UserIpDao, dan JdbcRoleDao. Hal ini berarti, setiap kueri yang akan dilakukan pada kelas tersebut bersumber dari basis data yang telah didefinisikan pada *dataSource*.

5.5.3.2 moodle-service-beans.xml

Berkas ini mempunyai fungsi untuk menginjeksi objek JdbcIpDao, UserIpDao, dan JdbcRoleDao ke dalam kelas MoodleAuthService.

5.5.3.3 moodle-filter-beans.xml

Berkas ini mempunyai fungsi utnuk menginjeksi objek MoodleAuthService ke kelas MoodleJWTAuthorizationFilter.

5.5.4 *Package* Model

Package ini diberi nama id.ac.its.api.wso2.moodlecesurity. *Package* ini berisi kelas-kelas yang memiliki *getter* dan *setter* untuk mengambil atribut pada data yang nantinya digunakan sebagai tipe data dari pertukaran data melalui JSON. Untuk mendukung JSON pada Java, diperlukan pustaka *Jackson*. Pemasangan pustaka *Jackson* sudah disertakan pada subbab 5.5.1.

5.5.5 *Layer* Data Access Object (DAO)

Layer ini berisi tiga package yang berisi kelas interface DAO, kelas realisasinya, dan juga kelas mapper. Kelas-kelas DAO

bertugas menyimpan perubahan data dari perangkat lunak ke basis data ataupun sebaliknya.

5.5.5.1 *Package DAO*

Pada *package* ini berisi *interface* DAO berisi *method* untuk mengakses basis data, yang dibagi berdasarkan kelas entitasnya. Kelas-kelas tersebut adalah

- IpDao
  Kelas IpDao ini berisi *method* yang berkaitan dengn pengambilan data Ip.
- RoleDao
  Kelas RoleDao ini berisi *method* yang berkaitan dengan pengambilan data peran pengguna.
- UserDao
  Kelas UserDao ini berisi *method* yang berkaitan dengan pengambilan data pengguna.

Pada Kode Sumber 5.10 merupakan kelas yang menunjukkan kelas *interface* UserDao

```
1.  public interface UserDao {
2.      List<User> getUser(String username);
3.  }
```

Kode Sumber 5.10 Contoh pada Kelas *Interface* UserDao

5.5.5.2 *Package* DAO JDBC

Package ini berisi kelas realisasi *method* dari *package* sebelumnya (*Package* DAO). Kelas-kelas pada package ini adalah

- JdbcIpDao
- JdbcRoleDao
- JdbcuserDao

Dalam kelas-kelas tersebut akan didefinisikan query berupa membaca data dari basis data yang berbeda sesuai dengan kelas entitas yang ditangani.

5.5.5.3 *Package* DAO Mapper

Pada *package* ini akan didefinisikan pemetaan hasil dari pengambilan data dari *package* DAO Jdbc. Setiap baris data dimodelkan menjadi sebuah objek pada *package model*. Kelas-kelas pada package ini adalah

- IpRowMapper
- RoleRowMapper
- UserRowMapper

5.5.6 *Package* Service

Package ini berisi kelas yang menggabungkan *data access object* (DAO) ke *filter*. Pada kelas ini terdapat logika yang diperlukan dalam pemrosesan data.

5.5.7 Package JWT

Package ini diberi nama id.ac.its.api.wso2.moodlecesurity.jwt.

5.5.7.1 MoodleJWTGenerator

Secara garis besar JWTGenerator mempunyai tugas untuk mengenerate token. Java class bernama MoodleJWTGenerator yang diperluas (*extends*) dari *class default* AbstractJWTGenerator yang merupakan class implementasi dari org.wso2.carbon.apimgt.keymgt.token.AbstractJWTGenerator.
Pada class ini mengimplementasikan 2 *method*, yaitu populateCustomClaims() dan populateStandardClaims(). Pada populateCustomClaims() akan menghasilkan beberapa *claim* khusus sesuai dengan keinginan pengembang dan menambahkannya ke JWT.

1	Generate waktu kadaluarsa (`expireIn`), dengan mendapatkan waktu saat ini dalam bentuk millisecond

	ditambah dengan time to live dikalikan 1000 (`System.currentTimeMillis() + getTTL() * 1000`)
2	Sediakan variable `dialect`
3	Set `claimsRetriever` dengan cara `getClaimsRetriever()`
4	IF `claimsRetriever` tidak NULL Maka `dialect` akan didapat dari `getValidationInfoDTO().getEndUserName()` ELSE Maka akan didapat dari `getDialectURI()`
5	Mendefinisikan tiap claim yang akan ditambahkan, sebagai contoh untuk menambahkan claim subscriber `String subscriber = validationContext.getValidationInfoDTO().getSubscriber();`
6	Menggabungkan / mengaitkan claim, sebagai contoh `claims.put(dialect + "/subscriber", subscriber);`
7	Mengembalikan (*return*) claim, sehingga hasil akhir untuk claim subscriber seperti ini `"http://wso2.org/claims/subscriber":"admin"` *Issuer*, yang menerbitkan JWT → `claims.put("iss", API_GATEWAY_ID);` *Expiration time*, waktu kadaluarsa dari token → `claims.put("exp", String.valueOf(expireIn));`

Kode Semu 5.3 *Method* populateCustomClaims()

```
1. String subscriber = validationContext.getValidation
   InfoDTO().getSubscriber();
2. String applicationName = validationContext.getValid
   ationInfoDTO().getApplicationName();
3. String applicationId = validationContext.getValidat
   ionInfoDTO().getApplicationId();
4. String applicationTier = validationContext.getValid
   ationInfoDTO().getApplicationTier();
5. String tier = validationContext.getValidationInfoDT
   O().getTier();
```

```
6.  String endUserName = validationContext.getValidatio
    nInfoDTO().getEndUserName();
7.  String userType = validationContext.getValidationIn
    foDTO().getUserType();
8.  String keyType = validationContext.getValidationInf
    oDTO().getType();
```

Kode Sumber 5.11 List dari Claim WSO2

Tabel 5.8 Penjelasan Kode Sumber 5.11

No. Baris	Kegunaan
1	Mendapatkan subscriber dari API, yang biasanya pengembang aplikasi
2	Mendapatkan nama application melalui permintaan API yang dilakukan
3	Mendapatkan id application melalui permintaan API yang dilakukan
4	Mendapatkan tingkat aplikasi, yang dipilih saat membuat aplikasi. Ini adalah Per Token Quota
5	Mendapatkan tingkatan untuk berlangganan
6	Mendapatkan *enduser* aplikasi yang tindakannya meminta API
7	Mendapatkan jenis pengguna yang dikeluarkan token aplikasi
8	Mendapatkan jenis kunci (Production or Sandbox) yang digunakan untuk permintaan API

5.5.7.2 MoodleJWTValidator.java

Secara umum kelas ini bertugas untuk memvalidasi jwt. Dalam kelas ini terdapat beberapa *method*, yaitu

- ***public boolean*** *isValid()*

  Method ini berfungsi untuk memvalidasi token dari JWT.

1	Mendapatkan token JWT yang dipisahkan menggunakan titik (.), untuk memisahkan *header*

	(`jwtTokenValues[0]`), *payload* (`jwtTokenValues[1]`), dan *signature* (`jwtTokenValues[0]`)
2	IF length `jwtTokenValues` lebih dari 0 Mengambil array ke 0 dari `jwtTokenValues` yang merupakan header getByte Encoding pakai base Membuat objek JSONparser Mengisi JSON dengan value yang di decode
3	`IF jwtAssertion != null && jwtSignature != null` Memverifikasi certificate pada JWT yang telah didapatkan pada *method* getAliasForX509CertThumb()

Kode Semu 5.4 *Method* isValid()

- ***private*** *String getAliasForX509CertThumb()*
  Method ini berguna untuk mendapatkan alias sertifikat X.509 dari *key store* yang nantinya akan digunakana pada *method* `isValid()`.
- ***private*** *String hexify(**byte** bytes[])*
  Method ini berguna untuk mengubah ata mengkonversikan byte array menjadi heksadesimal yang nantinya akan dipanggil pada *method* `getAliasForX509CertThumb()`.

5.5.8 Package JWT Filter

Package ini diberi nama id.ac.its.api.wso2.moodlecesurity.filter.

5.5.8.1 JWTAuthFilter.java

Pada kelas ini terdapat beberapa *method*, yaitu

- ***public void*** *filter(ContainerRequestContext ctx)*
  Method ini berfungsi untuk melakukan verifikasi apakah pengguna mempunyai kewenangan sesuai dengan perannya (*role*) dan juga IP *request* atau tidak.

1	Mendapatkan JWT Header dan ip *request*
2	IF return dari `checkIp(ipRequest) bernilai true` IF `jwtHeader` tidak kosong

```
Menyimpan JWT dari kelas MoodleJWTValidator
dalam variabel tokenValidator
IF tokenValidator valid
  Mendapatkan claim Json
   IF userType = APPLICATION_USER
       TRY
          IF methodRoles kosong
            Maka akan memanggil kelas
            checkPermissions dengan parameter
            (enduser, classRoles)
          ELSE
            Maka akan memanggil kelas
            checkPermissions dengan parameter
            (enduser, methodRoles)
       CATCH
         Maka akan menampilkan respon status
       forbidden
    ELSE IF userType = APPLICATION
       Akan memanggil fungsi extractType dan
       disimpan dalam variabel
       methodSecurityLevel
       IF methodSecurityLevel_kosong maka akan
       memberikan respon status forbidden
     ELSE akan memberikan respon status forbidden
   ELSE akan memberikan respon status
   unauthorized
 ELSE akan memberikan respon status unauthorized
ELSE akan memberikan respon status unauthorized
```

Kode Semu 5.5 *Method* filter(ContainerRequestContext ctx)

– ***private*** *List<String> extractRoles(AnnotatedElement element)* *Method* ini berfungsi untuk mendapatkan `Annotation Secure "roles"` yang terdapat pada API dan menyimpannya di dalam array `allowedRoles`.

– ***private*** *String extractType(AnnotatedElement element)*
Method ini berfungsi untuk mendapatkan `Annotation Secure` "type", pada sistem type ini bernilai default Application.

– ***private void*** *checkPermissions(String enduser, List<String> allowedRoles)*
Method ini berfungsi untuk mendapatkan peran (*role*) dari *enduser* dan juga mengecek apakah peran tersebut sesuai atau tidak.

1	Set `allowed` menjadi *false*
2	Menyimpan list role yang didapat dari `authService` dalam variabel `roles`
3	FOR setiap objek role Role list FOR setiap `allowedRole : allowedRoles` IF `allowedRole` sama dengan role Akan mengubah nilai `allowed` menjadi *true*

Kode Semu 5.6 *Method* checkPermissions(String enduser, List<String> allowedRoles)

– ***public static*** *String getClientIpAddress (HttpServletRequest request)*
Method ini mempunyai fungsi untuk mendapatkan IP dari pengguna yang melakukan *request*.

1	Menyimpan request header dalam variabel `xForwardedForHeader`
2	IF `xForwardedForHeader` kosong Maka `xForwardedForHeader` akan diset dengan `request.getRemoteAddr()`
3	Return hasil dari `xForwardedForHeader`

Kode Semu 5.7 *Method* getClientIpAddress (HttpServletRequest request)

– ***private boolean*** *checkIp (String Ip)*

Method ini berfungsi untuk mengecek apakah IP *request* dari pengguna mempunyai hak akses atau tidak.

1	Menyimpan list Ip yang didapat dari `authService` dalam variabel `ips`
2	Set nilai `allowed` menjadi *false*
3	IF size ips lebih dari 0 Maka set `allowed` mejadi *true* ELSE Menyimpan list subnet yang didapat dari `authService` dalam variabel `subnet` FOR setiap objek IP subnet list s Akan menyimpan ip request dalam variabel `utils` Apabila ip request berada di rentang subnet maka `allowed` akan menjadi *true*
4	Return nilai dari `allowed`

Kode Semu 5.8 checkIp (String Ip)

5.6 Implementasi Kustomisasi Antarmuka WSO2 API Store

Pada WSO2 ini pengguna mempunyai keleluasaan untuk mengatur sendiri antarmuka sesuai dengan yang diinginkan oleh pengguna itu sendiri.

5.6.1 Pengaturan Tema Baru sebagai Tema Utama

Terdapat dua cara untuk mengatur tema baru atau tema yang telah dibuat oleh pengguna menjadi tema utama, yaitu dengan menyimpan langsung dalam sistem file dan mengunggah di admin portal.

5.6.1.1 Menyimpan dalam Sistem File

Langkah ini dipilih apabila pengguna mempunyai hak akses ke file sistem. Langkah-langkah yang dilakukan adalah sebagai berikut:

1. Simpan folder **<THEME_HOME>** yang bersi tema baru.

2. Buka file **<API-M_HOME>/repository/deployment/server/jaggeryapps/store/site/conf/site.json** dan tambahkan kode di bawah ini

```
1 "theme" : {
2          "base" : "wso2",
3          "subtheme" : "ancient"
4      }
```

Kode Sumber 5.12 Contoh Kode Pengaturan Tema

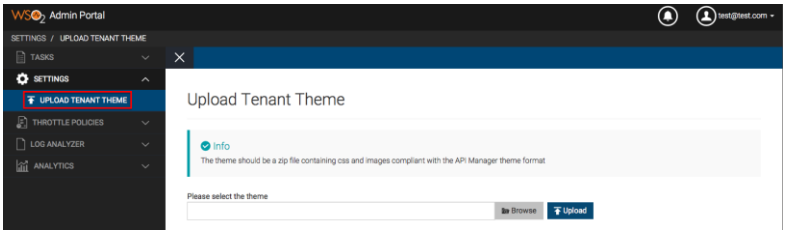
Tabel 5.9 Penjelasan Kode Sumber 5.12

No. Baris	Kegunaan
2	Menentukan nama tema yang akan digunakan
3	Menentukan sub tema

5.6.1.2 Mengunggah di Admin Portal

Langkah ini digunakan apabila pengguna tidak mempunyai hak akses. Pengguna dapat mengunggah tema ke Admin Portal seperti langkah dibawah

1. Masuk ke dalam folder <**THEME_HOME**>, pilih semua file di dalamnya dan klik kanan untuk memampatkan (*compress*).
2. Ganti nama file ZIP sesuai dengan nama sub tema.
3. Sign in pada halaman website WSO2 Admin Portal (**Error! Hyperlink reference not valid.**) menggunakan username tenan yang dimiliki.
4. Klik menu **Setting**, kemudian klik **Upload Tenant Theme** dan unggah file ZIP yang sebelumnya.
5. Akses API Store (**Error! Hyperlink reference not valid.**) untuk melihat hasil dari tema baru.



Gambar 5.8 Halaman Unggah Tema Baru

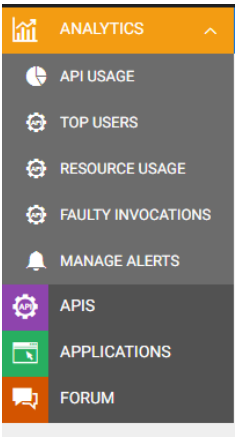
5.6.2 Kustomisasi Antarmuka WSO2 API Store

Terdapat beberapa halaman dari WSO2 API Store yang dikustom, yaitu

5.6.2.1 Kustomisasi Antarmuka Header

Pada Gambar 5.9 merupakan antarmuka header sebelum dilakukan kustomisasi. Pada halaman web asli API Store ini menggunakan *left navigation* seperti pada Gambar 5.10. *Left navigation* yang digunakan pada web ini tidak statis atau berpindah urutan setiap dilakukan pemilihan menu oleh pengguna. Oleh karena itu, *left navigation* ini dianggap kurang *user friendly*, sehingga *navigation* (menu) ditambahkan pada *header* yang nantinya akan digunakan pada tiap halaman antarmuka. *Header* ini nantinya ada dimuat dalam template layout/base/template.jag.

Gambar 5.9 Antarmuka *Header* Awal



Gambar 5.10 *Left Navigation* Web API Store

Gambar 5.11 Antarmuka *Header* Hasil Kustomisasi

5.6.2.2 Kustomisasi Antarmuka *Footer*

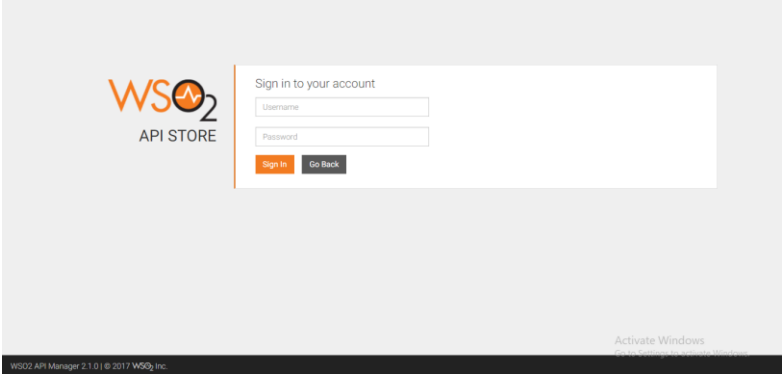
Footer ini berada di dalam halaman dasar pada halaman web, sehingga ketika halaman dasar ini digunakan maka di dalamnya telah terdapat *footer*. *Footer* ini nantinya akan dimuat dalam page/base/template.jag dan akan menjadi halaman dasar pada antarmuka web.

Gambar 5.12 Antarmuka *Footer* Awal

Gambar 5.13 Antarmuka *Footer* Hasil Kustomisasi

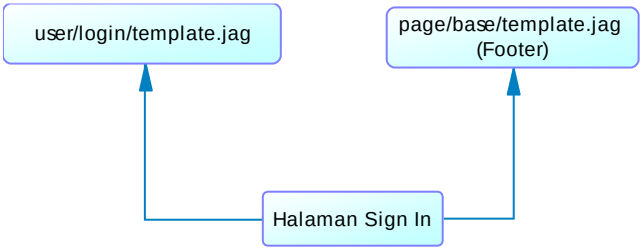
5.6.2.3 Kustomisasi Halaman Antarmuka *Sign In*

Halaman *sign in* ini berfungsi agar pengguna dapat masuk meggunakan username dan kata sandi pengguna. Gambar 5.14 merupakan antarmuka awal sebelum dilakukan kustomisasi.



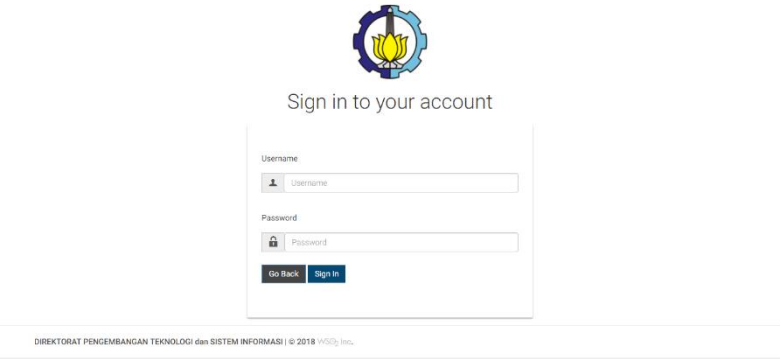
Gambar 5.14 Halaman Antarmuka *Sign In* Awal

Untuk menampilkan halaman *sign in* ini, diperlukan beberapa *template* yang berkaitan seperti pada Gambar 5.15. Pada halaman *sign in* ini berisi halaman dasar, yang di dalamnya berisi template dari halaman sign in yang terdapat pada **user\login\template.jag**.



Gambar 5.15 Keterkaitan Halaman Antarmuka *Sign In*

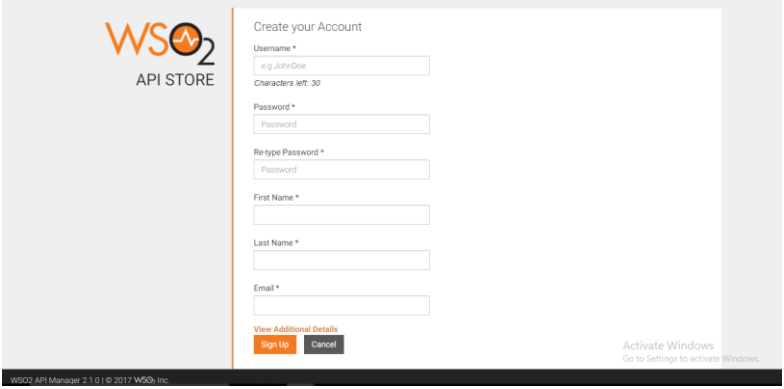
Hasil kustomisasi halaman antarmuka ini dapat dilihat pada Gambar 5.16.



Gambar 5.16 Halaman Antarmuka *Sign In* Hasil Kustomisasi

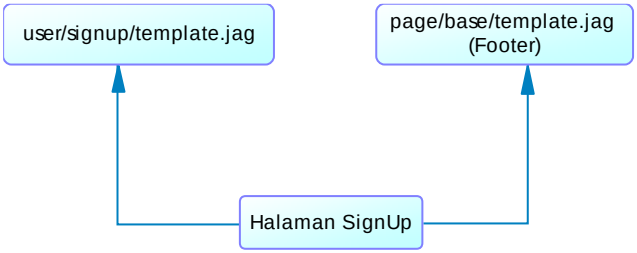
5.6.2.4 Kustomisasi Halaman Antarmuka *Sign Up*

Pada halaman *sign up* ini pengguna dapat menambahkan akun baru. Gambar 5.17 merupakan antarmuka *sign up* awal sebelum dilakukan kustomisasi.

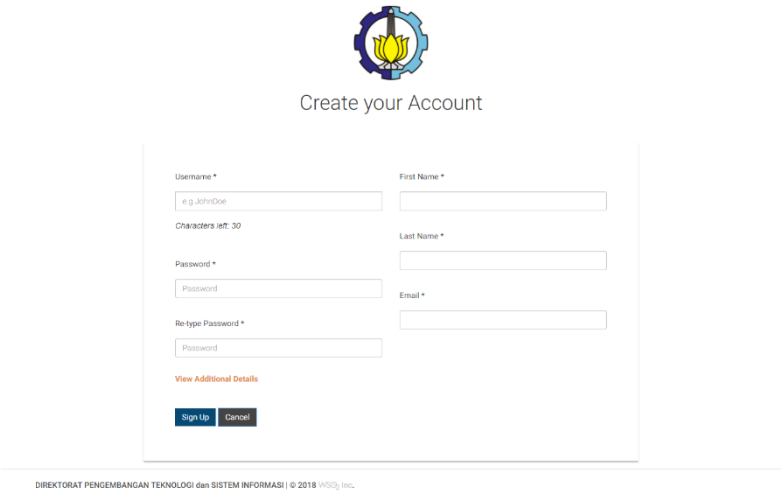


Gambar 5.17 Halaman Antarmuka *Sign Up* Awal

Untuk menampilkan halaman *sign up* ini diperlukan beberapa *template* yang berkaitan seperti pada Gambar 5.18. Pada halaman *sign up* ini berisi halaman dasar, yang di dalamnya berisi template dari halaman *sign up* yang terdapat pada **user\signup\template.jag**.



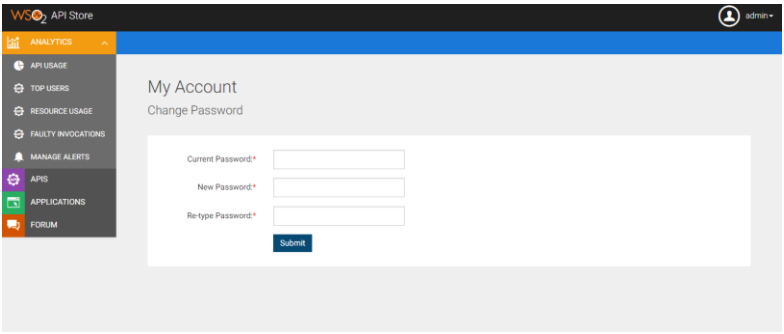
Gambar 5.18 Keterkaitan Halaman Antarmuka *Sign Up*



Gambar 5.19 Halaman Antarmuka *Sign Up* Hasil Kustomisasi

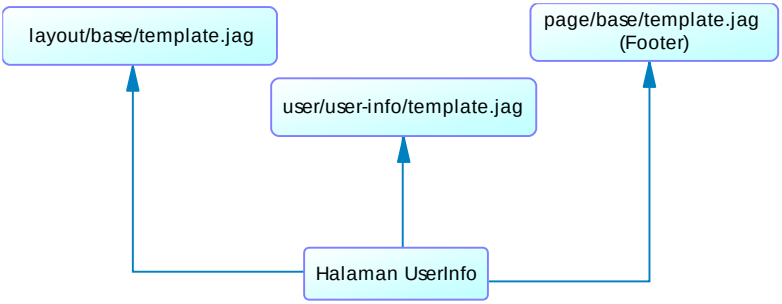
5.6.2.5 Kustomisasi Halaman Antarmuka *User Info*

Halaman user info ini berguna untuk apabila pengguna ingin mengubah kata sandi (*password*). Antarmuka awal dari halaman ini dapat dilihat pada gambar Gambar 5.20.

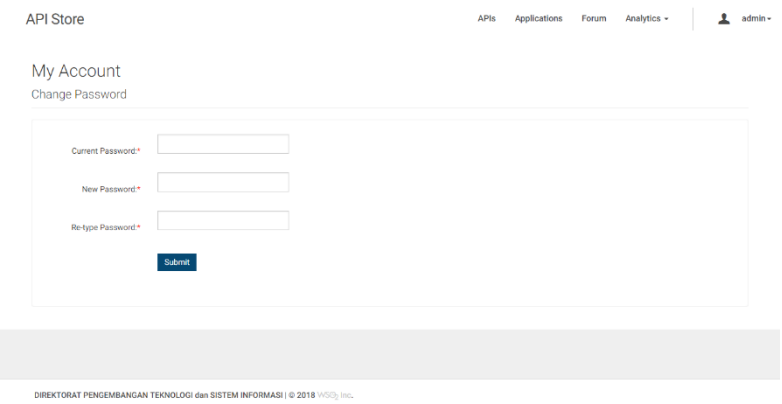


Gambar 5.20 Halaman Antarmuka *User Info* Awal

Untuk menampilkan halaman *user info* ini diperlukan beberapa *template* yang berkaitan seperti pada Gambar 5.21. Halaman ini terdiri dari user/user-info/template.jag yang merupakan *body* dari halaman. user/user-info/template.jag itu akan dimuat di dalam layout/base/template.jag. layout/base/template.jag berisi header yang nantinya akan dimuat di page/base/template.jag yang merupakan halaman dasar berisi footer. Hasil kustomisasi dari halaman ini dapat dilihat pada Gambar 5.22.



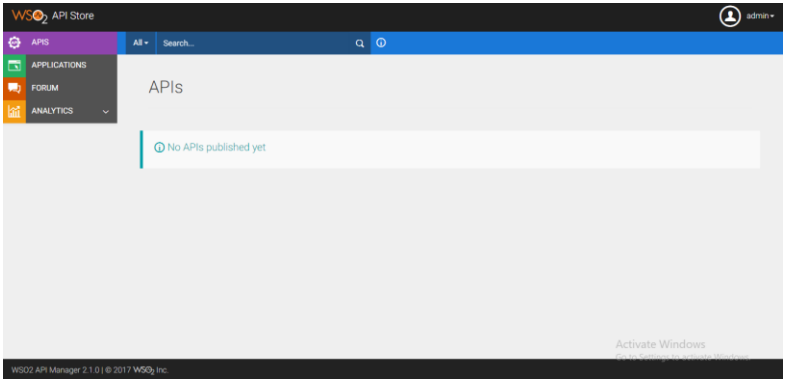
Gambar 5.21 Keterkaitan Halaman Antarmuka *User Info*



Gambar 5.22 Halaman Antarmuka *User Info* Hasil Kustomisasi

5.6.2.6 Kustomisasi Halaman Antarmuka APIs

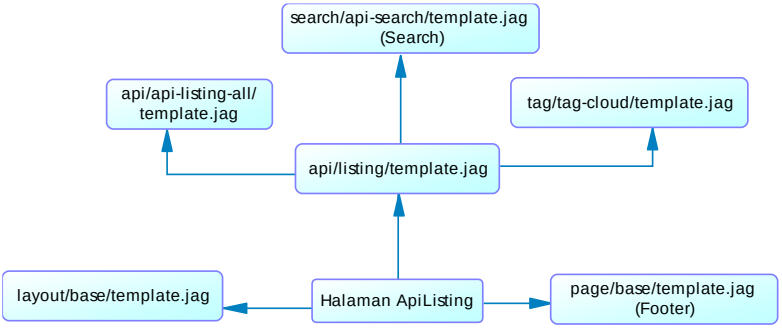
Halaman APIs ini berisi daftar API yang telah diterbitkan. Gambar 5.23 merupakan halaman awal sebelum dilakukan kustomisasi.



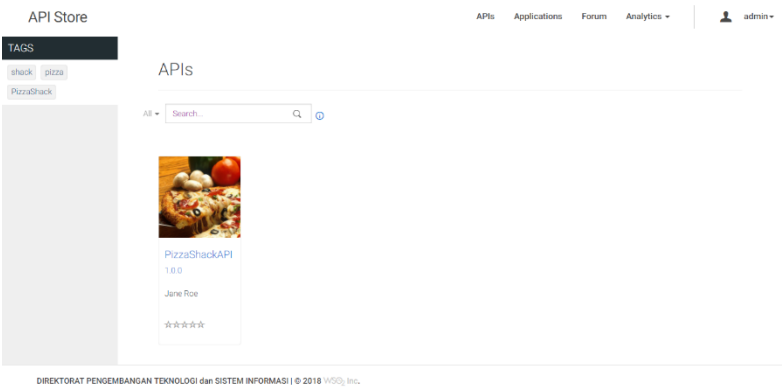
Gambar 5.23 Halaman Antarmuka APIs Awal

Untuk menampilkan halaman APIs ini diperlukan beberapa *template* yang berkaitan seperti pada Gambar 5.24. Halaman ini terdiri dari api/listing/template.jag, layout/base/template.jag, dan page/base/template.jag.

api/listing/template.jag yang merupakan *body* dari halaman terdiri dari beberapa *template*, yaitu api/api-listing-all/template.jag yang berisi daftar dari API, search/api-search/template.jag yang berisi *search bar*, dan tag/tag-cloud/template.jag yang berisi *tag*. api/api-listing/template.jag itu akan dimuat di dalam layout/base/template.jag. layout/base/template.jag berisi header yang nantinya akan dimuat di page/base/template.jag yang merupakan halaman dasar berisi footer. Hasil kustomisasi dari halaman ini dapat dilihat pada Gambar 5.24.



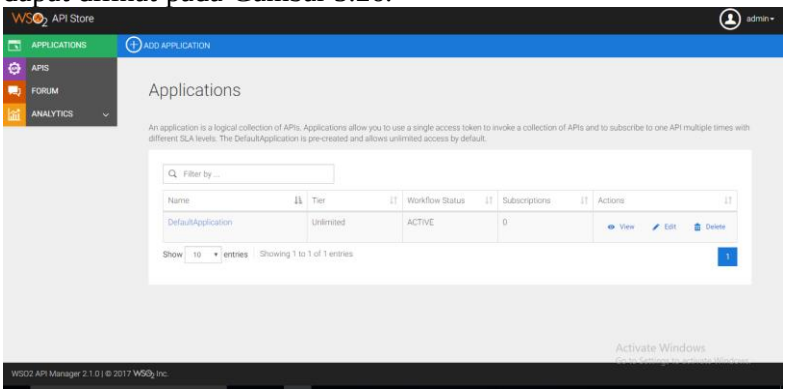
Gambar 5.24 Keterkaitan Halaman Antarmuka APIs



Gambar 5.25 Halaman Antarmuka APIs Hasil Kustomisasi

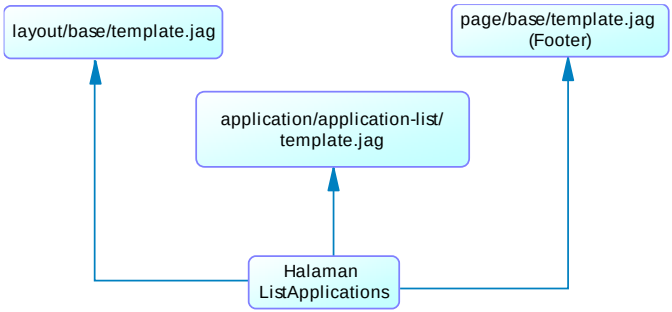
5.6.2.7 Kustomisasi Halaman Antarmuka *List Application*

Halaman ini berisi daftar dari *application*. Halaman antarmuka dari *list application* sebelum dilakukan kustomisasi dapat dilihat pada Gambar 5.26.

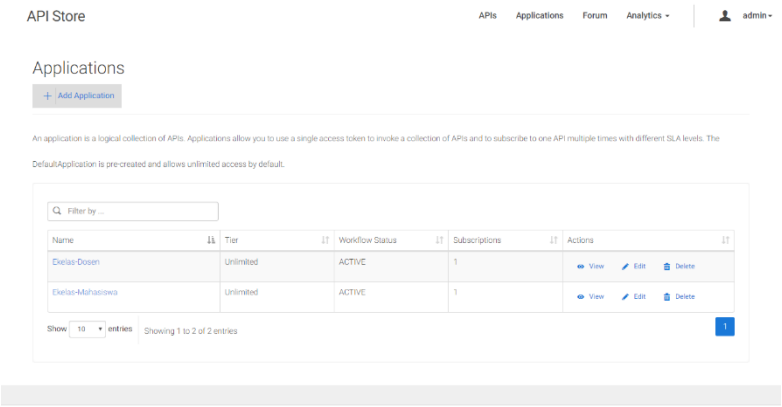


Gambar 5.26 Halaman Antarmuka *List Application* Awal

Untuk menampilkan halaman *list application*ini diperlukan beberapa *template* yang berkaitan seperti pada Gambar 5.27. Halaman ini terdiri dari application/application-list/template.jag yang merupakan *body* dari halaman yang berisi daftar dari *application*. application/application-list/template.jag nantinya akan dimuat di dalam layout/base/template.jag. layout/base/template.jag berisi *header* yang akan dimuat di page/base/template.jag yang merupakan halaman dasar berisi *footer*. Hasil kustomisasi dari halaman ini dapat dilihat pada Gambar 5.28.



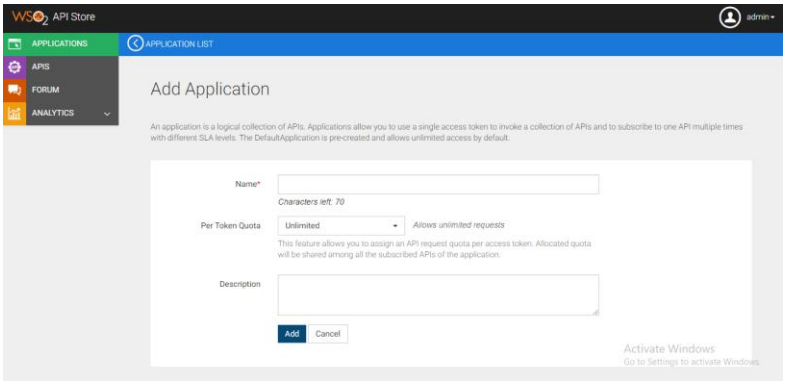
Gambar 5.27 Keterkaitan Halaman Antarmuka *List Application*



Gambar 5.28 Halaman Antarmuka *List Application* Hasil Kustomisasi

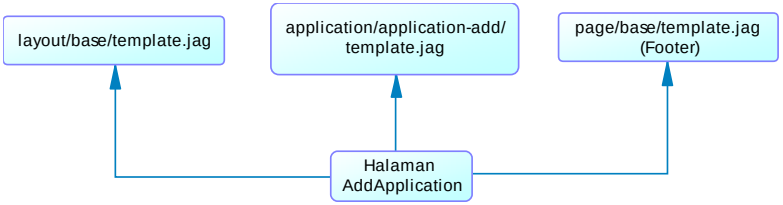
5.6.2.8 Kustomisasi Halaman Antarmuka *Add Application*

Halaman add application ini berfungsi untuk menambahkan *application* baru. Antarmuka halaman add application sebelum dilakukan kustomisasi dapat dilihat pada Gambar 5.29.

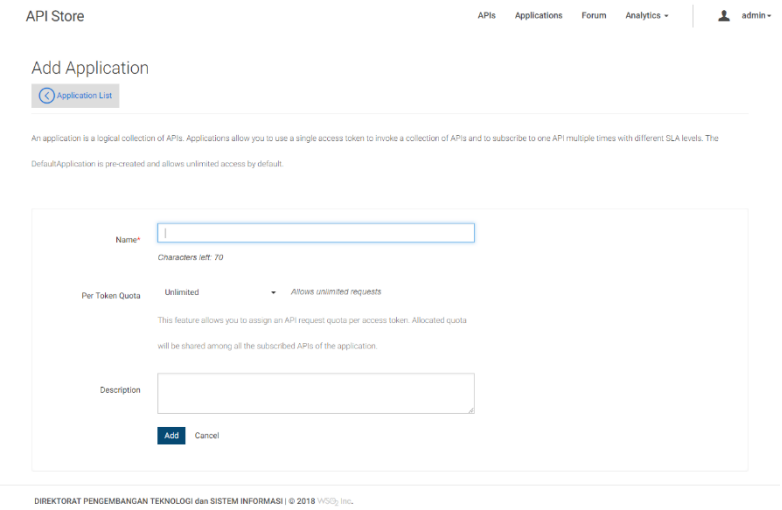


Gambar 5.29 Halaman Antarmuka *Add Application* Awal

Untuk menampilkan halaman *add application* ini diperlukan beberapa *template* yang berkaitan seperti padaGambar 5.30. Halaman ini terdiri dari layout/base/template.jag, page/base/template.jag, dan application/application-add/template.jag yang merupakan *body* dari halaman yang berisi form penambahan *application*. application/application-add/template.jag itu akan dimuat di dalam layout/base/template.jag. layout/base/template.jag berisi header yang nantinya akan dimuat di page/base/template.jag yang merupakan halaman dasar berisi footer. Hasil kustomisasi dari halaman ini dapat dilihat pada Gambar 5.31.



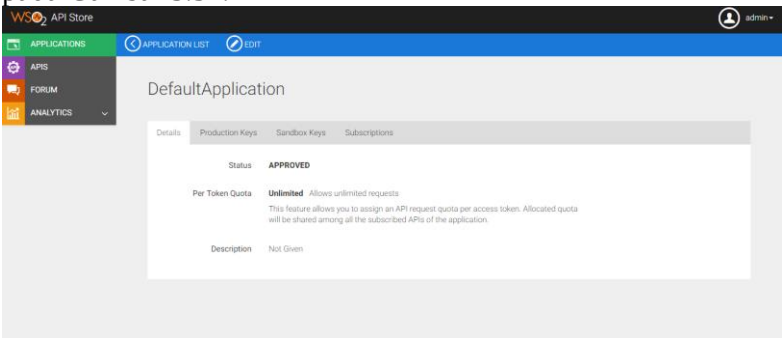
Gambar 5.30 Keterkaitan Halaman Antarmuka *Add Application*



Gambar 5.31 Halaman Antarmuka *Add Application* Hasil Kustomisasi

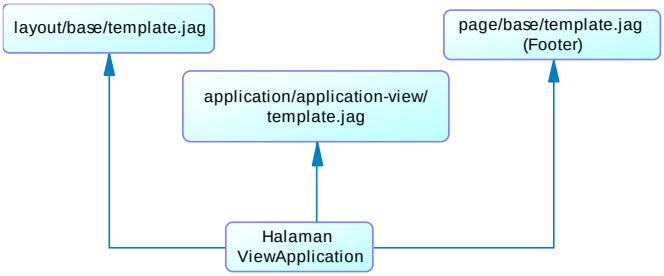
5.6.2.9 Kustomisasi Halaman Antarmuka *View Application*

Halaman ini berfungsi untuk menamppilkan rincian informasi dari *application* yang dipilih oleh pengguna. Antarmuka awal halaman *view application* ini dapat dilihat pada Gambar 5.32.

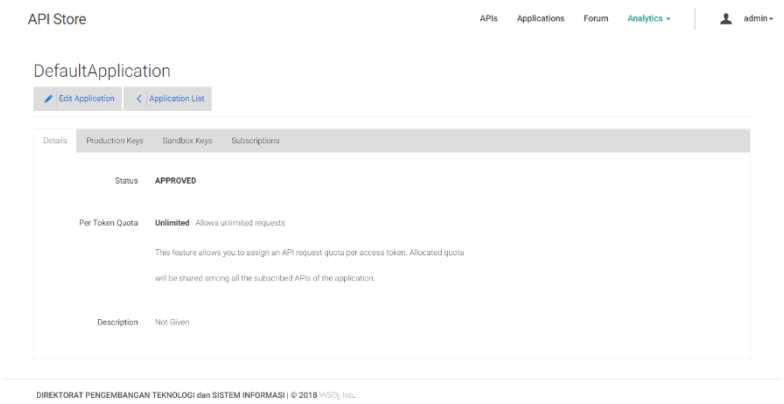


Gambar 5.32 Halaman Antarmuka *View Application* Awal

Untuk menampilkan halaman *view application* ini diperlukan beberapa *template* yang berkaitan seperti pada Gambar 5.33. Halaman ini terdiri dari layout/base/template.jag, page/base/template.jag, dan application/application-view/template.jag. application/application-view/template.jag merupakan *body* dari halaman yang akan menampilkan detail informasi *application* dan akan dimuat di dalam layout/base/template.jag. layout/base/template.jag berisi header yang nantinya akan dimuat di page/base/template.jag yang merupakan halaman dasar berisi footer. Hasil kustomisasi dari halaman ini dapat dilihat pada Gambar 5.31.



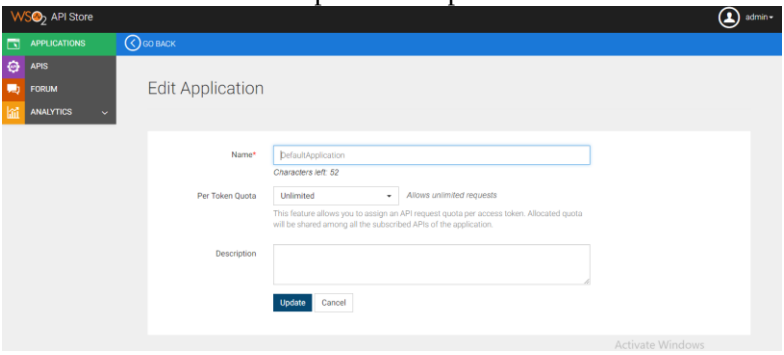
Gambar 5.33 Keterkaitan Halaman Antarmuka *View Application*



Gambar 5.34 Halaman Antarmuka *View Application* Hasil Kustomisasi

5.6.2.10 Kustomisasi Halaman Antarmuka *Edit Application*

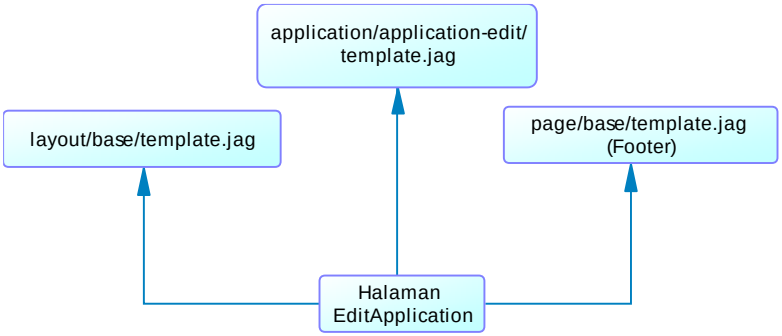
Halaman ini berfungsi untuk mengubah *application* yang telah ada. Halaman antarmuka *edit application* sebelum dilakukan kustomisasi dapat dilihat pada Gambar 5.35.



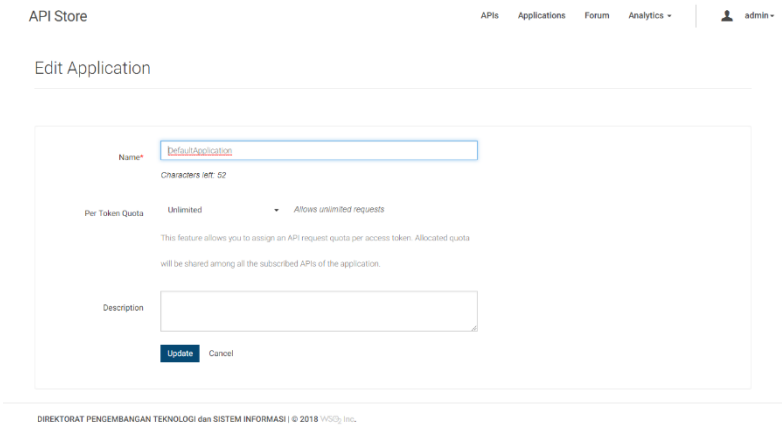
Gambar 5.35 Halaman Antarmuka Edit *Application* Awal

Untuk menampilkan halaman *edit application* ini diperlukan beberapa *template* yang berkaitan seperti pada Gambar 5.36. Halaman ini terdiri dari layout/base/template.jag, page/base/template.jag, dan application/application-

edit/template.jag. application/application-edit/template.jag merupakan *body* dari halaman yang akan menampilkan form untuk mengubah detail *application* dan akan dimuat di dalam layout/base/template.jag. layout/base/template.jag berisi header yang nantinya akan dimuat di page/base/template.jag yang merupakan halaman dasar berisi footer. Hasil kustomisasi dari halaman ini dapat dilihat pada Gambar 5.37Gambar 5.31.



Gambar 5.36 Keterkaitan Halaman Antarmuka Edit *Application*



Gambar 5.37 Halaman Antarmuka Edit *Application* Hasil Kustomisasi

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pengujian terhadap sistem yang telah dikembangkan dari implementasi yang telah dilakukan pada bab sebelumnya.

6.1 Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan pada lingkungan dan alat kakas sebagai berikut

Tabel 6.1 Lingkungan Pengujian

Spesifikasi	Deskripsi
Jenis Perangkat	Komputer
Merek Perangkat	DELL Inspiron 14 3000 Series
Sistem Operasi	Windows 10 Pro 64-bit
RAM	4 GB
Tools	Postman
Browser	Google Chrome Version 67.0

6.2 Pengujian

Pada subbab ini akan dijelaskan beberapa pengujian yang dilakukan terhadap sistem yang telah dibangun.

6.2.1 Pengujian Sistem Keamanan

Pada pengujian sekuriti ini, terdapat beberapa skenario pengujian seperti pada Tabel 6.2.

Tabel 6.2 Skenario Pengujian Sistem Keamanan

Kode Skenario	Skenario
S-1	Memasukkan username atau kata sandi yang tidak sesuai
S-2	Pengguna mempunyai peran yang tidak sesuai
S-3	IP *request* dari pengguna tidak mempunyai hak
S-4	Username, kata sandi, peran, dan IP request sesuai.

Hasil dari pengujian Skenario S-1 dapat dilihat pada Gambar 6.1. Dapat dilihat pada gambar tersebut bahwa *respon-code* yang

ditampilkan adalah 401, yang berarti statusnya *unauthorized*, karena proses otentikasi (username dan kata sandi salah) yang dilakukan tidak benar.

Gambar 6.1 Hasil Pengujian Skenario S-1

Hasil dari pengujian skenario S-2 dapat dilihat pada Gambar 6.2. Gambar tersebut menampilkan *respon-code* 403, yang berarti statusnya *forbidden* karena pengguna tidak mempunyai hak akses atau peran yang sesuai.

Gambar 6.2 Hasil Pengujian Skenario S-2

Hasil dari pengujian skenario S-3 dapat dilihat pada Gambar 6.3. Pada gambar tersebut menampilkan *respon-code* 401, yang berarti statusnya *unauthorized*. Hal ini berarti IP *request* dari pengguna tidak mempunyai hak akses.

Gambar 6.3 Hasil Pengujian Skenario S-3

Hasil dari pengujian skenario S-4 dapat dilihat pada Gambar 6.4. Pada gambar tersebut menampilkan *respon-code* 200, yang berarti statusnya *OK*. Hal ini berarti proses otentikasi, otorisasi, dan IP sudah sesuai.

Gambar 6.4 Hasil Pengujian Skenario S-4

6.2.2 Pengujian Manajemen API

Pada subbab ini akan dilakukan beberapa pengujian terhadap aplikasi manajemen API. Pada pengujian ini menggunakan API untuk sistem e-kelas yang sudah terdapat di aplikasi manajemen API.

6.2.2.1 Mengambil List Diskusi pada Forum

Detail pengujian mengambil list diskusi pada forum dijelaskan pada Tabel 6.3. Hasil dari pengujian terdapat pada

Gambar 6.5. Respon dari *request endpoint* ini merupakan list yang berisi id, *course*, forum, *name*, *message*, *first post*, *username*, *group id*, *assessed*, *time modified*, *user modified*, dan *replies*.

Tabel 6.3 Detail Pengujian Mengambil List Diskusi ada Forum

Endpoint (api.its.ac.id/ekelas/ 1.0)	***Method* HTTP**	**Hasil**
/forum/{id}/discussi--on	GET	Mengambil list diskusi pada forum

```
[
    {
        "id": 30,
        "course": 2,
        "forum": 2,
        "name": "zad",
        "message": "ttgg",
        "first_post": 65,
        "username": "WILLIAM SUHUD",
        "group_id": -1,
        "assessed": 0,
        "time_sort": 1527077303,
        "time_modified": 1527077303,
        "user_modified": 3,
        "replies": 1
    },
    {
        "id": 29,
        "course": 2,
        "forum": 2,
        "name": "y",
        "message": "y",
```

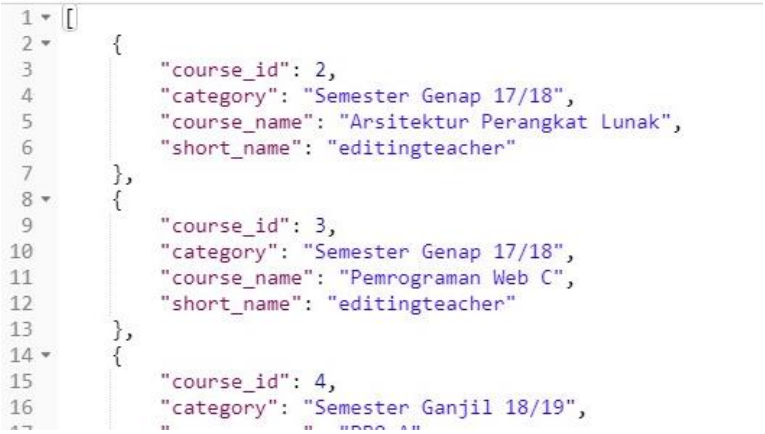

Gambar 6.5 Hasil Pengujian Mengambil List Diskusi ada Forum

6.2.2.2 Mengambil Data Kelas

Detail pengujian mengambil data kelas dijelaskan pada Tabel 6.4. Hasil dari pengujian terdapat pada Gambar 6.6. Respon dari *request endpoint* ini merupakan list yang berisi *course id*, *category*, *course name*, dan *short name*.

Tabel 6.4 Detail Pengujian Megambil Data Kelas

Endpoint (api.its.ac.id/ekelas/ 1.0)	*Method* HTTP	Hasil
/listcourse	GET	Mengambil detil data kelas



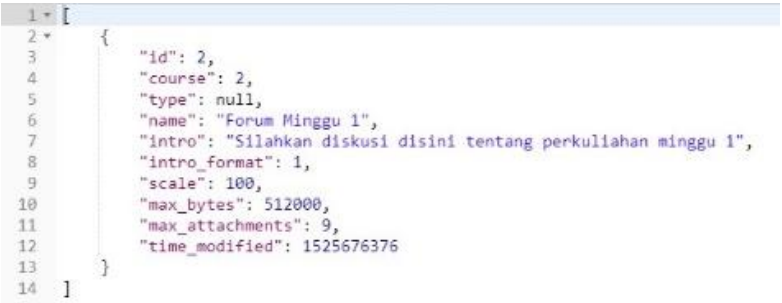
Gambar 6.6 Hasil Pengujian Mengambil Data Kelas

6.2.2.3 Mengambil Detail Forum

Detail pengujian mengambil detail forum dijelaskan pada Tabel 6.5. Hasil dari pengujian tersebut terdapat pada Gambar 6.7. Respon dari request endpoint ini berisi id, *course*, *type*, *name*, *intro*, *intro format*, *scale*, *max bytes*, *max attachments*, dan *time modified*.

Tabel 6.5 Detail Pengujian Mengambil Detail Forum

Endpoint (api.its.ac.id/ekelas/ 1.0)	*Method* HTTP	Hasil
/forum/{id}	GET	Mengambil detail forum



```
[
    {
        "id": 2,
        "course": 2,
        "type": null,
        "name": "Forum Minggu 1",
        "intro": "Silahkan diskusi disini tentang perkuliahan minggu 1",
        "intro_format": 1,
        "scale": 100,
        "max_bytes": 512000,
        "max_attachments": 9,
        "time_modified": 1525676376
    }
]
```


Gambar 6.7 Hasil Pengujian Mengambil Detail Forum

6.2.2.4 Mengambil Detail Kuis

Detail pengujian mengambil detail kuis dijelaskan pada tabel Tabel 6.6. Hasil dari pengujian tersebut terdapat pada Gambar 6.8. Respon dari rquest endpoint tersebut berisi *id*, *course*, *name*, *intro*, *intro format* dan lain sebagainya.

Tabel 6.6 Detail Pengujian Mengambil Detail Kuis

Endpoint (api.its.ac.id/ekelas/ 1.0)	***Method* HTTP**	**Hasil**
/quiz/{id}	GET	Mengambil detail kuis

```
[
    {
        "id": 1,
        "course": 2,
        "name": "Pre-Test Minggu 1",
        "intro": "<p>Pre-Test awal APL</p>",
        "intro_format": 1,
        "time_open": 0,
        "time_close": 0,
        "time_limit": 60,
        "overdue_handling": "autosubmit",
        "grace_period": 0,
        "preferred_behaviour": "deferredfeedback",
        "can_redo_questions": 0,
        "attempts": 1,
        "attempt_on_last": 0,
        "grade_method": 1,
        "decimal_points": 2,
        "question_decimal_points": -1,
        "review_attempt": 69904,
        "review_correctness": 4368,
        "review_marks": 4368,
```


Gambar 6.8 Hasil Pengujian Mengambil Detail Kuis

6.3 Evaluasi Pengujian

Pada subbab ini akan dijelaskan evaluasi dari pengujia sistem keamanan dan juga evaluasi tema antarmuka WSO2 API Store yang baru.

6.3.1 Evaluasi Pengujian Sistem Keamanan

Rangkuman mengenai hasil pengujian sistem keamanan yang diujicobakan dapat dilihat pada Tabel 6.7.

Tabel 6.7 Evaluasi Pengujian Sistem Keamaan

Kode Skenario	Terpenuhi
S-1	√
S-2	√
S-3	√
S-4	√

Berdasarkan data pada Tabel 6.7, seluruh skenario pengujian berhasil dilakukan. Sehingga, dapat disimpulkan bahwa sistem keamanan sesuai yang diharapkan

6.3.2 Evaluasi Tema

Pengujian ini dilakukan dengan membagikan kuisioner kepada 10 partisipan. Kuisioner ini terdiri dari bagian, bagian pertama untuk tema awal dan bagian kedua untuk tema setelah kustomisasi. Rekapitulasi penilaian pengujian terhadap antarmuka pengguna pada tema awal terdapat pada Tabel 6.8. Sedangkan untk tema hasil kustomisasi terdapat pada Tabel 6.9. Daftar pertanyaan penilaian antarmuka untuk tema awal dan setelah kustomisasi, adalah sebagai berikut:

1. Apakah antarmuka aplikasi sudah sesuai dengan fungsionalitas yang digunakan?
2. Apakah masing-masing elemen antarmuka mudah dipahami kegunaannya?
3. Apakah informasi yang diberikan oleh aplikasi mudah dipahami?
4. Apakah aplikasi sudah memiliki tata letak yang baik dan efektif?
5. Apakah antarmuka aplikasi memudahkan Anda untuk menyelesaikan pekerjaan?

Tabel 6.8 Rekapitulasi Penilaian Antarmuka Tema Awal

Pertanyaan ke-	Penilaian										Rata-rata
	1	2	3	4	5	6	7	8	9	10	
1	0	0	0	0	1	2	4	3	0	0	7,1
2	0	0	0	0	1	7	2	0	0	0	6,1
3	0	0	0	0	1	6	3	0	0	0	6,2
4	0	0	0	0	5	4	0	1	0	0	5,7
5	0	0	0	1	1	3	4	1	0	0	6,3
Nilai Akhir											6,28

Tabel 6.9 Rekapitulasi Penilaian Antarmuka Tema Hasil Kustomisasi

Pertanyaan ke-	Penilaian										Rata-rata
	1	2	3	4	5	6	7	8	9	10	
1	0	0	0	0	0	0	0	7	3	0	8,3
2	0	0	0	0	0	0	1	7	2	0	8,1

Pertanyaan ke-	Penilaian										Rata-rata
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
3	0	0	0	0	0	0	0	8	2	0	8,2
4	0	0	0	0	0	0	0	5	5	0	8,5
5	0	0	0	0	0	0	0	4	6	0	8,6
Nilai Akhir											8,34

Pada Tabel 6.9 dapat dilihat bahwa nilai akhir dari tema hasil kustomisasi mendapat nilai 8,34 dan lebih tinggi dari nilai tema awal yang mendapat nilai 6,28.

{Halaman ini sengaja dikosongkan}

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh selama pengerjaan tugas akhir dan saran mengenai pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

7.1 Kesimpulan

Berikut merupakan kesimpulan yang dapat diambil dari proses pengembangan dan hasil uji coba.

1. Telah didapatkan standar baku dalam penamaan *resource* pada REST *service* yang nantinya dapat digunakan dalam pegembangan API di ITS.
2. Sistem keamanan yang dibangun telah berhasil mengimplementasikan sistem otentikasi menggunakan OAuth2, otorisasi menggunakan JWT, dan pembatasan akses berdasarkan IP *request*.
3. Kustomisasi API Store telah berhasil dilakukan dengan menggunakan kerangka kerja Jaggery *web app*, hasil kustomisasi menghasilkan tema yang diimplementasikan pada api.its.ac.id, dan dari nilai akhir yang didapatkan para pengguna lebih menyukai tema dari hasil kustomisasi.
4. Dengan adanya sistem ini pengembang juga dapat dengan mudah mengelola siklus hidup dari API ini, memberikan layanan jauh lebih cepat, serta sistem yang lebih aman.

7.2 Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian yang telah dilakukan.

1. Dapat dilakukan kustomisasi pada antarmuka API Store yang lebih baik lagi untuk *icon* dan juga untuk pengaturan pada perangkat bergerak.

{Halaman ini sengaja dikosongkan}

DAFTAR PUSTAKA

[1] Feridi, "Mengenal RESTful Web Services," *CodePolitan.com*. [Online]. Available: https://www.codepolitan.com/mengenal-restful-web-services. [Accessed: 05-Jun-2017].

[2] "What is API management? - Definition from WhatIs.com," *SearchMicroservices*. [Online]. Available: http://searchmicroservices.techtarget.com/definition/API-management. [Accessed: 07-Jun-2017].

[3] "Quick Start Guide - API Manager 2.0.0 - WSO2 Documentation." [Online]. Available: https://docs.wso2.com/display/AM200/Quick+Start+Guide. [Accessed: 28-May-2018].

[4] R. Mufrizal, "Belajar OAuth2," *Rizki Mufrizal*. [Online]. Available: https://rizkimufrizal.github.io/belajar-oauth2/. [Accessed: 05-Jun-2017].

[5] auth0.com, "JWT.IO - JSON Web Tokens Introduction." [Online]. Available: http://jwt.io/. [Accessed: 06-Jun-2017].

[6] "Maven – POM Reference." [Online]. Available: https://maven.apache.org/pom.html. [Accessed: 28-May-2018].

[7] "Maven – Introduction to the Dependency Mechanism." [Online]. Available: https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html. [Accessed: 31-May-2018].

[8] "Overview - API Manager 2.1.0 - WSO2 Documentation." [Online]. Available: https://docs.wso2.com/display/AM210/Overview. [Accessed: 05-Jun-2017].

[9] "OAuth 2.0 — OAuth." [Online]. Available: https://oauth.net/2/. [Accessed: 05-Jun-2017].

[10] "Writing a Custom User Store Manager - Identity Server 5.0.0 - WSO2 Documentation." [Online]. Available: https://docs.wso2.com/display/IS500/Writing+a+Custom+User+Store+Manager. [Accessed: 27-May-2018].

[11]“Customizing WSO2 API Manager API Store and Publisher - Part 1.” [Online]. Available: https://wso2.com/library/tutorials/2012/09/customizing-api-store-publisher-part1/. [Accessed: 30-May-2018].

BIODATA PENULIS

Raras Anggita, anak ketiga dari tiga bersaudara yang lahir di Probolinggo pada tanggal 4 Oktober 1995. Penulis telah menempuh pendidikan formal mulai dari SD Katolik Mater Dei Probolinggo (2002-2008), SMP Katolik Mater Dei Probolinggo (2008-2011), SMA Negeri 2 Lumajang (2011-2014) dan terakhir sebagai mahasiswa Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember dengan rumpun mata kuliah Rekayasa Perangkat Lunak (2014-2018).

Lulus dari SMA penulis melanjutkan pendidikan di jurusan teknik informatika. Semasa kuliah penulis aktif mengikuti berbagai kepanitiaan diantaranya Schematics 2015 dan Schematics 2016. Penulis juga aktif menjadi anggota Himpunan Mahasiswa Teknik Computer-Informatika pada tahun 2015 dan menjadi sekertaris Departemen Minat Bakat Himpunan Mahasiswa Teknik Computer-Informatika pada tahun 2016. Penulis dapat dihubungi melalui email theresararas@gmail.com