TUGAS AKHIR – IS184853

PERANCANGAN DAN PENGEMBANGAN APLIKASI *BACKEND* UNTUK SISTEM MANAJEMEN VENDOR UMKM GIYOMI.ID MENGGUNAKAN SOA SPRING BOOT

DESIGN AND DEVELOPMENT OF BACKEND APPLICATION FOR MSME GIYOMI.ID VENDOR MANAGEMENT SYSTEM USING SOA SPRING BOOT

ETIKA PAHANG KRISDYAN
NRP 05211640000056

Dosen Pembimbing I
Mahendrawathi E.R., S.T., M.Sc., Ph. D.
Dosen Pembimbing II
Rully Agus Hendrawan, M.Eng.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – IS184853

PERANCANGAN DAN PENGEMBANGAN APLIKASI *BACKEND* UNTUK SISTEM MANAJEMEN VENDOR UMKM GIYOMI.ID MENGGUNAKAN SOA SPRING BOOT

ETIKA PAHANG KRISDYAN
NRP 05211640000056

Dosen Pembimbing I
Mahendrawathi E.R., S.T., M.Sc., Ph. D.
Dosen Pembimbing II
Rully Agus Hendrawan, M.Eng.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

Halaman ini sengaja dikosongkan

FINAL PROJECT – IS184853

DESIGN AND DEVELOPMENT OF BACKEND APPLICATION FOR MSME GIYOMI.ID VENDOR MANAGEMENT SYSTEM USING SOA SPRING BOOT

ETIKA PAHANG KRISDYAN
NRP 05211640000056

Supervisor I
Mahendrawathi E.R., S.T., M.Sc., Ph. D.
Supervisor II
Rully Agus Hendrawan, M.Eng.

DEPARTMENT OF INFORMATION SYSTEMS
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology
Surabaya 2020

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

PERANCANGAN DAN PENGEMBANGAN APLIKASI *BACKEND* UNTUK SISTEM MANAJEMEN VENDOR UMKM GIYOMI.ID MENGGUNAKAN SOA SPRING BOOT

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat Memperoleh
Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:

ETIKA PAHANG KRISDYAN
NRP. 05211640000056

Surabaya, 18 Agustus 2020

KEPALA
DEPARTEMEN SISTEM INFORMASI

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN INSTITUT TEKNOLOGI SEPULUH NOPEMBER FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS DEPARTEMEN SISTEM INFORMASI

Dr. Mudjahidin, S.T., M.T.
NIP. 19701010 200312 1 001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

PERANCANGAN DAN PENGEMBANGAN APLIKASI *BACKEND* UNTUK SISTEM MANAJEMEN VENDOR UMKM GIYOMI.ID MENGGUNAKAN SOA SPRING BOOT

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:

ETIKA PAHANG KRISDYAN
NRP. 05211640000056

Disetujui Tim Penguji : Tanggal Ujian :
Periode Wisuda : September 2020

Mahendrawathi E.R., S.T., M.Sc., Ph. D. **(Pembimbing I)**

Rully Agus Hendrawan, M.Eng. **(Pembimbing II)**

Erma Suryani, ST., MT., Ph.D. **(Penguji I)**

Andre Parvian Aristio, S.Kom., M.Sc. **(Penguji II)**

Halaman ini sengaja dikosongkan

PERANCANGAN DAN PENGEMBANGAN APLIKASI *BACKEND* UNTUK SISTEM MANAJEMEN VENDOR UMKM GIYOMI.ID MENGGUNAKAN SOA SPRING BOOT

Nama Mahasiswa	**:**	**Etika Pahang Krisdyan**
NRP	**:**	**05211640000056**
Departemen	**:**	**Sistem Informasi**
Dosen Pembimbing I	**:**	**Mahendrawathi E.R., S.T., M.Sc., Ph. D.**
Dosen Pembimbing II	**:**	**Rully Agus Hendrawan, M.Eng.**

ABSTRAK

Giyomi.id merupakan UMKM di bidang garmen. Giyomi.id bekerjasama dengan vendor untuk proses produksi garmennya. Ketergantungan Giyomi.id kepada vendor membuat pengawasan dan dokumentasi performa vendor menjadi sangat penting. Kesalahan dalam memilih vendor yang performanya tidak mencapai target, disebabkan oleh metode pengawasan konvensional sehingga dapat berdampak pada keterlambatan proses produksi dan penjualan sampai menciptakan kerugian finansial bagi Giyomi.id.

Bening 2019 melakukan penelitian dengan luaran berupa prototipe sistem manajemen vendor Giyomi.id, sehingga dibutuhkan adanya penelitian lanjutan mengenai pengembangan aplikasi sistem manajemen vendor tersebut. Melihat permasalahan pada pengawasan vendor Giyomi.id dan hasil penelitian yang dilakukan oleh Bening pada tahun 2019, maka dikembangkan sebuah aplikasi backend sistem manajemen vendor beserta dokumentasi aplikasi backend. Aplikasi backend dikembangkan melalui kerangka kerja Spring

Boot yang mendukung sistem untuk menggunakan arsitektur SOA dan didokumentasikan menggunakan Swagger. Pengembangan aplikasi backend menggunakan Waterfall Software Development Life Cycle (SDLC) agar pengembangan dapat berjalan tepat waktu.

Hasil dari penelitian ini adalah aplikasi backend sistem manajemen vendor sesuai prototipe sistem manajemen vendor yang telah tersedia dengan fitur utama yaitu manajemen Surat Perintah Kerja (SPK), manajemen vendor dan manajemen akun pengguna. Penelitian ini juga menghasilkan dokumentasi sistem backend yang diperlukan untuk penelitian selanjutnya, yaitu pengembangan aplikasi frontend sistem manajemen vendor.

Kata kunci: Sistem Manajemen Vendor, Backend, SOA, Spring Boot, Swagger

DESIGN AND DEVELOPMENT OF BACKEND APPLICATION FOR MSME GIYOMI.ID VENDOR MANAGEMENT SYSTEM USING SOA SPRING BOOT

Name	**:**	**Etika Pahang Krisdyan**
NRP	**:**	**05211640000056**
Department	**:**	**Sistem Informasi**
Supervisor I	**:**	**Mahendrawathi E.R., S.T., M.Sc., Ph. D.**
Supervisor II	**:**	**Rully Agus Hendrawan, M.Eng.**

ABSTRACT

Giyomi.id is a MSME in the field of garment. Giyomi.id cooperates with vendors for the production process of garments. Giyomi.id's dependence on vendors makes monitoring and documentation of vendor performance very important. Errors in selecting vendors whose performance did not reach the target, due to conventional monitoring methods so that it can have an impact on delays in the production and sales process to create financial losses for Giyomi.id.

Bening 2019 conducts research with an output in the form of a prototype of the Giyomi.id vendor management system, so that further research is needed on the development of the vendor management system application. Seeing the problems with Giyomi.id vendor supervision and the results of research conducted by Bening in 2019, a vendor management system backend application was developed along with backend application documentation. The backend application was developed through the Spring Boot framework which supports the system to using SOA architecture and documentation with Swagger. Backend application development uses Waterfall Software Development Life Cycle (SDLC) so that development can run on time.

The results of this study are the vendor management system backend application that matches the prototype that has been available with the main features of Work Order Management (SPK), vendor management and user account management. This research also produces backend system documentation needed for further research, namely the development of frontend vendor management system applications.

Keywords: Vendor Management System, Backend, SOA, Spring Boot, Swagger

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini:

Nama	: Etika Pahang Krisdyan
NRP	: 05211640000056
Tempat/Tanggal lahir	: Pasuruan, 21 Mei 1998
Fakultas/Departemen	: ELECTICS/Sistem Informasi
Nomor Telp/Hp/email	: 081231684642/etika.pahang.k.1998@gmail.com

Dengan ini menyatakan dengan sesungguhnya bahwa penelitian/makalah/tugas akhir saya yang berjudul

PERANCANGAN DAN PENGEMBANGAN APLIKASI *BACKEND* UNTUK SISTEM MANAJEMEN VENDOR UMKM GIYOMI.ID MENGGUNAKAN SOA SPRING BOOT

Bebas Dari Plagiarisme Dan Bukan Hasil Karya Orang Lain.

Apabila dikemudian hari ditemukan seluruh atau sebagian penelitian/makalah/tugas akhir tersebut terdapat indikasi plagiarisme, maka saya bersedia menerima sanksi sesuai peraturan dan ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sesungguhnya dan untuk dipergunakan sebagaimana mestinya.

Surabaya, Agustus 2020

Etika Pahang Krisdyan
NRP.05211640000056

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Dengan mengucapkan rasa syukur kepada Tuhan Yang Maha Esa, akhirnya penulis dapat menyelesaikan laporan penelitian Tugas Akhir yang berjudul **Perancangan dan Pengembangan Aplikasi Backend Untuk Sistem Manajemen Vendor UMKM Giyomi.Id Menggunakan SOA Spring Boot** yang menjadi salah satu syarat kelulusan pada Departemen Sistem Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember Surabaya.

Penyelesaian Tugas Akhir ini, diiringi dengan berbagai bantuan dari berbagai pihak yang selalu memberi dukungan, saran, dan doa sehingga penelitian berjalan dengan lancar. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Kedua orang tua dan keluarga penulis, yang tidak pernah lelah memberikan dukungan, motivasi, doa serta nasihat yang tulus.
2. Ibu Mahendrawathi E.R., S.T., M.Sc., Ph. D. dan Bapak Rully Agus Hendrawan, M.Eng. selaku dosen pembimbing yang telah meluangkan banyak waktu untuk sabar membimbing serta memberikan arahan penulis dalam pengerjaan tugas akhir ini.
3. Ibu Erma Suryani, ST., MT., Ph.D. dan Bapak Andre Parvian Aristio, S.Kom., M.Sc. selaku dosen penguji yang telah memberikan kritik serta saran yang membangun dalam proses pengerjaan tugas akhir
4. Teman-teman Artemis yang memberikan semangat serta menemani penulis dari awal perkuliahan hingga menyelesaikan tugas akhir ini.
5. Teman-teman di Laboraotium SE, RDIB, MSI, dan ADDI yang telah menemani dan menjadi tempat diskusi penulis dalam pengerjaan tugas akhir saat di Lab.

6. Be yang selalu menemani dari awal hingga akhir, memberikan semangat, menjadi rekan 24/7 yang selalu ada, menghibur penulis, dan mendukung penulis dalam pengerjaan Tugas Akhir serta menjadi tempat penulis dalam berkeluh kesah.
7. Pihak Giyomi.id yang telah memfasilitasi penelitian sehingga penulis bisa menyelesaikan tugas akhir ini sesuai dengan tujuan.
8. Pihak lainnya yang berkontribusi dalam tugas akhir yang belum dapat penulis sebutkan satu per satu.

Terima kasih atas segala bentuk dukungan, bantuan, saran, dan doa yang sudah diberikan kepada penulis. Penulis menyadari bahwa Tugas Akhir ini masih jauh dari sempurna. Oleh karena itu, penulis menerima segala masukan, pertanyaan, dan kritikan yang membangun. Semoga penelitian tugas akhir dapat memberikan manfaat bagi para pembaca.

Surabaya, Juni 2020

Penulis

DAFTAR ISI

LEMBAR PENGESAHAN v
LEMBAR PERSETUJUAN vii
ABSTRAK ix
ABSTRACT xi
SURAT PERNYATAAN BEBAS PLAGIARISME xiii
KATA PENGANTAR xv
DAFTAR ISI xvii
DAFTAR GAMBAR xxi
DAFTAR TABEL xxiii
DAFTAR KODE xxv
BAB I PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Rumusan Masalah 4
1.3. Batasan Tugas Akhir 5
1.4. Tujuan Tugas Akhir 5
1.5. Manfaat Penelitian 5
1.6. Relevansi 6
BAB II KAJIAN PUSTAKA 7
2.1. Penelitian Sebelumnya 7
2.2. Otentikasi dan Otorisasi 12
2.3. Aplikasi Backend 12
2.4. Service Oriented Architecture 13
2.5. Layanan Web 13
2.6. Spring Boot 14
2.7. Software Development Life Cycle (SDLC) 14

2.7.1. SDLC Waterfall 14
2.8. Swagger 15
2.9. Bahasa Pemrograman Java 15
2.10. JUnit.....16
2.11. ReadyAPI.....16
2.12. IntelliJ IDEA.....17
BAB III METODOLOGI 19
3.1. Diagram Metodologi.....19
3.2. Uraian Metodologi.....21
3.2.1. Identifikasi dan Perumusan Masalah 21
3.2.2. Studi Literatur 21
3.2.3. Analisis 22
3.2.3.1. Analisis Kebutuhan Fungsional 22
3.2.3.2. Analisis Kebutuhan Layanan Web.....22
3.2.4. Perancangan 23
3.2.4.1. Perancangan Model Data 23
3.2.4.1.1. Perancangan Model Data Konseptual.....24
3.2.4.1.2. Perancangan Model Data Logis.....24
3.2.4.2. Perancangan Layanan Web.....24
3.2.4.3. Perancangan Kebijakan Otentikasi dan Otorisasi 24
3.2.5. Pengembangan 25
3.2.5.1. Pengembangan Model Data 25
3.2.5.2. Pengembangan Layanan 25
3.2.5.3. Pengembangan Kebijakan Otentikasi dan Otorisasi 26

3.2.5.4. Pengembangan Dokumentasi Aplikasi 26
3.2.6. Pengujian 26
3.2.6.1. Pengujian Unit 26
3.2.6.2. Pengujian Fungsional 27
3.2.6.3. Pengujian Performa 27
3.2.6.4. Pengujian Keamanan 27
3.2.7. Penyusunan Laporan Tugas Akhir 27
BAB IV ANALISIS DAN PERANCANGAN 29
4.1. Analisis 29
4.1.1. Analisis Kebutuhan Fungsional 29
4.1.2. Analisis Kebutuhan Layanan Web 30
4.2. Perancangan 31
4.2.1. Perancangan Model Data 31
4.2.1.1. Perancangan Model Data Konseptual 31
4.2.1.2. Perancangan Model Data Logis 33
4.2.2. Perancangan Layanan Web 42
4.2.3. Perancangan Kebijakan Otentikasi dan Otorisasi 44
BAB V PENGEMBANGAN DAN PENGUJIAN 47
5.1. Pengembangan 47
5.1.1. Pengembangan Model Data 47
5.1.2. Pengembangan Layanan Web 50
5.1.3. Pengembangan Kebijakan Otentikasi dan Otorisasi 53
5.1.4. Pengembangan Dokumentasi Layanan Web 54
5.2. Pengujian 56

5.2.1. Pengujian Unit 56
5.2.2. Pengujian Fungsional.....57
5.2.3. Pengujian Performa.....59
5.2.4. Pengujian Keamanan 61
BAB VI KESIMPULAN DAN SARAN.....63
6.1. Kesimpulan.....63
6.2. Saran 65
DAFTAR PUSTAKA.....67
BIODATA PENULIS.....71
LAMPIRAN A Keterangan Giyomi.id.....73
LAMPIRAN B Sampel Desain Prototipe.....74
LAMPIRAN C Rancangan Model Data Logis.....75
LAMPIRAN D Dokumentasi Pengujian 76
D.1. Fungsional Manajemen Vendor.....76
D.2. Fungsional Manajemen Pengguna.....83
D.3. Fungsional Manajemen SPK 90
D.4. Performa Kecepatan Layanan Web API11.....116
D.5. Performa Kecepatan Layanan Web API8.....124
D.6. Keamanan 133

DAFTAR GAMBAR

Gambar 1.1 Kerangka Kerja Riset Laboratorium Sistem Enterprise 6
Gambar 2.1 Interaksi Otentikasi dan Otorisasi [10] 12
Gambar 2.2 Paradigma API 13
Gambar 2.3 Tahapan SDLC Waterfall 15
Gambar 3.1 Diagram Metodologi 1 19
Gambar 3.2 Diagram Metodologi 2 20
Gambar 4.1 Diagram Rancangan Model Data Konseptual Versi 1 32
Gambar 4.2 Diagram Rancangan Model Data Konseptual Versi 2 33
Gambar 4.3 (a) Rancangan Model Data Logis User 35
Gambar 4.4 (b) Rancangan Model Data Logis Role 35
Gambar 4.5 (a) Rancangan Model Data Logis Hak 35
Gambar 4.6 (b) Rancangan Model Data Logis Role Hak 35
Gambar 4.7 (a) Rancangan Model Data Logis Koleksi 36
Gambar 4.8 (b) Rancangan Model Data Logis Jenis Koleksi 36
Gambar 4.9 (a) Rancangan Model Data Logis Varian 37
Gambar 4.10 (b) Rancangan Model Data Logis SPK Varian 37
Gambar 4.11 (a) Rancangan Model Data Logis SPK 37
Gambar 4.12 (b) Rancangan Model Sata Logis Nilai SPK 37
Gambar 4.13 Rancangan Model Data Logis Vendor 38
Gambar 4.14 (a) Rancangan Model Data Logis Panjang Kain 38
Gambar 4.15 (b) Rancangan Model Data Logis Satuan Panjang Kain 38
Gambar 4.16 (a) Rancangan Model Data Logis Ukuran 39

Gambar 4.17 (b) Rancangan Model Data Logis Ukuran Jumlah 39
Gambar 4.18 Rancangan Model Data Logis Ukuran Jumlah Penerimaan 40
Gambar 4.19 (a) Rancangan Model Data Logis Proses.....40
Gambar 4.20 (b) Rancangan Model Data Logis SPK Proses .40
Gambar 4.21 Rancangan Model Data Logis Catatan 41
Gambar 4.22 (a) Rancangan Model Data Logis Status 41
Gambar 4.23 (b) Rancangan Model Data Logis SPK Status..41
Gambar 4.24 Diagram alir rancangan otentikasi dan otorisasi pengguna layanan 44
Gambar 5.1 Dokumentasi layanan web.....55
Gambar 5.2 Dokumentasi pengujian unit 56
Gambar 5.3 Dokumentasi pengujian fungsional manajemen SPK.....58
Gambar 5.4 Dokumentasi pengujian fungsional manajemen vendor.....58
Gambar 5.5 Dokumentasi pengujian fungsional manajemen akun pengguna.....59
Gambar 5.6 Dokumentasi pengujian performa 1.....60
Gambar 5.7 Dokumentasi pengujian performa 2.....61
Gambar 5.8 Dokumentasi pengujian keamanan 62
Gambar B.1 (a)(b)(c)(d) Sampel Desain Prototipe Sistem Manajemen Vendor 74
Gambar C.1 Rancangan Model Data Logis.....75

DAFTAR TABEL

Tabel 2.1 Design and Development of Backend Application .. 7
Tabel 2.2 Perancangan Prototipe Sistem Manajemen Vendor . 8
Tabel 2.3 JSON Web Token (JWT) untuk Authentication 9
Tabel 2.4 Development of an E-learning Back-end System .. 10
Tabel 2.5 Rancang Bangun Web Service 11
Tabel 3.1 Daftar Kebutuhan Fungsional 22
Tabel 3.2 Daftar Kebutuhan Layanan Web 23
Tabel 4.1 Daftar Kebutuhan Fungsional 29
Tabel 4.2 Daftar Kebutuhan Layanan Web 30
Tabel 4.3 Daftar rancangan layanan web 42
Tabel 4.4 Pemetaan layanan web yang berhak diminta oleh role 45
Tabel 5.1 Daftar class entitas 48
Tabel 5.2 Daftar class controller 51

Halaman ini sengaja dikosongkan

DAFTAR KODE

Kode 5.1 Mendaftarkan dependency JPA pada aplikasi Spring Boot 47

Kode 5.2 Mendefinisikan class entitas User 49

Kode 5.3 Mendefinisikan interface JPARepository dengan menggunakan User sebagai generic type 50

Kode 5.4 Class Controller RestfulCatatan 52

Kode 5.5 Membuat JWT 53

Kode 5.6 Melakukan ekstraksi username pengguna dari JWT 53

Kode 5.7 Melakukan pemetaan role dengan layanan web 54

Kode 5.8 Method dalam class konfigurasi Swagger 55

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

Bagian ini mendeskripsikan secara umum tentang tugas akhir meliputi latar belakang masalah, rumusan masalah, batasan masalah, dan tujuan tugas akhir, manfaat yang muncul dengan adanya tugas akhir dan relevansi.

1.1. Latar Belakang

Giyomi.id adalah UMKM yang memproduksi pakaian wanita untuk dijual kepada pelanggan melalui daring maupun luring. Giyomi.id menggunakan *marketplace* dan media sosial sebagai wahana menjual secara daring, sedangkan Giyomi.id menjual produknya secara luring di pasar dan bazar. Giyomi.id melakukan proses produksi mulai dari perancangan ide desain produk-produk baru, membuat sampel produk, dan menugaskan kepada penjahit yang disebut dengan vendor untuk memproduksi produk, menerima produk jadi dari vendor sampai pemasaran produk [1].

Proses produksi Giyomi.id dimulai dari mengembangkan ide produk baru sampai menghasilkan desain *mockup* baju. Proses dilanjutkan dengan merencanakan produksi, mencakup merencanakan jumlah produk yang akan di produksi, biaya produksi dan lini masa produksi yang harus dilakukan oleh vendor, yaitu subkontraktor atau penjahit yang melakukan proses pembuatan baju. Proses selanjutnya yang dilakukan adalah pengadaan bahan baku yang dibutuhkan melalui pemasok bahan baku. Bahan baku tersebut dikirim oleh pemasok bahan baku kepada vendor. Proses selanjutnya adalah memberikan Surat Perintah Kerja (SPK) kepada vendor agar pembuatan produk dapat dimulai. Setelah proses pembuatan produk oleh vendor selesai, vendor mengirimkan produk jadi tersebut kepada Giyomi.id [1].

Proses produksi Giyomi.id sangat bergantung pada pihak ketiga yaitu vendor. Performa vendor yang tidak mencapai target dapat menjadi masalah terhadap proses produksi seperti

keterlambatan produksi yang mengganggu proses penjualan sampai menciptakan kerugian finansial. Pengawasan performa vendor seperti mengingatkan akan batas waktu pembuatan produk dan dokumentasi performa vendor menjadi hal yang penting bagi Giyomi.id mengingat vendor-vendor Giyomi.id tidak hanya melayani produksi Giyomi.id, mereka juga menerima pesanan jasa produksi dari pihak lainnya. Giyomi.id perlu untuk mempertimbangkan performa vendor sebagai acuan memutuskan keputusan keberlangsungan kerja sama dengan vendor, namun Giyomi.id belum melakukan dokumentasi performa vendor sebagai acuan pemilihan vendor [2].

Tugas akhir yang dilakukan oleh Bening (2019) telah menghasilkan prototipe sistem manajemen vendor sesuai kebutuhan Giyomi.id dalam bentuk dokumentasi visual yang dapat digunakan sebagai refrensi untuk penelitian lebih lanjut yaitu pengembangan sistem berupa aplikasi *mobile* (sistem operasi Android dan iOS) sistem manajemen vendor. Sampel prototipe sistem manajemen vendor tersebut tercantum pada LAMPIRAN B Sampel Desain Prototipe. Berdasarkan prototipe sistem manajemen vendor tersebut secara umum sistem harus dapat memfasilitasi pengguna untuk melakukan manajemen Surat Perintah Kerja (SPK), manajemen vendor dan manajemen akun pengguna. Pengguna aplikasi mobile sistem manajemen vendor tersebut adalah petugas produksi Giyomi.id (staf produksi) dan *Chief Executive Officer* (CEO) Giyomi.id. Tugas akhir tersebut belum melakukan pengembangan aplikasi *mobile* sebagai solusi nyata yang dapat digunakan oleh Giyomi.id [2].

Melihat kondisi diatas diperlukan adanya penelitian lanjutan yang dapat menghasilkan aplikasi sistem manajemen vendor berbasis *mobile* yang dapat diaplikasikan di Giyomi.id. Pengaplikasian sistem tersebut dilakukan demi mengurangi risiko keterlambatan produksi karena kesalahan memilih vendor yang memiliki performa tidak mencapai target dan berdampak pada keterlambatan penjualan hingga mencipatakan kerugian finansial. Tugas akhir ini adalah penelitian lanjutan

yang mengubah prototipe sistem manajemen vendor menjadi komponen sistem yang nyata. Sistem manajemen vendor tersebut harus dapat digunakan pada *platform mobile* android dan *mobile* iOS.

Service Oriented Architecture (SOA) adalah arsitektur pengembangan perangkat lunak yang sering digunakan dalam pengembangan aplikasi *mobile*, walaupun kini telah banyak variasi arsitektur lainnya seperti *cloud computing* atau *microservices*. SOA banyak digunakan karena mendukung pengembangan aplikasi dengan merakit layanan web yang dapat diakses melalui internet sehingga pengembang dapat mempercepat pengembangan perangkat lunak. Mengacu kepada SOA, satu layanan web dapat dikonsumsi oleh berbagai *platform*. Pada SOA sistem dibagi menjadi 2 bagian yaitu aplikasi *frontend* dan aplikasi *backend*. Aplikasi *backend* adalah perangkat lunak pada bagian server sedangkan aplikasi *frontend* adalah perangkat lunak pada bagian *client*. Bentuk luaran aplikasi *backend* adalah layanan web, sedangkan bentuk luaran aplikasi *frontend* adalah antarmuka aplikasi dengan pengguna. Aplikasi *frontend* akan mengonsumsi layanan web yang disediakan oleh aplikasi *backend* (*server-side*) [3].

Dokumentasi aplikasi *backend* diperlukan oleh konsumen layanan web. Dokumentasi aplikasi *backend* yang dimaksud adalah dokumentasi layanan-layanan web yang dihasilkan oleh aplikasi *backend*. Dokumentasi aplikasi *backend* memungkinkan konsumen layanan web untuk mengetahui masukan yang dibutuhkan oleh layanan web dan luaran yang akan dihasilkan dari layanan web serta kebijakan layanan web [4]. Swagger adalah alat yang dapat membangun dokumentasi layanan web aplikasi *backend* secara otomatis melalui kode program, dengan menggunakan Swagger, proses pembuatan dokumentasi aplikasi *backend* akan menjadi lebih cepat [5]. Selain dokumentasi, pengembangan aplikasi *backend* membutuhkan *Software Development Life Cycle* (SDLC) untuk memastikan kualitas aplikasi *backend*, mempermudah kontrol implementasi dan memastikan aplikasi dapat memenuhi

kebutuhan pengguna [6]. *Waterfall* adalah salah satu jenis SDLC, *waterfall* memiliki kelebihan dapat membuat pengembangan perangkat lunak menjadi tepat waktu [7].

Tugas akhir ini fokus pada perancangan dan pengembangan aplikasi *backend* sistem manajemen vendor. Aplikasi *backend* akan bertidak sebagai penyedia layanan web yang dapat dikonsumsi oleh aplikasi *frontend* agar menjadi aplikasi *mobile* yang dapat digunakan oleh Giyomi.id. Aplikasi *backend* akan dikembangkan berdasarkan prototipe sistem manajemen vendor yang telah dihasilkan penelitian yang dilakukan oleh Bening (2019). Penelitian ini akan menggunakan metode SDLC *waterfall* dan menggunakan Swagger sebagai alat untuk membuat dokumentasi aplikasi *backend*. Aplikasi b*ackend* ini dapat menjadi aplikasi *backend* sistem manajemen vendor Giyomi.id yang dapat diintegrasikan dengan hasil penelitian selanjutnya yaitu aplikasi *frontend* sistem manajemen vendor.

1.2. Rumusan Masalah

Giyomi.id dalam proses produksinya perlu melakukan pengambilan keputusan keberlangsungan kerjasama terhadap vendor berdasarkan performa vendor. Giyomi.id membutuhkan informasi tentang performa tiap vendor. Kondisi saat ini, Giyomi.id belum mendokumentasikan riwayat performa vendor sehingga belum ada sumber yang dapat memberikan informasi akan performa tiap vendor. Kesalahan pemilihan vendor mengakibatkan permasalahan pada proses produksi Giyomi.id yaitu keterlambatan proses produksi dan memunculkan kerugian finansial dikarenakan performa vendor yang tidak baik.

Penelitian yang dilakukan oleh Bening pada tahun 2019 menghasilkan prototipe sistem manajemen vendor yang telah dirancang dapat menjadi solusi akan permasalahan tersebut. Perlu adanya perancangan dan pengembangan aplikasi *backend* sistem manajemen vendor berdasarkan desain prototipe dengan fitur utama manajemen Surat Perintah Kerja (SPK), manajemen vendor dan manajemen akun pengguna yang telah dihasilkan

oleh penelitian sebelumnya yang dapat diintegrasikan dengan hasil penelitian selanjutnya yaitu aplikasi *frontend* untuk menjadi sebuah aplikasi *mobile* sistem manajemen vendor yang dapat digunakan oleh Giyomi.id.

1.3. Batasan Tugas Akhir

Aplikasi *backend* sistem manajemen vendor Giyomi.id dirancang dan dikembangkan sesuai dengan prototipe sistem manajemen vendor hasil tugas akhir yang dikerjakan oleh Bening pada tahun 2019.

1.4. Tujuan Tugas Akhir

Tujuan dari tugas akhir ini antara lain:

1. Merancang aplikasi *backend* berdasarkan prototipe sistem manajemen vendor [2].
2. Mengembangkan aplikasi *backend* menggunakan bahasa pemrograman Java dan kerangka kerja Spring Boot.
3. Membuat dokumentasi aplikasi *backend* menggunakan Swagger.
4. Menguji aplikasi *backend* dengan pengujian unit, pengujian fungsional, pengujian performa dan pengujian keamanan.

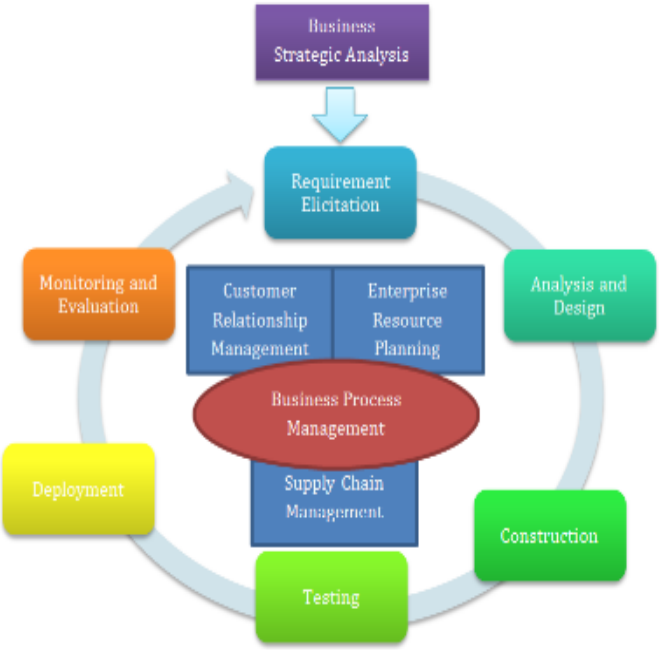
1.5. Manfaat Penelitian

Berikut adalah manfaat yang diharapkan dari pengerjaan tugas akhir ini:

1. Giyomi.id dapat menggunakan sistem untuk menjalankan proses produksinya untuk mempermudah proses pengawasan dan dokumentasi.
2. Meningkatkan kemampuan analisa vendor dalam proses pemilihan vendor sehingga mengurangi risiko yang diakibatkan oleh kesalahan memilih vendor.

1.6. Relevansi

Tugas Akhir ini mengangkat topik mengenai *Enterprise Resource Planning*, dimana topik ini memiliki kontribusi pada tujuan tugas akhir Laboratorium *Sistem Enterprise* (SE) Departemen Sistem Informasi ITS Surabaya, yaitu untuk meningkatkan pemahaman mengenai bagaimana suatu solusi IT di dalam organisasi atau perusahaan dapat membantu mencapai tujuan bisnis pada suatu organisasi. Tugas Akhir ini memiliki revelan dengan mata kuliah yang dipelajari di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember, yaitu *Sistem Enterprise* (SE), Manajemen Proses Bisnis (MPB), Dasar Desain Pengembangan Perangkat Lunak (DDPL), Bahasa Pemrograman (BP) dan Teknologi Basis Data (TBD). Kerangka kerja riset Laboratorium *Sistem Enterprise* dapat divisualisasikan pada Gambar 1.1.



Gambar 1.1 Kerangka Kerja Riset Laboratorium Sistem Enterprise

BAB II
KAJIAN PUSTAKA

Bab ini akan membahas tentang literatur yang relevan yang akan digunakan dalam pengerjaan tugas akhir, terdiri dari penelitian sebelumnya dan dasar teori.

2.1. Penelitian Sebelumnya

Dalam tugas akhir ini, digunakan beberapa penelitian terdahulu sebagai pedoman dan referensi dalam melaksanakan proses-proses dalam penelitian, seperti yang terdapat pada Tabel 2.1, Tabel 2.2, Tabel 2.3 JSON Web Token (JWT) untuk Authentication Tabel 2.4 Development of an E-learning Back-end System, dan Tabel 2.5 Rancang Bangun Web Service berisi informasi penelitian sebelum serta hubungan penelitian terhadap tugas akhir ini.

Tabel 2.1 *Design and Development of Backend Application*

Judul Penelitian	*Design and Development of Backend Application for Public Complaint Systems Using Microservice* Spring Boot [8]
Penulis, tahun	Hatma Suryotrisongko, Dedy Puji Jayanto, Aris Tjahyanto, 2017
Deskripsi umum penelitian	Penelitian ini membahas tentang desain dan pengembangan aplikasi backend *E-Government* menggunakan arsitektur *microservice* serta menggunakan kerangka kerja Spring Boot. Tahapan penelitian yang dilakukan dimulai dari analisa kebutuhan fungsional aplikasi, lalu di lanjutkan dengan pemodelkan *microservice*, selanjutnya pembuatan desain *use case*, lalu pengembangan aplikasi dan yang terakhir testing aplikasi dengan *blackbox testing*. Pada tahapan pemodelan *microservice*, penulis membuat daftar *class controller*, daftar

	metode *representational state transfer* (REST), daftar REST *Uniform Resource Loocator* (URL), diagram arsitektur aplikasi serta perenanaan pengamanan sistem yang sesuai dengan kebutuhan aplikasi.
Keterkaitan penelitian	Penelitian ini memiliki keterkaitan dengan tugas akhir. Luaran yang dihasilkan oleh metode yang digunakan untuk mengembangkan aplikasi *backend* pada tahap analisa kebutuhan fungsional sampai menghasilkan daftar layanan dapat diaplikasikan dalam tugas akhir serta penelitian ini juga memiliki kerangka kerja yang sama dengan kerangka kerja yang akan digunakan pada tugas akhir yaitu kerangka kerja Spring Boot.

Tabel 2.2 Perancangan Prototipe Sistem Manajemen Vendor

Judul Penelitian	Perancangan Prototipe Sistem Manajemen Vendor dengan Metode *Iterative Prototyping* dan *Moderated Usability Test*. (Studi Kasus: Giyomi.Id) [2]
Penulis, tahun	Erica Maulidina Bening, 2019
Deskripsi umum penelitian	Penelitian ini berfokus pada pembuatan prototipe sistem manajemen vendor sesuai dengan kebutuhan Giyomi.id dan terdokumentasi secara visual dengan baik, demi penelitian selanjutnya yaitu pengembangan aplikasi sesuai prototipe yang telah dibuat. Penelitian ini menggunakan metode *iterative prototyping* yang mengharuskan proses penelitian untuk secara berulang untuk meminta respon balik dari pengguna akan

	prototipe sistem manajemen vendor yang telah dibuat. Penelitian ini menghasilkan prototipe sistem manajemen vendor yang dapat digunakan sebagai panduan bagi penelitian selanjutnya untuk mengembangkan aplikasi agar dapat digunakan secara riil.
Keterkaitan penelitian	Penelitian ini menghasilkan prototipe sistem manajemen vendor yang akan menjadi panduan untuk perancangan aplikasi *backend* sistem manajemen vendor Giyomi.id. Penelitian ini juga berguna untuk tugas akhir agar dapat berfokus pada pengembangan aplikasi *backend* tanpa perlu untuk melakukan analisa kebutuhan pengguna.

Tabel 2.3 JSON Web Token (JWT) untuk *Authentication*

Judul Penelitian	JSON *Web* Token (JWT) untuk *Authentication* pada Interoperabilitas Arsitektur Berbasis RESTful *Web Service*
Penulis, tahun	Rohmat Gunawan, Alam Rahmatulloh, 2019
Deskripsi umum penelitian	Penelitian ini mengambangkan layanan *web* yang skalabel, aman dan mampu berinteraksi *multi-platform*. Layanan *web* yang dibuat berbasis *Representational State Transfer* (REST) dan menggunakan JSON *Web Token* (JWT) sebagai keamanan layanan web. Penelitian ini juga menjelaskan komponen penyusun JWT dan cara kerja JWT. Algoritma *encoding* yang digunakan pada penelitian ini adalah Base64. Penelitian ini juga membahas tentang perancangan serta otentikasi dan otorisasi pengguna, dimana pengguna

	memiliki jabatan yang berbeda-beda dan memiliki hak-hak atas akses layanan yang berbeda-beda.
Keterkaitan penelitian	Tugas akhir akan menggunakan JWT untuk melakukan otorisasi dan otentikasi pada pengguna sistem. Penelitian ini memberikan contoh penggunaan JWT untuk diimplementasikan dalam pengembangan aplikasi *backend*. Penelitian ini juga memberikan referensi tentang perancangan kebijakan otorisasi dan otentikasi pengguna.

Tabel 2.4 Development of an E-learning Back-end System

Judul Penelitian	*Development of an E-learning Back-end System for Code Assessment in Elementary Programming Practice* [9]
Penulis, tahun	Atsushi Nunome, Hiroaki Hirata, Masayuki Fukuzawa, 2010
Deskripsi umum penelitian	Penelitian ini mengembangkan aplikasi *backend E-Learning* untuk penilaian dalam pelajaran pemrograman dasar. Kelompok pengguna dari aplikasi *backend* yang dibuat terdiri dari guru dan murid yang setiap kelompok pengguna memiliki hak-hak atas layanan web yang berbeda. Aplikasi *backend* yang dikembangkan mendapatkan akun murid dari *Learning Management System* (LMS) yang telah tersedia, sehingga aplikasi *backend* yang dikembangkan tidak perlu memiliki tanggungjaawab terhadap manajemen akun. Aplikasi *backend* yang dikembangkan pada penelitian ini dirancang untuk bekerja secara optimal

	dengan melakukan *caching* pada sumber daya yang sering diakses.
Keterkaitan penelitian	Penelitian ini menjelaskan kebijakan *caching* pada sumber daya yang sering dipakai demi meningkatkan kecepatan pelayanan. Tugas akhir ini akan melakukan *caching* yang serupa kepada layanan web yang berpotensi akan sering diakses.

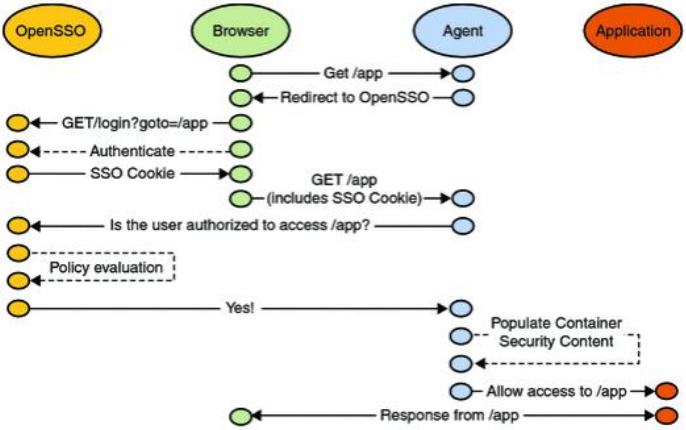
Tabel 2.5 Rancang Bangun *Web Service*

Judul Penelitian	Rancang Bangun *Web Service* pada Sistem Informasi Akademik Berbasis *Representational State Transfer* Menggunakan *Service-Repository Pattern* [7].
Penulis, tahun	Oky Suryadi, 2019
Deskripsi umum penelitian	Penelitian ini fokus pada merancang dan mengembangkan layanan web dari sistem informasi akademik yang telah ada di Institut Teknologi Sepuluh Nopember agar layanan yang terdapat pada sistem informasi tersebut dapat digunakan pada *mobile platform*. Layanan web yang digunakan berbasis REST agar layanan dapat mengirim informasi langsung menggunakan *typical encoding* seperti JSON, sehingga permintaan layanan yang dilakukan dapat dengan mudah dilakukan. SDLC yang digunakan bertipe *waterfall* dan pengujian yang dilakukan berupa testing unit. Pengembangan layanan web yang dilakukan pada modul aktivitas, formulir rencana studi (FRS), indeks prestasi dosen (IPD), kurikulum, penilaian, penjadwalan, profil, dan satuan kredit ekstrakurikuler mahasiswa

	(SKEM). Pengembangan dilakukan dengan bahasa pemrograman *Hypertext Preprocessor* (PHP).
Keterkaitan penelitian	Penelitian ini mengembangkan layanan berjenis REST. Tugas akhir juga akan mengembangkan layanan berjenis REST.

2.2. Otentikasi dan Otorisasi

Otentikasi dan otorisasi adalah identifikasi pengguna dalam sebuah sistem aplikasi. Otorisasi fokus pada identifikasi apa saja hak-hak suatu pengguna pada sistem terkait. Otentikasi fokus pada identifikasi siapa pengguna yang sedang mengakses sistem terkait pada suatu waktu tertentu [10]. Interaksi otentikasi dan otoriasasi pada sebuah aplikasi dapat dijelaskan oleh Gambar 2.1.

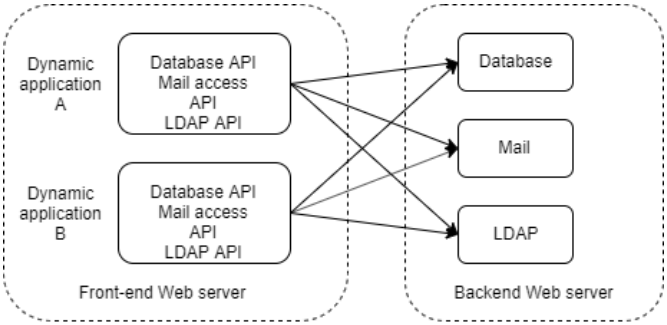


Gambar 2.1 Interaksi Otentikasi dan Otorisasi [10]

2.3. Aplikasi Backend

Aplikasi *backend* adalah aplikasi yang berjalan pada bagian server yang bertujuan untuk menghilangkan ketergantungan aplikasi bagian *client* dari penggunaan spesifik *platform*.

Aplikasi *client* mengakses aplikasi *backend* melalui spesifik *application programming interface* (API). Salah satu wujud API adalah layanan web [11]. Paradigma yang diadopsi pada server dapat dijelaskan oleh Gambar 2.2.



Gambar 2.2 Paradigma API

2.4. Service Oriented Architecture

Service Oriented Architecture (SOA) adalah arsitektur pengembangan perangkat lunak yang sering digunakan, walaupun kini telah banyak variasi arsitektur lainnya seperti *cloud computing* atau *microservices*. SOA banyak digunakan karena mendukung pengembangan aplikasi dengan merakit layanan yang dapat diakses melalui internet sehingga pengembang dapat mempercepat pengembangan perangkat lunak [3].

2.5. Layanan Web

Layanan web adalah sebuah sistem terdistribusi yang memiliki komponen yang dapat diakses menggunakan protokol HTTP (*Hyper Text Transport Protocol*) maupun HTTPS (HTTP *Secure*). Layanan web memiliki dua teknik yang paling sering digunakan, yaitu *Simple Object Access Protocol* (SOAP) dan *Representationial State Transfer* (REST). Pada layanan web berbasis SOAP, klien mengirimkan pesan SOAP aplikasi *backend* melalui layanan web dan aplikasi *backend* merespon dalam jenis pesan SOAP pula. Pada layanan web berbasis

REST, klien mengirim pesan kepada *backend* dalam bentuk JavaScript *Object Notation* (JSON) dan aplikasi *backend* merespon pesan tersebut dengan bentuk JSON juga. Dalam layanan RESTful, HTTP itu sendiri dapat diketahui sebagai suatu API [12].

2.6. Spring Boot

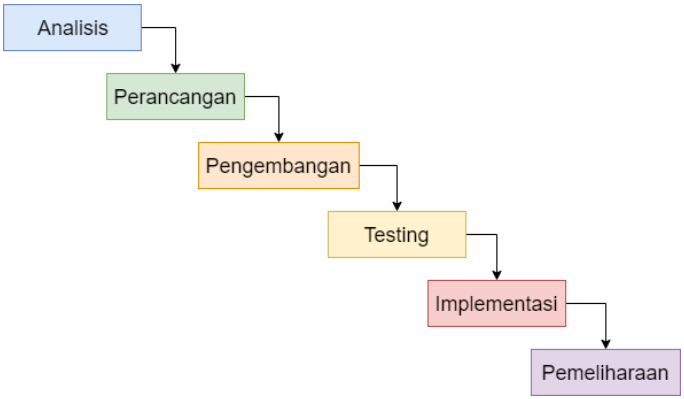
Spring Boot adalah kerangka kerja pengembangan aplikasi pada *platform* web menggunakan bahasa pemrograman Java yang memiliki beberapa kelebihan. Spring Boot mendukung pengembangan aplikasi *backend* dengan gaya arsitektur *representational state transfer* (REST). Koneksi ke basis data juga telah dipermudah dengan paket Spring Boot. Kerangka kerja Spring Boot juga mendukung *dependency injection* yang dapat mempermudah konfigurasi *dependency* [8].

2.7. Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC) adalah pedoman atau metodologi yang digunakan untuk proses pengembangan perangkat lunak. Biasanya SDLC terdiri dari beberapa komponen seperti perancangan, pengembangan dan pengujian perangkat lunak. Dengan SDLC diharapkan perangkat lunak dapat menjadi berkualitas sesuai dengan kebutuhan. SDLC terdiri dari beberapa jenis seperti *waterfall*, *spiral*, *agile* dan prototipe. Jenis-jenis SDLC tersebut digunakan demi memastikan keberhasilan proses pengembangan perangkat lunak [7].

2.7.1. SDLC Waterfall

SDLC *Waterfall* adalah SDLC *sequential* dimana kebutuhan harus selesai sebelum dilanjutkan ke tahap selanjutnya. Pengujian pada *waterfall* dilakukan pada saat pengembangan telah selesai sepenuhnya dilakukan. Setiap tahap pada SDLC *waterfall* harus diselesaikan dalam jangka waktu tertentu dan setiap tahap harus memiliki dokumentasi atau keluaran [13]. Tahapan pada SDLC waterfall dapat divisualisasikan oleh Gambar 2.3.



Gambar 2.3 Tahapan SDLC Waterfall

Berdasarkan gambar tersebut terlihat bahwa tahapan SDLC *waterfall* berjumlah 6 tahap yaitu analisis, perancangan, pengembangan, testing atau pengujian, implementasi dan pemeliharaan. Tahapan tersebut harus dilaksanakan secara berurutan mulai dari atas dan berjalan menurun bawah [13].

2.8. Swagger

Swagger adalah alat yang dapat membangun dokumentasi layanan aplikasi *backend* secara otomatis, dengan menggunakan Swagger, proses pengembangan aplikasi backend akan menjadi lebih cepat. Swagger dapat digunakan untuk mendokumentasikan layanan web yang memiliki *versioning*. Calon pengguna layanan web dapat mengamati dengan mudah seluruh versi layanan web dan informasi tentang masukan apa yang dibutuhkan untuk menggunakan layanan web serta keluaran yang akan dihasilkan oleh layanan web. Dokumentasi layanan web akan secara otomatis diperbaharui jika terjadi perubahan pada layanan web [5].

2.9. Bahasa Pemrograman Java

Java adalah bahasa pemrograman berorientasi objek dan bertujuan umum, konkuren dan *strongly typed*. Bahasa pemrograman Java dimunculkan pada tahun 1995 oleh

perusahaan Sun Microsystems yang kemudian diakuisisi oleh perusahaan Oracle Corporation. Java telah berkembang seiring berjalannya waktu dengan menambah fitur-fiturnya seperti yang ada pada Java versi 8 [14].

2.10. JUnit

JUnit adalah alat yang dapat digunakan untuk melakukan pengujian unit pada program yang dibuat dengan bahasa pemrograman java. Pengujian yang fokus menguji kode program telah bekerja sesuai kebutuhan. Pengujian unit dilakukan pada unit-unit kode program dalam sebuah sistem. Alat yang dapat digunakan untuk melakukan penguian unit menggunakan bahasa pemrograman Java adalah JUnit. [15].

2.11. ReadyAPI

ReadyAPI adalah sebuah prangkat lunak yang dapat membantu pengembang layanan web untuk memastikan layanan web memiliki kualitas tinggi, performa yang baik dan keamanan yang baik. ReadyAPI memungkinkan pengembang layanan web untuk melakukan pengujian fungsional, pengujian performa dan pengujian keamanan dalam satu alat. ReadyAPI adalah *software* berbayar namun pengembang layanan web dapat memasang *software* ReadyAPI dengan izin percobaan secara gratis [16]. Pengujian yang dapat dilakukan dengan menggunakan ReadyAPI antara lain:

a. Pengujian Fungsional

Pengujian yang fokus kepada fungionalitas layanan web. Pengujian fungsional memastikan luaran atau respon dari layanan web sesuai dengan kebutuhan layanan web. Alat yang dapat. [17].

b. Pengujian Performa

Pengujian yang fokus kepada kecepatan suatu layanan web dapat menerima permintaan pengguna sampan pengguna menerima luaran dari layanan web tersebut [17].

c. Pengujian Keamanan

Pengujian yang fokus kepada mengukur *vulnerability* sebuah layanan web dari berbagai ancaman keamanan seperti *SQL Injection* dan *Weak Authentication* [18].

2.12. IntelliJ IDEA

IntelliJ IDEA adalah *Integrated Development Environment* (IDE) untuk bahasa pemrogramman Java. IntelliJ IDEA memungkinkan penggunanya untuk melakukan penulisan kode program Java dengan penyelesaian kode otomatis. IntelliJ IDEA dikembangkan oleh perusahaan bernama JetBrains. IntelliJ IDEA dapat digunakan dengan mode bayar, percobaan atau lisesnsi pelajar [19].

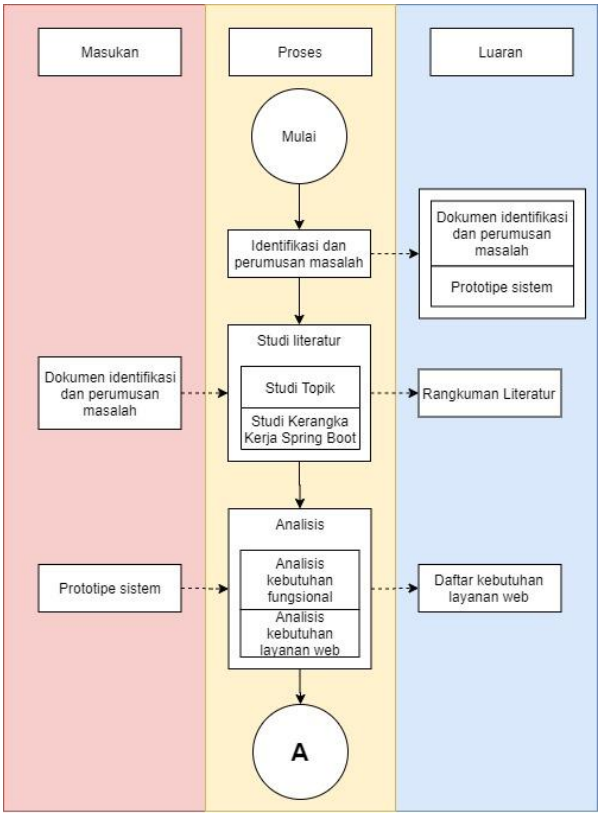
Halaman ini sengaja dikosongkan

BAB III
METODOLOGI

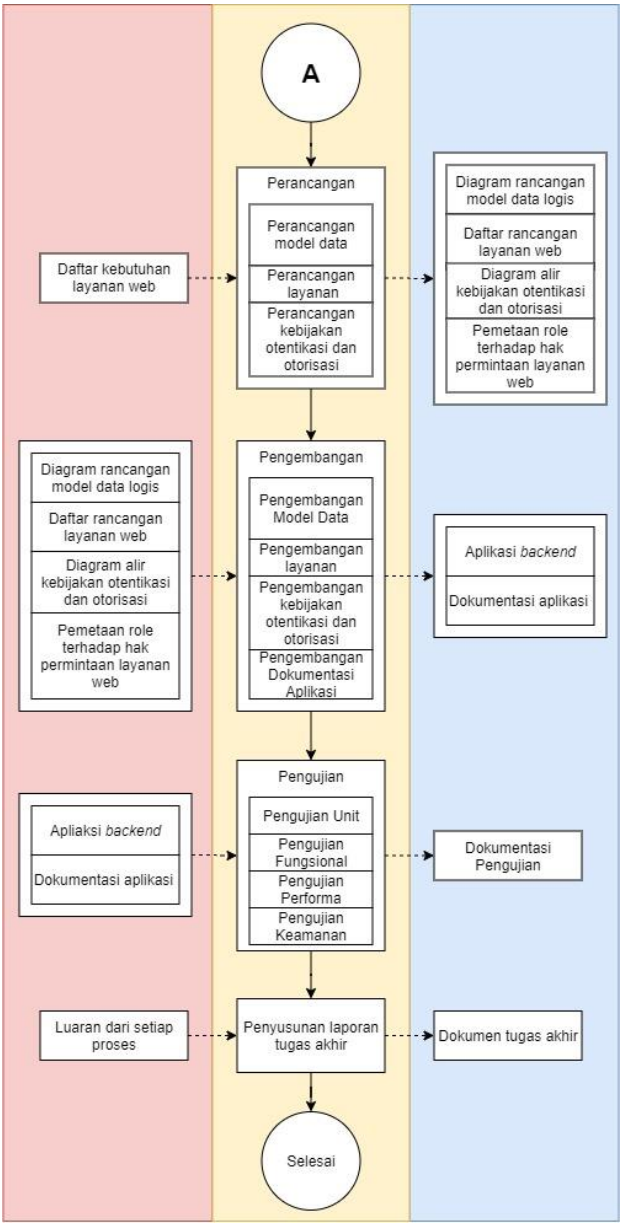
Pada bagian ini dijelaskan metodologi yang digunakan sebagai panduan untuk menyelesaikan penelitian tugas akhir ini.

3.1. Diagram Metodologi

Visualisasi melalui diagram metodologi tugas akhir ini berada pada Gambar 3.1 dan Gambar 3.2.



Gambar 3.1 Diagram Metodologi 1



Gambar 3.2 Diagram Metodologi 2

3.2. Uraian Metodologi

Pada bagian ini menjelaskan seluruh tahapan yang dilakukan penelitian berdasarkan metodologi yang digambarkan pada Gambar 3.1 Diagram Metodologi 1 dan Gambar 3.2 Diagram Metodologi 2

3.2.1. Identifikasi dan Perumusan Masalah

Tahap awal ini dimulai dengan melakukan identifikasi masalah melalui studi penelitian yang dilakukan oleh Bening pada tahun 2019 dan Kinasih pada tahun 2019 [1], [2]. Tahap ini dilakukan dengan tujuan untuk mengamati dan menemukan masalah yang akan diteliti dalam tugas akhir dan merumuskan masalah yang harus diselesaikan, batasan penelitian, tujuan penelitian, manfaat penelitian, dan relevansi. Luaran pada tahap ini adalah dokumen identifikasi masalah dan perumusan masalah serta prototipe sistem manajemen vendor.

3.2.2. Studi Literatur

Pada tahap ini dilakukan dengan mengumpulkan informasi mengenai topik yang akan diangkat untuk memahami topik dan kerangka kerja yang digunakan untuk tugas akhir. Fase ini berguna untuk membangun pemahaman dasar dalam mengerjakan tugas akhir dan mencari penelitian sebelumnya pada topik terkait. Luaran pada tahap ini adalah rangkuman literatur.

3.2.3.1. Studi Topik

Studi topik dilakukan dengan melakukan studi pada literatur terkait topik tugas akhir dalam bentuk buku, *paper*, publikasi dan penelitian sebelumnya.

3.2.3.2. Studi Kerangka Kerja Spring Boot

Studi kerangka kerja dilakukan dengan melakukan studi literatur kerangka kerja Spring Boot yang digunakan pada tugas akhir ini dalam bentuk penelitian sebelumnya dan dokumentasi kerangka kerja.

3.2.3. Analisis

Tahap ini akan dilakukan penggalian kebutuhan mengenai aplikasi *backend* yang akan dibuat. Tahap ini memerlukan masukan berupa prototipe sistem manajemen vendor dan akan menghasilkan luaran berupa daftar kebutuhan layanan web yang harus dimiliki oleh aplikasi *backend*.

3.2.3.1. Analisis Kebutuhan Fungsional

Tahap ini adalah tahap menarik kebutuhan fungsional dari prototipe sistem manajemen vendor. Tahap ini akan menghasilkan tabel yang menjelaskan daftar kebutuhan fungsional. Kolom tabel tersebut terdiri *ID* kebutuhan layanan dan aktornya. [8]. Tabel hasil tahap analisis kebutuhan funsional terlihat seperti Tabel 3.1.

Tabel 3.1 Daftar Kebutuhan Fungsional

ID	*Role*	Kebutuhan Fungsional
FR1	*Role* 1	Kebutuhan Fungsional 1
FR2	*Role* 2	Kebutuhan Fungsional 2
FR3	*Role* 3	Kebutuhan Fungsional 3

Pada Tabel 3.1 terdapat kolom ID yang berisi simbol identifikasi unik untuk setiap kebutuhan fungsional, kolom *role* yang merepresetasikan aktor atau peran pengguna dalam aplikasi *backend* dan kolom kebutuhan fungsional yang merepresentasikan fungsional dari aplikasi *backend*.

3.2.3.2. Analisis Kebutuhan Layanan Web

Tahap ini adalah tahap menganalisis kebutuhan layanan web. Pada satu kebutuhan fungsional dapat dilayani oleh satu atau banyak layanan web. Tahap ini akan menghasilkan tabel daftar kebutuhan layanan dengan kolom terdiri *ID* layanan kebutuhan layanan, kebutuhan layanan dan *ID* kebutuhan fungsional yang

membutuhkan layanan tersebut [8]. Tabel hasil pada tahap analisis kebutuhan funsional akan terlihat seperti Tabel 3.2.

Tabel 3.2 Daftar Kebutuhan Layanan Web

ID	ID FR	Kebutuhan Layanan Web
API1	FR1	Layanan Web 1
API2	FR1	Layanan Web 2
API3	FR2	Layanan Web 3
API4	FR2	Layanan Web 4
API5	FR3	Layanan Web 5

Pada Tabel 3.2 terdapat kolom ID yang berisi simbol identifikasi unik untuk setiap layanan web, kolom ID FR untuk pemetaan kebutuhan fungsional yang didukung oleh layanan web terkait dan kolom kebutuhan layanan web yang merepresentasikan layanan yang akan dimiliki oleh aplikasi *backend*.

3.2.4. Perancangan

Tahap ini akan dilakukan perancangan aplikasi *backend* yang akan dibuat. Tahap ini memerlukan masukan berupa daftar kebutuhan layanan dan akan menghasilkan keluaran berupa daftar kebutuhan layanan web berdasarkan prototipe sistem manajemen vendor yang harus dapat difasilitasi oleh aplikasi *backend*, diagram rancangan model data logis, diagram alir kebijakan otentikasi dan otorisasi serta pemetaan *role* pengguna terhadap layanan web yang berhak diminta oleh pengguna dengan *role* terkait.

3.2.4.1. Perancangan Model Data

Perancangan model data diperlukan agar bentuk data yang akan dipakai mendukung data yang konsisten dan menghindari duplikasi data [20]. Langkah pertama yang harus dilakukan dalam perancangan data adalah merancang model data konseptual [21].

3.2.4.1.1. Perancangan Model Data Konseptual

Perancangan model data konseptual merupakan perancangan model data yang merepresentasikan kebutuhan relasi antar data sesuai kebutuhan pengguna. Model data konseptual berisi entitas, relasi antar entitas dan tipe relasi antar entitas [20]. Pembuatan model data berupa *Entity Relationship Diagram* (ERD) menggunakan alat DataGrip [22]. Pada tahap in juga akan dilakukan validasi master data yang akan ada pada sistem manajemen vendor.

3.2.4.1.2. Perancangan Model Data Logis

Perancangan model data logis merupakan perancangan model data yang merepresentasikan bentuk data lebih detil daripada model data konseptual. Model data logis memberikan informasi tentang entitas, tipe data masing-masing atribut dalam entitas, *primary keys* dan *foreign keys*. [23]. Tahap perancangan model data logis akan menghasilkan luaran berupa diagram model data logis.

3.2.4.2. Perancangan Layanan Web

Tahap perancangan layanan web terdiri dari definisi sumber daya yang akan dihasilkan dari layanan dan metode layanan. Deskripsi akan menjadi acuan implementasi layanan dalam melakukan pengolahan data sebelum data dikeluarkan oleh layanan. Metode layanan mendeskripsikan metode apa yang harus digunakan pengguna layanan untuk menggunakan layanan, metode layanan dapat berupa *POST*, *GET*, *DELETE*, *PUT* [24]. Tahap perancangan layanan web akan menghasilkan luaran berupa daftar rancangan layanan web sesuai dengan daftar kebutuhan layanan web.

3.2.4.3. Perancangan Kebijakan Otentikasi dan Otorisasi

Tahap perancangan kebijakan otentikasi dan otorisasi terdiri dari perancangan diagram alir kebijakan oentikasi dan otorisasi serta pemetaan *role* pengguna terhadap layanan web sebagai hak *role* tersebut untuk melakukan permintaan layanan terkait.

Otentikasi fokus kepada identifikasi siapa yang meminta layanan web, sedangkan otorisasi fokus pada identifikasi hak-hak layanan web untuk diminta oleh pengguna [25]. Hasil dari tahapan ini adalah diagram alir kebijakan otentikasi dan otorisasi serta daftar pemetaan *role* pengguna dengan hak untuk melakukan permintaan layanan web.

3.2.5. Pengembangan

Tahapan pengembangan adalah tahapan yang mengimplementasikan rancangan-rancangan yang telah dibuat pada tahap perancangan, sehingga tahap ini membutuhkan luaran yang dihasilkan oleh tahap perancangan sebagai masukan dan akan menghasilkan aplikasi *backend* serta dokumentasi aplikasi *backend.* Seluruh penulisan kode program Java akan dilakukan dengan penggunaan *Integrated Development Environment* (IDE) Intellij IDEA lisensi pelajar.

3.2.5.1. Pengembangan Model Data

Implementasi model data akan menggunakan *Java Persistence* API (JPA) yang telah tersedia pada kerangka kerja Spring Boot. JPA adalah *library* yang tersedia pada Spring Boot yang memungkinkan pengembang untuk membuat model data melalui Java *Class*. Java *Class* tersebut akan merepresentasikan model data dan menjadi *interface* bagi pengembang untuk melakukan aksi pada data seperti mamgambil, menghapus dan memodifikasi data [26].

3.2.5.2. Pengembangan Layanan

Implementasi layanan akan menggunakan anotasi REST *Controller* yang telah tersedia pada pada Spring Boot. REST *Controller* dapat menerima permintaan akses melalui protokol HTTP, menerima data masukan serta mengembalikan data luaran. Pada REST *Controller* pengembang harus mendefinisikan metode sesuai dengan perancangan layanan yang telah dibuat sebelumnya [27].

3.2.5.3. Pengembangan Kebijakan Otentikasi dan Otorisasi

Implelementasi kebijakan otentikasi dan otorisasi akan menggunakan fitur keamanan yang ada pada kerangka kerja Sring Boot dan diimplementasikan dengan JWT OAuth2 [28]. Penggunaan fitur keamanan dari keranka kerja Sring Boot akan disesuaikan dengan diagram alir kebijakan otentikasi dan otorisasi yang telah dibuat pada tahap perancangan kebijakan otentikasi dan otorisasi. Pengembangan kebijakan otorisasi akan disesuaikan dengan daftar pemetaan r*ole* terhadap hak untuk melakukan permintaan layanan web yang telah dibuat pada tahapan perancangan kebijakan otentikasi dan otorisasi. Setelah pengembangan kebijakan otentikasi dan otorisasi dilakukan maka telah terbentuk aplikasi *backend* sebagai luaran tahap pengembangan.

3.2.5.4. Pengembangan Dokumentasi Aplikasi

Pengembangan dokumentasi aplikasi *backend* menggunakan Swagger yang dapat memproduksi dokumentasi aplikasi *backend* secara otomatis berdasarkan layanan yang telah dikembangkan. Dokumentasi aplikasi *backend* tersebut akan dapat diakses melalui *browser*. Luaran dari tahap pengembangan dokumentasi aplikasi adalah sebuah dokumentasi aplikasi *backend* yang dapat diakses melalui *browser*.

3.2.6. Pengujian

Pengujian yang dilakukan adalah pengujian unit, pengujian fungsional, pengujian performa dan pengujian keamanan. Setiap pengujian yang dilakukan akan menhasilkan dokumentasi pengujian.

3.2.6.1. Pengujian Unit

Pengujian unit adalah pengujian yang pertama kali dilakukan untuk memstikan bahwa program Java yang telah tertulis dapat bekerja sesuai dengan kebutuhan. Unit yang dimaksud dalam

hal ini adalah *method* yang dibuat dengan kode program Java [15]. Pengujian dilakukan dengan menuliskan kode program pengujian dalam bahasa pemgrograman Java yang dibantu dengan alat JUnit. Pengujian unit akan dilakukan pada unit-unit yang penting. Dokumantasi pengujian unit adalah tangkapan layar hasil pengujian unit pada IDE IntelliJ IDEA.

3.2.6.2. Pengujian Fungsional

Pengujian fungsional adalah pengujian yang dilakukan setelah pengujian unit selesai dilakukan. Pengujian fungsonal akan dilakukan dengan bantuan alat ReadyAPI. Layanan web yang diuji hanya layanan web penting yang mendukung fungsionalitas fitur-fitur utama aplikasi *backend* yaitu manajemen vendor, manajemen Surat Perintah Kerja (SPK) dan manajemen pengguna.

3.2.6.3. Pengujian Performa

Pengujian performa yang akan dilakukan adalah pengujian kecepatan aplikasi *backend* untuk merespon permintaan layanan web yang dilakukan oleh pengguna layanan web. Ada 2 jenis pengujian performa yang akan dilakukan yaitu pengujian kecepatan respon layanan web yang memiliki kompleksitas tinggi pada saat digunakan oleh satu pengguna dan pengujian kecepatan respon layanan web yang memiliki kemungkinan diminta oleh pengguna dengan frekuensi tinggi.

3.2.6.4. Pengujian Keamanan

Pengujian keamanan yang dimaksud adalah pengujian aplikasi *backend* dari ancaman-ancaman berupa *SQL Injection* dan *Weak Authentication.* Pengujian keamanan akan menggunakan alat ReadyAPI dan dilakukan terhadap seluruh layanan web yang dimiliki oleh aplikasi *backend*.

3.2.7. Penyusunan Laporan Tugas Akhir

Tahap ini adalah tahapan menyatukan informasi-informasi yang didapatkan dari tahapan pengerjaan tugas akhir.

Halaman ini sengaja dikosongkan

BAB IV
ANALISIS DAN PERANCANGAN

4.1. Analisis

Analisis dilakukan untuk memastikan peracangan dan pengembangan aplikasi *backend* sesuai dengan prototipe sistem manaemen vendor yang menjadi luaran penelitian yang dilkukan oleh Bening pada tahun 2019.

4.1.1. Analisis Kebutuhan Fungsional

Analisis yang dilakukan adalah menganalisis prototipe sistem manajemen vendor untuk menjadi daftar kebutuhan fungsional. Sampel prototipe sistem manajemen vendor dapat dilihat di Lampiran B. Analisa ini memiliki luaran sebuah tabel kebutuhan fungsional yang dapat dilihat pada Tabel 4.1.

Tabel 4.1 Daftar Kebutuhan Fungsional

ID	*Role*	Kebutuhan Fungsional
FR1	Admin, Staf	Melakukan *Login*
FR2	Admin, Staf	Melakukan *Logout*
FR3	Admin, Staf	Melihat Akun Pengguna
FR4	Admin	Membuat Akun Pengguna
FR5	Admin	Memodifikasi Akun Pengguna
FR6	Admin	Menghapus Akun Pengguna
FR7	Admin, Staf	Melihat SPK
FR8	Admin, Staf	Membuat SPK
FR9	Admin, Staf	Memodifikasi SPK
FR10	Admin, Staf	Menghapus SPK
FR11	Admin, Staf	Melihat Vendor
FR12	Admin, Staf	Membuat Vendor
FR13	Admin, Staf	Memodifikasi Vendor
FR14	Admin, Staf	Menghapus Vendor

Pada Tabel 4.1 dapat diketahui bahwa terdapat 14 kebutuhan fungsional hasil analisis prototipe sistem manajemen vendor aplikasi manajemen vendor Giyomi.

4.1.2. Analisis Kebutuhan Layanan Web

Luaran analisis kebutuhan fungsional selanjutnya menjadi masukan analisis kebutuhan layanan web. Tabel daftar kebutuhan layanan web berdasarkan daftar kebutuhan fungsional dapat terlihat pada **Error! Reference source not found.**.

Tabel 4.2 Daftar Kebutuhan Layanan Web

ID	ID FR	Layanan Web
API1	FR1	*Login* Akun pengguna
API2	FR2	*Logout* Akun Pengguna
API3	FR3	Melihat daftar akun pengguna
API4	FR3	Melihat akun pengguna
API5	FR4	Membuat akun pengguna
API6	FR5	Memodifikasi akun pengguna
API7	FR6	Menghapus akun pengguna
API8	FR7	Melihat daftar SPK(Surat Perintah Kerja)
API9	FR7	Melihat daftar SPK berdasarkan status
API10	FR7	Melihat satu SPK
API11	FR8	Membuat SPK dengan status tertentu
API12	FR8	Mengunggah foto produk
API13	FR8	Melihat daftar proses
API14	FR8	Membuat proses
API15	FR8	Melihat daftar satuan panjang kain
API16	FR8	Membuat satuan panjang kain
API17	FR8	Melihat daftar ukuran
API18	FR8	Membuat ukuran
API19	FR8	Melihat daftar varian

API20	FR8	Membuat varian
API21	FR9	Memodifikasi status SPK
API22	FR9	Memodifikasi proses SPK
API23	FR10	Menghapus SPK
API24	FR11	Melihat daftar Vendor
API25	FR11	Melihat Vendor
API26	FR12	Membuat Vendor
API27	FR13	Memodifikasi Vendor
API28	FR14	Menghapus Vendor

Pada Tabel 4.2 dapat diketahui bahwa terdapat sejumlah 28 layanan web untuk memenuhi 14 kebutuhan fungsional. Untuk memenuhi setiap kebutuhan fungsinal diperlukan satu atau lebih dari satu layanan web. Dengan adanya layanan web untuk melakukan *log in*, setiap pengguna yang mengakses terhadap layanan web harus teridentifikasi identitas dan hak-hak nya.

4.2. Perancangan

Pada tahap perancangan, penulis membuat rancangan-rancangan yang diperlukan luarannya untuk tahap pengembangan. Perancangan yang dilakukan disesuaikan dengan luaran yang didapatkan pada tahap analisis.

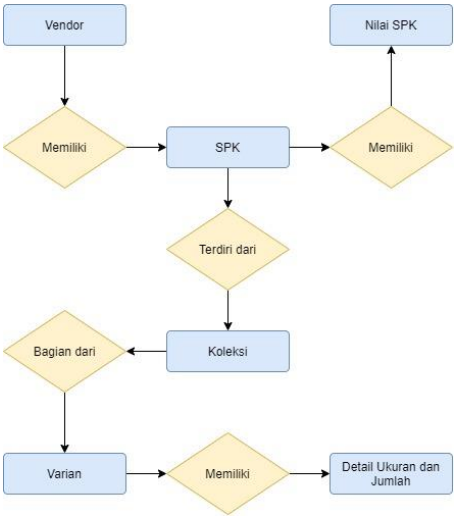
4.2.1. Perancangan Model Data

Perancangan pertama yang dilakukan adalah perancangan model data. Sejumlah 2 perancangan model data yang dilakukan yaitu perancangan model data konseptual dan perancangan data logis. Perancangan model data logis akan dilakukan jika perancangan model data konseptual telah selesai.

4.2.1.1. Perancangan Model Data Konseptual

Perancangan model data konseptul dilakukan dengan mengamati daftar kebutuhan layanan web dan prototipe sistem manajemen vendor. Terdapat beberapa entitas yang telah

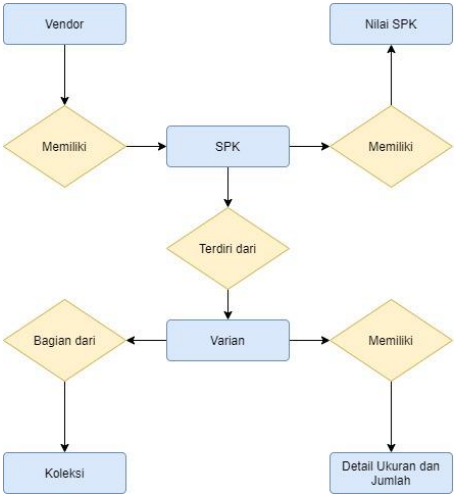
teridentifikasi pada tahapan perancangan model data konseptual yaitu Vendor, SPK (Surat Perintah Kerja), Nilai SPK, Varian, Koleksi serta Detail Ukuran dan Jumlah. Diagram rancangan model data konseptual versi pertama dapat terlihat pada Gambar 4.1.



Gambar 4.1 Diagram Rancangan Model Data Konseptual Versi 1

Gambar 4.1 dirancang berdasarkan kebutuhan entitas untuk memenuhi kebutuhan layanan web. Pada tahapan perancangan model data konseptual juga dilakukan proses validasi hasil rancangan model data konseptual secara langsung dengan pengguna. Penulis melakukan proses validasi model data konseptual pada tanggal 26 Februari 2020. Proses validasi tersebut dilakukan dengan aktifitas penulis memperlihatkan diagram rancnagan model data konseptual versi 1 dan menjelaskan diagram tersebut serta mengharapkan *feedback* dari pengguna. Proses validasi tersebut menghasilkan kesimpulan bahwa diagram rancangan model data versi 1 belum sepenuhnya valid menurut pengguna.

Selanjutnya penulis melakukan perbaikan diagram rancangan model data konseptual sesuai dengan *feedback* pengguna menjadi diagram rancangan model data konseptual versi 2 seperti yang terlihat pada Gambar 4.2 diagram rancangan model data konseptual versi 2 tersebut kemudian telah tervalidasi oleh pengguna.



Gambar 4.2 Diagram Rancangan Model Data Konseptual Versi 2

4.2.1.2. Perancangan Model Data Logis

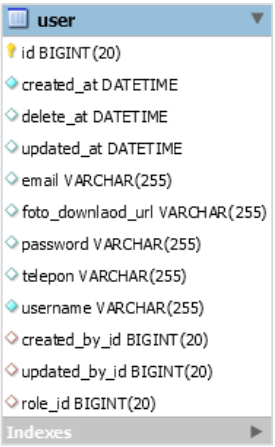
Perancangan model data logis berdasarkan pada hasil rancangan model data konseptual yang selanjutnya dikembangkan lebih detail menuju definisi-definisi tabel dan kolom pada tabel tersebut. Untuk memenuhi kebutuhan sistem sesuai dengan perancangan model data konseptual diperlukan sejumlah 21 tabel yang dirancang pada perancangan model data logis. Proses normalisasi juga dilakukan untuk membuat model data yang dirancang menjadi senormal mungkin. Setiap tabel yang dirancang pada perancangan model data logis memiliki kolom-kolom umum yang dimiliki oleh seluruh tabel yang ada. Kolom-kolom tersebut meliputi:

1. *Created At* (Nama kolom: created_at)
   Kolom yang memiliki tipe data tanggal dan menerangkan kapan rekaman tabel terkait dibuat oleh pengguna.
2. *Updated At* (Nama kolom: updated_at)
   Kolom yang memiliki tipe data tanggal dan menerangkan kapan rekaman tabel terkait diubah oleh pengguna.
3. *Deleted At* (Nama kolom: deleted_at)
   Kolom yang memiliki tipe data tanggal dan menerangkan kapan rekaman tabel terkait dihapus oleh pengguna.
4. *Created By* (Nama kolom: created_by)
   Kolom yang memiliki tipe data angka sebagai *foreign key* menuju tabel *User* dan digunakan sebagai untuk aktifitas identifikasi siapa pengguna yang membuat rekaman pada tabel.
5. *Updated By* (Nama kolom: updated_by)
   Kolom yang memiliki tipe data angka sebagai *foreign key* menuju tabel *User* dan digunakan sebagai untuk aktifitas identifikasi siapa pengguna yang mengubah rekaman pada tabel.

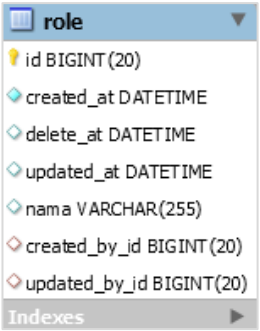
Hasil perancangan model data logis secara menyeluruh berupa diagram rancangan model data logis dapat dilihat pada Lampiran C. Berikut adalah penjabaran dari masing masing tabel pada hasil rancangan model data logis. Setiap tabel pada rancangan model data logis ini memiliki nama kolom yang dilengkapi dengan tipe data dan panjang maksimal kolom.

1. *User* dan *Role*

Tabel *user* digunakan untuk menyimpan data-data pengguna seperti *email*, tautan foto, nomor telepon, *username* dan *password* keamanan pengguna. Sedangkan tabel *Role* digunakan untuk menyimpan peran pengguna dalam sistem. Setiap pengguna memiliki 1 peran dalam sistem.



(a)

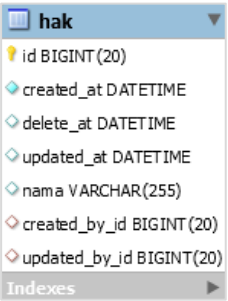


(b)

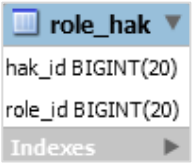
Gambar 4.3 (a) Rancangan Model Data Logis User
Gambar 4.4 (b) Rancangan Model Data Logis Role

2. Hak dan *Role* Hak

Tabel Hak menyimpan wewenang peran dalam sistem. Sebuah peran bisa memiliki banyak wewenang dan sebuah wewenang bisa dimiliki oleh banyak peran sekaligus. Wewenang didefinisikan sebagai sebuah layanan web. Hubungan banyak ke banyak antara tabel Peran dengan tabel Hak dilakukan dengan membuat tabel relasi *Role* Hak.



(a)

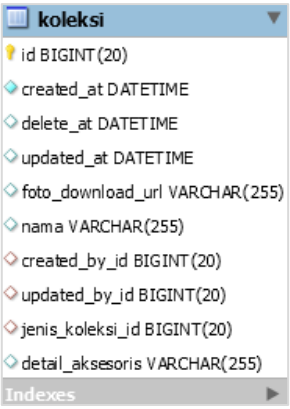


(b)

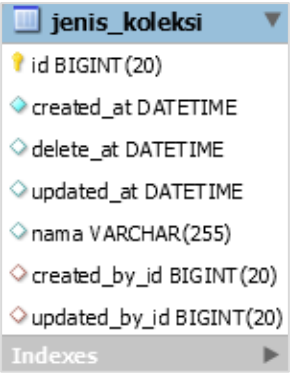
Gambar 4.5 (a) Rancangan Model Data Logis Hak
Gambar 4.6 (b) Rancangan Model Data Logis Role Hak

3. Koleksi dan Jenis Koleksi

Tabel Koleksi merepresentasikan koleksi dalam sistem (Contoh nama koleksi: “Amira *Collection*”). Data yang tersimpan pada tabel ini adalah nama dari sebuah koleksi dan tautan foto dari koleksi tersebut. Tabel Koleksi terhubung dengan tabel Jenis Koleksi yang mendefinisikan jenis dari koleksi. (Contoh nama jenis koleksi; “Atasan”). Data jenis koleksi yang disimpan adalah nama jenis koleksi.



(a)

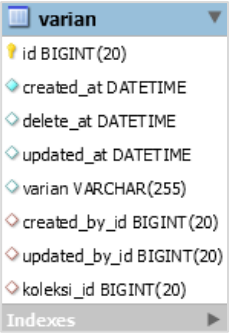


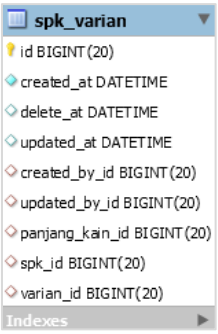
(b)

Gambar 4.7 (a) Rancangan Model Data Logis Koleksi
Gambar 4.8 (b) Rancangan Model Data Logis Jenis Koleksi

4. Varian dan SPK Varian

Tabel Varian merepresentasikan varian pada SPK (contoh nama varian:”Merah”). Data yang disimpan pada tabel varian adalah nama varian. Setiap varian dimiliki oleh sebuah koleksi. Kemudian Tabel SPK Varian merepresentasikan tabel relasi antara tabel SPK dan tabel Varian untuk memenuhi kebutuhan hubungan satu ke banyak. Setiap SPK dapat memiliki satu atau banyak varian. Pada tabel SPK Varian juga memiliki data terkait panjang kain yang dibutuhkan untuk memproses varian terkait.



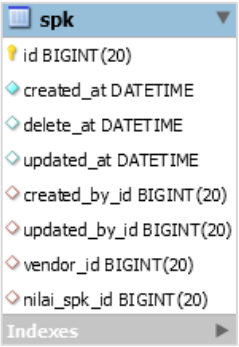


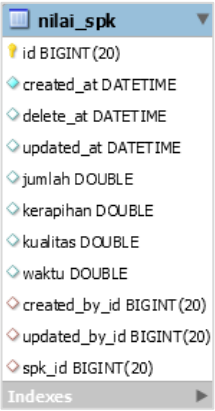
(a) (b)

Gambar 4.9 (a) Rancangan Model Data Logis Varian
Gambar 4.10 (b) Rancangan Model Data Logis SPK Varian

5. SPK dan Nilai SPK

Tabel yang merepresentasikan sebuah SPK. Sebuah SPK memiliki satu vendor dan nilai vendor untuk SPK tersebut. Penilaian terhadap vendor yang terkait dalam sebuah SPK disimpan dalam sebuah Tabel Nilai SPK. Nilai tersebut meliputi ketepatan jumlah, kerapihan, kualitas dan ketepatan waktu penyelesaian. Seluruh nilai disimpan dengan bentuk presentase.



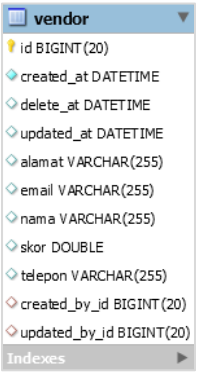


(a) (b)

Gambar 4.11 (a) Rancangan Model Data Logis SPK
Gambar 4.12 (b) Rancangan Model Sata Logis Nilai SPK

6. Vendor

Setiap SPK yang dibuat ditujukan kepada satu vendor tertentu. Data vendor yang disimpan meliputi alamat, *email*, nama, skor dan nomor telepon vendor. Tabel Vendor memiliki relasi satu ke banyak dengan tabel SPK.

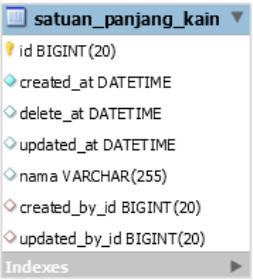


Gambar 4.13 Rancangan Model Data Logis Vendor

7. Panjang Kain dan Satuan Panjang Kain

Tabel Panjang kain meyimpan data panjang kain yang dibutuhkan untuk memproses satu varian dari sebuah SPK. Setiap Panjang kain memiliki satuan, untuk kebutuhan tersebut tabel Panjang Kain akan berelasi dengan tabel Satuan Panjang Kain (contoh satuan Panjang kain: “Meter”, “Yard”, “Kg”).

(a)

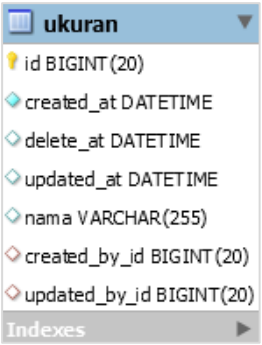


(b)

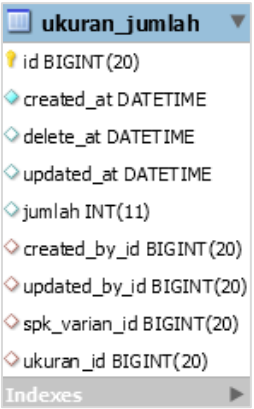
Gambar 4.14 (a) Rancangan Model Data Logis Panjang Kain
Gambar 4.15 (b) Rancangan Model Data Logis Satuan Panjang Kain

8. Ukuran dan Jumlah Ukuran

Tabel Ukuran merepresentasikan ukuran sebuah varian (contoh nama ukuran: “S”,”M”, “XL”). Data yang disimpan pada tabel Ukuran adalah nama dari ukuran tersebut. Tabel Ukuran Jumlah berelasi banyak ke banyak antara tabel Ukuran dan SPK Varian. Selain mendefinisikan relasi tersebut, tabel Ukuran Jumlah juga menyimpan jumlah dari sebuah ukuran terhadap SPK Varian (contoh representasi tabel Ukuran jumlah terhadap tabel SPK Varian: “Baju Amera Merah dengan ukuran XL berjumlah 20”).



(a)

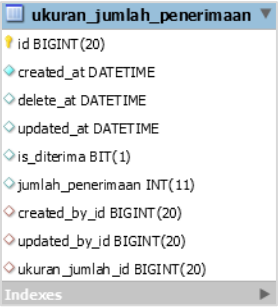


(b)

Gambar 4.16 (a) Rancangan Model Data Logis Ukuran
Gambar 4.17 (b) Rancangan Model Data Logis Ukuran Jumlah

9. Ukuran Jumlah Penerimaan

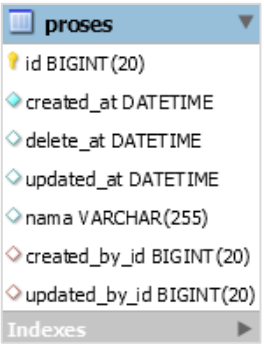
Tabel ukuran jumlah penerimaan menyimpan data penerimaan barang dari vendior untuk sebuah ukuran pada varian dari sebuah SPK. Prototipe sistem manajemen vendor mengharuskan sistem untuk memfasilitasi penerimaan secara parsial untuk ukuran pada sebuah varian (contoh penerimaan parsial:”Pengguna 1 menerima 10 dan menolak 5 baju Amrea Merah XL dari total rencana produksi 20 baju Amrea Merah XL pada tanggal 20 Mei 2020”).



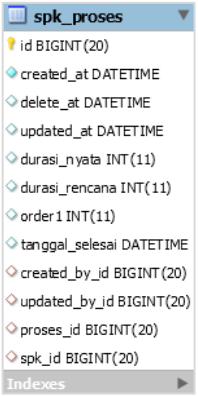
Gambar 4.18 Rancangan Model Data Logis Ukuran Jumlah Penerimaan

10. Proses dan SPK Proses

Tabel Proses merepresentasikan proses yang harus dilakukan vendor untuk menyelesaikan sebuah SPK. Data yang disimpan pada tabel Proses adalah nama dari proses terkait (contoh nama proses: “Jahit”). Lalu terdapat Tabel relasi antara tabel SPK dan tabel Proses untuk memenuhi kebutuhan hubungan satu ke banyak. Pada tabel SPK Proses juga memiliki data durasi target proses dan durasi realisasinya serta urutan dari proses relatif terhadap proses yang lain dalam satu SPK. Pada tabel SPK Proses juga terdapat tanggal perkiraan selesainya SPK terkait.



(a)

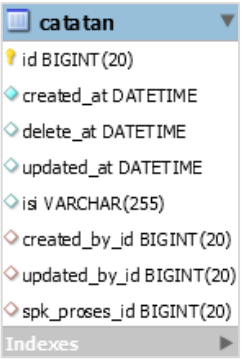


(b)

Gambar 4.19 (a) Rancangan Model Data Logis Proses
Gambar 4.20 (b) Rancangan Model Data Logis SPK Proses

11. Catatan

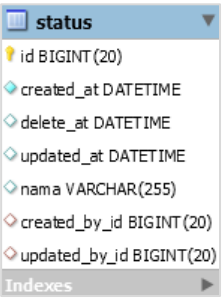
Tabel Catatan menyimpan catatan dari pengguna untuk sebuah proses pada sebuah SPK. Tabel Catatan ini berelasi banyak ke satu dengan SPK Proses. Data yang disimpan pada tabel Catatan adalah isi dari catatan untuk proses pada SPK tersebut.



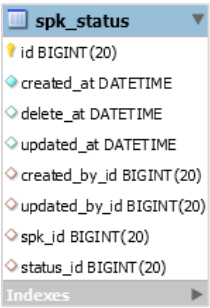
Gambar 4.21 Rancangan Model Data Logis Catatan

12. Status dan SPK Status

Tabel Status menyimpan status yang mungkin melekat pada sebuah SPK (contoh nama status:”Draft”,”Valid”). Kemudian tabel ini berelasi satu ke banyak antara tabel SPK dengan tabel Status. Satu SPK bisa pernah memiliki banyak status, namun hanya memiliki satu status aktif yaitu status terakhir yang pernah melekat pada SPK tersebut.



(a)



(b)

Gambar 4.22 (a) Rancangan Model Data Logis Status
Gambar 4.23 (b) Rancangan Model Data Logis SPK Status

4.2.2. Perancangan Layanan Web

Perancangan layanan web yang dilakukan terdiri dari perencanaan versi layanan, definisi dari sumber daya atau luaran yang akan dihasilkan dari layanan dan metode layanan. Hasil perancangan layanan web adalah berupa tabel daftar layanan web yang terllihat pada tabel. Daftar layanan yang ada pada tabel tersebut sesuai dengan luaran analisis kebutuhan layanan web, terdapat definisi luaran untuk masing-masing layanan web dan HTTP Method atau yang dapat disebut dengan label metode protocol yang akan digunakan dalam pembuatan layanan web [12].

Tabel 4.3 Daftar rancangan layanan web

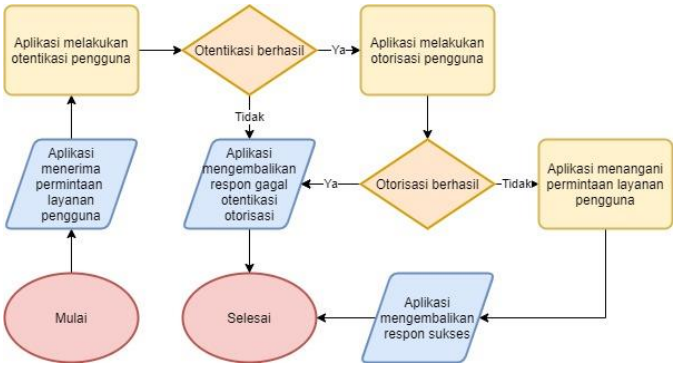
ID	Layanan Web	Luaran	HTTP Method
API1	Login	JWT pengguna	POST
API2	Logout	-	GET
API3	Melihat daftar akun pengguna	Daftar seluruh akun pengguna	GET
API4	Melihat akun pengguna	Satu akun pengguna yang diminta	GET
API5	Membuat akun pengguna	Satu akun pengguna yang telah dibuat	POST
API6	Memodifikasi akun pengguna	Satu Akun pengguna yang telah dimodifkasi	PUT
API7	Menghapus akun pengguna	-	DELETE
API8	Melihat daftar SPK	Daftar seluruh SPK	GET
API9	Melihat daftar SPK berdasarkan status	Daftar seluruh SPK dengan status tertentu	GET

API10	Melihat satu SPK	Satu SPK yang diminta	GET
API11	Membuat SPK dengan status tertentu	Satu SPK yang telah dibuat	POST
API12	Mengunggah foto koleksi	Tautan untuk mengunduh foto koleksi	POST
API13	Melihat daftar proses	Daftar seluruh proses	GET
API14	Membuat proses	Satu proses yang telah dibuat	POST
API15	Melihat daftar satuan panjang kain	Daftar seluruh satuan panjang kain	GET
API16	Membuat satuan panjang kain	Satu satuan Panjang kain yang telah dibuat	POST
API17	Melihat daftar ukuran	Daftar seluruh ukuran	GET
API18	Membuat ukuran	Satu ukuran yang dibuat	POST
API19	Melihat daftar varian	Daftar seluruh varian	GET
API20	Membuat varian	Satu varian yang dibuat	GET
API21	Memodifikasi status SPK	Satu status SPK yang telah dimodifikasi	PUT
API22	Memodifikasi proses SPK	Satu proses SPK yang telah dimodifikasi	PUT
API23	Menghapus SPK	-	DELETE
API24	Melihat daftar Vendor	Daftar seluruh vendor	GET

API25	Melihat Vendor	Satu vendor yang diminta	GET
API26	Membuat Vendor	Satu vendor yang telah dibuat	GET
API27	Memodifikasi Vendor	Satu vendor yang telah dimodifikasi	PUT
API28	Menghapus Vendor	-	DELETE

4.2.3. Perancangan Kebijakan Otentikasi dan Otorisasi

Hasil perancangan kebijakan otentikasi dan otorisasi adalah diaram alir rancangan otentikasi dan otorisasi pengguna layanan web serta pemetaan hak dengan *role* (aktor) pengguna layanan web. Otentikasi dan otorisasi dilakukan secara berurutan yaitu otentikasi dan dilanjutkan dengan autentikasi sesuai dengan hasil penelitian yang dilakukan oleh Madhuri pada tahun 2010 [10]. Diagram alir perancangan kebijakan otorisasi dan otentikasi dapat dilihat pada Gambar 4.24



Gambar 4.24 Diagram alir rancangan otentikasi dan otorisasi pengguna layanan

Pada Gambar 4.24 tersebut, otentikasi dilakukan dengan mengidentifikasi data pengguna yang dikirim dalam bentuk *Json Web Token* (JWT) yang dikirimkan oleh pengguna dalam

setiap permintaan layanan web. Untuk mendapatkan JWT pengguna, pengguna harus melakukan permintaan layanan *login* dengan mengikutsertakan *username* dan *password* pengguna. Jika otentikasi tidak berhasil, maka aplikasi *backend* akan mengembalikan respon gagal otentikasi otorisasi. Jika otentikasi berhasil maka aplikasi akan berlanjut menuju proses otorisasi pengguna.

Otorisasi dilakukan setelah otentikasi pengguna berhasil dilakukan. Otorisasi dengan mengidentifikasi hak pengguna dari *role* pengguna. Jika pengguna memiliki hak untuk meminta layanan terkait, maka permintaan layanan akan dilanjutkan menuju penanganan permintaan layanan sesuai layanan yang diminta dan mengembalikan respon sukses. Jika pengguna tidak memiliki hak untuk melakukan permintaan tehadap layanan terkait maka aplikasi *backend* akan mengembalikan respon gagal otentikasi otorisasi.

Berdasarkan hasil analisis kebutuhan fungsional dan analisis kebutuhan layanan web, dibuat daftar hak-hak masing masing *role* pengguna layanan. Pemetaan hak dengan *role* pengguna layanan web dapat terlihat pada Tabel 4.4.

Tabel 4.4 Pemetaan layanan web yang berhak diminta oleh *role*

Role	Layanan Web
Admin	API1, API2, API3, API4, **API5, API6, API7**, API8, API9, API10, API11, API12, API13, API14, API15, API16, API17, API18, API19, API20, API21, API22, API23, API24, API25, API26, API27, API28
Staf	API1, API2, API3, API4, API8, API9, API10, API11, API12, API13, API14, API15, API16, API17, API18, API19, API20, API21, API22, API23, API24, API25, API26, API27, API28

Dapat terlihat dapat Tabel 4.4 bahwa *role* Admin berhak meminta seluruh layanan web, sedangankan aktor Staf berhak meminta seluruh layanan web kecuali layanan web dengan id API5, API6, API7.

BAB V
PENGEMBANGAN DAN PENGUJIAN

5.1. Pengembangan

Tahap pengembangan adalah tahap implementasi luaran tahap perancangan melalui kode program Java dan kerangka kerja Spring Boot. Untuk menuliskan kode program Java dengan kerangka kerja Spring Boot, penulis membutuhkan pemahaman implementasi kode program Java dengan Spring Boot dan kemampuan bahasa pemrograman Java tingkat lanjut [29]. Terdapat 4 pengembangan yang penulis lakukan, yaitu pengembangan model data, pengembangan layanan web, pengembangan kebijakan otentikasi dan otorisasi serta pengembangan dokumentasi layanan web.

5.1.1. Pengembangan Model Data

Implementasi model data akan menggunakan *Java Persistence* API (JPA) yang telah tersedia pada kerangka kerja Spring Boot. Penggunaan JPA pada Spring Boot dimulai dengan mendaftarkan *dependency* JPA pada *depencency manager* Spring Boot. *Platform* Maven digunakan sebagai *depencency manager*. Kode program untuk mendaftarkan *dependency* JPA dengan menggunakan *platform* Maven dapat terlihat pada Kode 5.1

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Kode 5.1 Mendaftarkan dependency JPA pada aplikasi Spring Boot

Pendeklarasian model data menggunakan JPA dilakukan dengan membuat *class* Java yang mengimplementasikan *interface* JPARepository dengan parameter *generic class* model entitas. Oleh karena itu, penulisan kode program untuk membuat *class* entitas sesuai dengan rancangan model data

logis yang telah dibuat. Daftar entitas yang dibuat sebelum implementasi JPA dapat diterangkan oleh Tabel 5.1.

Tabel 5.1 Daftar *class* entitas

No	Nama Class Entitas	Deskripsi
1	Catatan	Catatan pada setiap proses pada SPK.
2	Hak	Hak yang melekat pada suatu *user role.*
3	JenisKoleksi	Jenis dari sebuah koleksi. (Contoh: Atasan, Bawahan)
4	Koleksi	Koleksi yang dimiliki sejumlah 1 oleh setiap SPK.
5	PanjangKain	Panjang kain yang memiliki satuan terdefinisi oleh entitas SatuanPanjangKain.
6	Proses	Proses yang mungkin akan dilakukan pada suatu SPK. (Contoh: Jahit, *Finishing*)
7	Role	*Role* yang dimiliki oleh setiap entitas *User*.
8	SatuanPanjang Kain	Satuan yang digunakan pada entitas PanjangKain.
9	SPK	Entitas yang merepresentasikan data dari sebuah SPK
10	SPKProses	Proses yang telah melekat pada sebuah SPK.
11	SPKStatus	Status yang telah melekat pada sebuah SPK.
12	SPKVarian	Varian yang melekat pada sebuah SPK.
13	Status	Kemungkinan status yang sedang dimiliki oleh SPK. (Contoh: *Draft*, Menunggu di Validasis)

14	Ukuran	Variasi ukuran yang mungkin ada pada sebuah varian. (Contoh: S, M, XL)
15	UkuranJumlah	Jumlah dari sebuah ukuran yang telah melekat para varian SPK.
16	User	Pengguna aplikasi yang memiliki sebuah Role.
17	Varian	Varian yang mungkin melekat pada sebuah SPK dari sebuah koleksi. (Contoh: Merah, Corak Batik)
18	Vendor	Vendor yang ditugaskan pada setiap SPK.

Pada Tabel 5.1 terlihat bahwa terdapat sejumlah 18 *class* entitas yang dibuat untuk memenuhi kebutuhan sistem yang berasal dari luaran perancnagan model data. Seluruh entitas tersebut kemudian didefinisikan dengan program Java. Kode 5.2 memberikan contoh implementasi *class* entitas *User*.

```
@Entity
public class User extends AuditModel<User>{
    @Column(nullable = false, unique = true)
    private String username;
    private String telepon;
    private String password;
    private String email;
    @Column(name = "foto_downlaod_url")
    private String fotoDownloadUrl;
    @ManyToOne
    private Role role;
    public User(String username, String password,
                String email, String telepon,
                String fotoDownloadUrl, Role role) {
        this.role = role;
        this.username = username;
        this.password = password;
        this.email = email;
        this.telepon = telepon;
        this.fotoDownloadUrl = fotoDownloadUrl;
    }
```

Kode 5.2 Mendefinisikan *class* entitas *User*

Kode 5.2 mendeklarasikan bahwa entitas *User* memiliki beberapa atribut seperti "*username*", "telepon", "*password*", "*email*", "fotoDownloadUrl" dan "role". Satu *class* entitas *User* tersebut akan terdefinisi sebagai satu tabel pada basis data sistem. *Class* entitas yang telah dibuat selanjutnya digunakan sebagai generic type pada *class* JPARepository agar JPA sepenuhnya dapat digunakan. Contoh penggunaan *class* JPARepository dengan memasukan *class* entitas User sebagai *generic type* yang akan menghasilkan sebuah sebuah *repository* untuk entitas *User* dapat dilihat pada Kode 5.3

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
    List<User> findAllByRoleOrderByCreatedAt(Role role);

}
```

Kode 5.3 Mendefinisikan *interface* JPARepository dengan menggunakan *User* sebagai *generic type*

Pada Kode 5.4 terdeklarasi sebuah *repository* entitas User yang dapat melakukan pencarian entitas User dengan menggunakan kata kunci username dan dapat mencari entitas User dengan *role* tertentu.

5.1.2. Pengembangan Layanan Web

Pengembangan layanan web dilakukan dengan pembuatan REST *controller class* Java dan mengikutsertakan anotasi REST *Controller* pada *class* Java tersebut. Setiap *method* pada *rest controller class* akam merepresentasikan satu layanan web yang dapak dikonsumsi melalui *protocol* http. Setiap *method* pada *rest controller class* harus ditambah sebuah anotasi yang mendefinisikan *path* dan http *method* dari layanan yang akan dibuat. Daftar *class controller* sesuai dengan rancangan layanan web dapat terlihat pada Tabel 5.2. Pada tabel tersebut berisi daftar *class controller* yang dibuat untuk memenuhi kebutuhan layanan web sesuai dengan luaran perancangan layanan web. Terdapat dua jenis *controller* dalam daftar tersebut, antara lain:

1. *RESTful Controller*
   Restful Controller adalah *controller* yang mempunyai paket fungsi untuk melakukan pembuatan entitas baru (*Create*), membaca entitas (*Read*), modifikasi entitas (*Update*), dan menghapus entitas (*Delete*). Paket fungsi tersebut wajarnya disingkat CRUD.
2. *Rest Controller*
   Rest Controller adalah *controller* selain *Restful Controller*

Tabel 5.2 Daftar *class controller*

No	Nama *Class Controller*	Deskripsi
1	RestAuth	Layanan autentikasi *user*
2	RestCdn	Layanan penyimpanan media gambar SPK
3	RestSPK	Layanan pembuatan dan melihat SPK beserta data-data SPK terkait
4	RestUser	Layanan melihat data-data *user*
5	RestfulCatatan	Layanan CRUD entitas catatan.
6	RestfulJenisKoleksi	Layanan CRUD entitas Jenis Koleksi
7	RestfulKoleksi	Layanan CRUD entitas Koleksi
8	RestfulPanjangKain	Layanan CRUD entitas Panjang Kain
9	RestfulProses	Layanan CRUD entitas Proses
10	RestfulSatuanPanjangKain	Layanan CRUD entitas Satuan Panjang Kain
11	RestfulSPK	Layanan CRUD entitas SPK
12	RestfulSPKProses	Layanan CRUD entitas SPK Proses

13	RestfulSPKStatus	Layanan CRUD entitas SPK Status
14	RestfulStatus	Layanan CRUD entitas Status
15	RestfulUkuran	Layanan CRUD entitas Ukuran
16	RestfulUkuranJumlah	Layanan CRUD entitas Ukuran Jumlah
17	RestfulUkuran JumlahPenerimaan	Layanan CRUD entitas Ukuran Jumlah Penerimaan
18	RestfulVarian	Layanan CRUD entitas Varian
19	RestfulVendor	Layanan CRUD entitas Vendor

Controller Class yang dibuat harus menyertakan anotasi-anotasi Spring Boot agar terdeteksi sebagai layanan web. Contoh implementasi *Controller Class* pada program java dapat terlihat pada Kode 5.4

```
@RestController
@RequestMapping(value = "/restful/catatans")
public class RestfulCatatanController
        implements RestfulController<CatatanRestfulModel, Catatan> {

    private CatatanRepository catatanRepository;
    private SpkProsesRepository spkProsesRepository;

    public RestfulCatatanController(CatatanRepository catatanRepository,
                                    SpkProsesRepository spkProsesRepository) {
        this.catatanRepository = catatanRepository;
        this.spkProsesRepository = spkProsesRepository;
    }

    @Override
    @GetMapping(value = "/{id}")
    public ResponseEntity<CatatanRestfulModel> getSingle(Long id) {
        return ResponseEntity.ok(new CatatanRestfulModel(findById(id)));
    }
```

Kode 5.4 *Class Controller* RestfulCatatan

5.1.3. Pengembangan Kebijakan Otentikasi dan Otorisasi

Otentikasi dan otorisasi pengguna dilakukan dengan menggunakan JWT yang didapatkan oleh pengguna dengan meminta layanan web login. JWT dibuat dengan kunci rahasia dan *username* pengguna serta dibangun dengan algoritma HS512. Kode program java yang digunakan untuk membuat JWT dapat terlihat pada Kode 5.5 yang telah dibuat selanjutnya dikembalikan kepada pengguna untuk digunakan pada setiap permintaan layanan.

```
private String doGenerateToken(Map<String, Object> claims, String subject) {
    return Jwts.builder().setClaims(claims).setSubject(subject)
            .setIssuedAt(new Date(System.currentTimeMillis()))
            .signWith(SignatureAlgorithm.HS512, secret).compact();
}
```

Kode 5.5 Membuat JWT

Setiap permintaan layanan yang dilakukan oleh pengguna wajib mengikutsertakan JWT yang telah didapatkan agar aplikasi *backend* dapat melakukan otentikasi dan otorisasi. Otentikasi dilakukan dengan melakukan ekstraksi *username* pengguna dari JWT pengguna. Kode program java untuk melakukan ekstraksi username pengguna dari JWT dapat terlihat pada Kode 5.6 Jika username ekstraksi *username* berhasil dilakukan dan *username* pengguna terdaftar pada sistem, maka otentikasi pengguna berhasil dilakukan.

```
public String getUsernameFromToken(String token) {
    return getClaimFromToken(token, Claims::getSubject);
}
```

Kode 5.6 Melakukan ekstraksi username pengguna dari JWT

Otorisasi dilakukan dengan identifikasi hak pengguna dari *role* pengguna dan melakukan pemetaan *role* pengguna dengan setiap layanan web. Kode program untuk melakukan pemetaan *role* dengan layanan web dapat terlihat pada Kode 5.7. Kode

program tersebut mendefinisikan bahwa hanya pengguna yang memiliki *role* dengan nama "*admin*" yang dapat melakukan pembuatan pengguna baru, memodifikasi pengguna dan menghapus pengguna.

```
@Override
protected void configure(HttpSecurity httpSecurity) throws Exception {
    httpSecurity
            .csrf() CsrfConfigurer<HttpSecurity>
            .disable() HttpSecurity
        .authorizeRequests() ExpressionUrlAuthorizationConfigurer<H>.ExpressionInterceptUrlRegistry
            .antMatchers( ...antPatterns: "/rest/auth") ExpressionUrlAuthorizationConfigurer<H>.AuthorizedUrl
            .permitAll() ExpressionUrlAuthorizationConfigurer<H>.ExpressionInterceptUrlRegistry
        .antMatchers(HttpMethod.POST, ...antPatterns: "/rest/users").hasAnyRole( ...roles: "admin")
        .antMatchers(HttpMethod.PUT, ...antPatterns: "/rest/users").hasAnyRole( ...roles: "admin")
        .antMatchers(HttpMethod.DELETE, ...antPatterns: "/rest/users").hasAnyRole( ...roles: "admin")
```

Kode 5.7 Melakukan pemetaan role dengan layanan web

Jika otorisasi pengguna berhasil dilakukan dan pengguna memiliki *role* yang berhak untuk melakukan permintaan terhadap layanan terkait, maka permintaan layanan dari pengguna akan dilanjutkan penanganannya sesuai layanan yang diminta. Setelah pengembangan kebijakan otentikasi dan otorisasi selesai dilakukan maka telah terbentuk aplikasi *backend* yang menjadi salah satu luaran tahapan pengembangan.

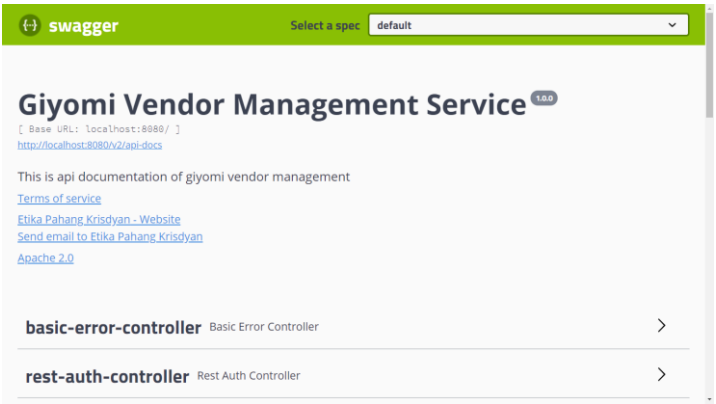
5.1.4. Pengembangan Dokumentasi Layanan Web

Dokumentasi API dikembangkan dengan menggunakan fasilitas yang diberikan oleh kerangka kerja Spring Boot yaitu integrasi dengan *platform* dokumentasi layanan web Swagger. Integrasi antara kerangka kerja Spring Boot dengan Swagger dapat dilakukan dengan pembuatan *class* java untuk konfigurasi dokumentasi layanan web dengan Swagger. Kode program *class* java untuk konfigurasi dokumentasi layanan web dengan Swagger dapat terlihat pada Kode 5.8

```
@Bean
public Docket apiDocket() {
    Contact contact = new Contact(CONTACT_NAME, CONTACT_URL, CONTACT_EMAIL);
    return new Docket(DocumentationType.SWAGGER_2)
            .apiInfo(new ApiInfo(TITLE, DESC, VERSION, TERM_URL, contact,
                    LICENCE, LICENCE_URL, Collections.emptyList()))
            .globalOperationParameters(Collections.singletonList(
                    new ParameterBuilder()
                            .name("Authorization")
                            .modelRef(new ModelRef( type: "string"))
                            .parameterType("header")
                            .required(true)
                            .hidden(true)
            .build()
    ));
}
```

Kode 5.8 Method dalam *class* konfigurasi Swagger

Konfigurasi Swagger pada kerangka kerja Spring Boot tersebut akan menghasilkan sebuah tampilan web yang dapat diakses melalui *browser* berisi dokumentasi layanan web. Gambar 5.1 menunjukan dokumentasi layanan web yang dihasilkan.



Gambar 5.1 Dokumentasi layanan web

Pada Gambar 5.1 terlihat bahwa aplikasi *backend* memiliki *base url* http://localhost:8080. Gambar tersebut adalah tangkapan layar dari dokumentasi aplikasi *backend* yang telah dibuat dan diakses melalui *browser*.

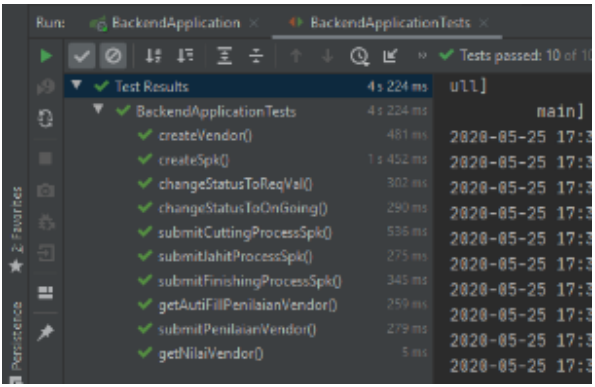
5.2. Pengujian

Tahap pengujian dilakukan adalah pengujian unit, pengujian fungsional, pengujian performa dan pengujian keamanan. Tahap pegujian akan menghasilkan luaran berupa dokumentasi pengujian. Seluruh tahap pengujian dilakukan menggunakan komputer dengan spesifikasi sebagai berikut:

a. *Processor* : Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz (8 CPUs), ~2.6GHz
b. *Memory* (RAM): 8GB
c. Sistem Operasi : Windows 10 Education 64-bit
d. *Graphic Card* : Nvidia GeForce 940m

5.2.1. Pengujian Unit

Pada tahap ini merupakan tahapan dimana kode program Java yang telah dibuat diuji apakah kode program java telah memiliki luaran sesuai dengan kebutuhan. Tidak semua unit pada kode program diuji. Pengujian unit hanya dilakukan pada unit-unit yang memiliki fungsi penting. Pengujian unit dilakukan dengan membuat kode program Java dibantu dengan alat JUnit. Tangkapan layar hasil pengujia unit dapat dilihat pada Gambar 5.2



Gambar 5.2 Dokumentasi pengujian unit

Pada Gambar 5.2 tersebut terdapat simbol centang yang memberikan arti bahwa unit tersebut telah lolos uji. Dari sejumlah 10 unit yang diuji, seluruhnya telah berstatus lolos pengujian.

5.2.2. Pengujian Fungsional

Pengujian fungsional dilakukan dengan alat ReadyAPI yang telah memiliki fitur pengujian fungsional layanan web. Seluruh proses pengujian mengikuti proses-proses yang ada pada alat ReadyAPI. ReadyAPI memiliki fitur untuk membuat pengujian fungsionalitas secara otomatis dengan melakukan *import* dokumentasi Swagger. Setelah dokumentasi aplikasi *backend* tersambung dengan ReadyAPI kemudian dilakukan pemilihan layanan web yang akan menjadi target uji. Layanan web yang menjadi target uji pada pengujian fungsional ini adalah layanan web yang mendukung fungsionalitas fitur-fitur utama yaitu manajemen Surat Perintah Kerja (SPK), manajemen vendor dan manajemen akun pengguna. Dokumentasi pengujian fungsional dapat terlihat pada Gambar 5.3, Gambar 5.4, dan Gambar 5.5.

SMARTBEAR ReadyAPI

TestSuite Results Report for SPK Management

Result Metrics

Result Metrics

Start Time	Sun Jun 07 12:53:21 WIB 2020
End Time	Sun Jun 07 12:53:26 WIB 2020
Time Taken	4486 ms
TestCase Count	18
Failed TestCase Count	0
TestStep Count	45
Failed TestStep Count	0
Assertion Count	83
Failed Assertion Count	0

Gambar 5.3 Dokumentasi pengujian fungsional manajemen SPK

Dokumentasi pengujian fungsional pada yang terlihat pada Gambar 5.3 menunjukkan bahwa seluruh layanan web yang mendukung fungsionalitas fitur manajemen SPK telah lolos uji.

SMARTBEAR ReadyAPI

TestSuite Results Report for Vendor Management

Result Metrics

Result Metrics	
Start Time	Sun Jun 07 12:53:18 WIB 2020
End Time	Sun Jun 07 12:53:19 WIB 2020
Time Taken	891 ms
TestCase Count	4
Failed TestCase Count	0
TestStep Count	4
Failed TestStep Count	0
Assertion Count	15
Failed Assertion Count	0

Gambar 5.4 Dokumentasi pengujian fungsional manajemen vendor

Dokumentasi pengujian fungsional pada yang terlihat pada Gambar 5.4 menunjukkan bahwa seluruh layanan web yang mendukung fungsionalitas fitur manajemen vendor telah lolos uji.

SMARTBEAR ReadyAPI

TestSuite Results Report for User Management

Result Metrics

Result Metrics	
Start Time	Sun Jun 07 12:47:10 WIB 2020
End Time	Sun Jun 07 12:47:11 WIB 2020
Time Taken	901 ms
TestCase Count	4
Failed TestCase Count	0
TestStep Count	4
Failed TestStep Count	0
Assertion Count	8
Failed Assertion Count	0

Gambar 5.5 Dokumentasi pengujian fungsional manajemen akun pengguna

Dokumentasi pengujian fungsional pada yang terlihat pada Gambar 5.5 menunjukkan bahwa seluruh layanan web yang mendukung fungsionalitas fitur manajemen pengguna telah lolos uji. Dokumentasi lengkap pengujian fungsional dapat dilihat di Lampiran D.

5.2.3. Pengujian Performa

Pengujian performa dilakukan dengan alat ReadyAPI. Pengujian performa yang dilakukan adalah pengujian kecepatan respon layanan web dengan kompleksitas tertingi dan pengujian kecepetan respon layanan web dengan frekuensi permintaan tinggi secara simultan. Dua pengujian kecepatan tersebut menggunakan *best practice* yang telah disediakan dalam alat ReadyAPI.

Pengujian performa pertama berupa pengujian kecepatan respon layanan web dengan frekuensi tinggi menggunakan sampel layanan web melihat daftar SPK simbol identifikasi unik API8. Layanan web dengan kode unik API8 ditunjuk sebagai sampel karena layanan web tersebut memiliki kemungkinan permintaan frekuensi tinggi sesuai dengan prototipe sistem manajemen vendor.

Pengujian dilakukan dengan melakukan simulasi permintaan layanan web secara simultan dilakukan oleh 10 pengguna secara bersamaan untuk menguji kecepatan respon layanan web pada saat diminta oleh beberapa pengguna secara bersamaan. Penggunaan sejumlah 10 pengguna mengikuti rekomendasi alat ReadyAPI. Simulasi tersebut dilakukan selama 30 detik untuk mengetahui berbagai variasi kecepatan yang dihasilkan. Batas atas toleransi kecepatan pada pengujian pertama ini adalah 100 milidetik sesuai rekomendasi ReadyAPI. Dokumentasi pengujian performa pertama dapat dilihat pada Gambar 5.6

Test Step Metrics

LoadUI Single Request Test Case 2

Test step	Min (ms)	Max (ms)	Median (ms)	Last (ms)	Count	TPS	Err	Err %
Request 1	0	72	26	43	280	7	0	0
Test Case Level	7	72	28	43	330	9	0	0

Response time details

Test Step	Avg (ms)	75%	90%	95%	98%
Request 1	29	40.00	52.00	57.95	65.38
Test Case Level	30	41.00	53.00	58.45	65.18

Gambar 5.6 Dokumentasi pengujian performa 1

Gambar 5.6 memberikan informasi bahwa kecepatan respon tercepat pada angka 7 milidetik dan kecepatan respon terlama pada angka 72 milidetik serta rata-rata yaitu 30 milidetik. Dengan pengujian selama 30 detik dengan sejumlah 330 permintaan layanan web, layanan web memiliki sebaran kecepatan dibawah batas atas toleransi kecepatan yaitu 100 milidetik.

Pengujian performa kedua berupa pengujian kecepatan respon layanan web yang memiliki potensi kecepatan respon terlama atau memiliki kompleksitas tertinggi. Sesuai dengan kode program Java yang telah dibuat memenuhi kebutuhan layanan web, layanan web membuat SPK dengan sumbol identifikasi unik API11 adalah layanan web yang memiliki kompleksitas paling tinggi daripada layanan-layanan web yang lain. Pengujian perfora kedua ini dilakukan dengan melakukan simulasi permintaan layanan web API11 oleh satu pengguna selama 30 detik. Dikarenakan layanan web API11 adalah layanan web yang memiliki kompleksitas tinggi namun cenderung jarang diminta oleh pengguna, maka batas atas toleransi kecepatan pada pengujian kedua ini adalah 1000 milidetik (1 detik). Dokumentasi pengujian performa kedua dapat dilihat pada Gambar 5.7

Test Step Metrics

Create A SPK

Test step	Min (ms)	Max (ms)	Median (ms)	Last (ms)	Count	TPS	Err	Err %
SPK 1	307	993	567	993	18	0	0	0
Test Case Level	307	993	553	993	22	1	0	0

Response time details

Test Step	Avg (ms)	75%	90%	95%	98%
SPK 1	580	700.00	804.90	993.00	993.00
Test Case Level	570	684.00	784.00	993.00	993.00

Gambar 5.7 Dokumentasi pengujian performa 2

Gambar 5.7 memberikan informasi bahwa kecepatan respon tercepat pada angka 307 milidetik dan kecepatan respon terlama pada angka 993 milidetik serta rata-rata yaitu 570 milidetik. Dengan pengujian selama 30 detik dengan sejumlah 22 permintaan layanan web, layanan web memiliki sebaran kecepatan dibawah batas atas toleransi kecepatan yaitu 1000 milidetik. Dokumentasi lengkap pengujian performa dapat dilihat di Lampiran D.

5.2.4. Pengujian Keamanan

Pengujian keamanan dilakukan dengan melakukan proses *scanning* pada seluruh layanan web yang dimiliki oleh aplikasi *backend*. Kerentanan yang dideteksi pada pengujian keamanan ini adalah *SQL Injection* dan *Weak Authentication* yang disediakan oleh alat ReadyAPI. Pengujian dilakukan dengan menjalankan *template* pengujian yang telah disediakan oleh ReadyAPI. Dokumentasi pengujian keamanan dapat dilihat pada Gambar 5.8

SMARTBEAR ReadyAPI

SecurityTest Results for Security Test 1

SecurityTest Metrics

Overview

Project	Giyomi VMS Backend
TestSuite	Security Test Suite 1
TestCase	Giyomi Vendor Management Service TestCase
Security Test Name	Security Test 1
Description	

Base Metrics

Number of TestSteps	105
Number of SecurityScans	210

Gambar 5.8 Dokumentasi pengujian keamanan

Gambar 5.8 memperlihatkan bahwa dengan total 5033 *scanning* yang dilakukan, menghasilkan kesimpulan seluruh layanan web yang dimiliki oleh aplikasi *backend* telah lolos uji kerentanan dengan alat ReadyAPI. Dokumentasi lengkap pengujian keamanan dapat dilihat di Lampiran D.

BAB VI
KESIMPULAN DAN SARAN

Pada bagian ini akan menjelaskan mengenai kesimpulan dari pengerjaan tugas akhir ini dan saran untuk penelitian selanjutnya yang relevan.

6.1. Kesimpulan

Berikut merupakan kesimpulan yang dapat diambil dari tugas akhir yang telah dilakukan.

1. Informasi yang didapatkan dari tahap analisis prototipe sistem manajemen vendor Giyomi.id adalah sebagai berikut:
   a. Terdapat 14 kebutuhan fungsional yang harus dipenuhi oleh aplikasi *backend* sistem manajemen vendor giyomi.
   b. Terdapat 28 layanan web yang harus dimiliki oleh aplikasi *backend* untuk memenuhi seluruh kebutuhan fungsional.
2. Aplikasi *backend* telah terdokumentasi menggunakan platform Swagger.
3. Pengujian-pengujian yang dilakukan terhadap beberapa aspek dari aplikasi *backend* berstatus lolos uji, pengujian-pengujian yang dilakukan sebagai berikut:
   a. Pengujian Unit
      Terdapat 10 Unit yang diuji dan seluruuhnya berstatus lolos uji.
   b. Pengujian Fungsional
      Pengujian fungsionalitas fitur-fitur utama yaitu manajemen Surat Perintah Kerja (SPK), manajemen vendor dan manajemen akun penggua berstatus lolos uji.

c. Pengujian Performa
   Pengujian performa kecepatan layanan web dengan kompleksitas tinggi berstatus lolos uji dengan masksimal kecepatan respon 993 milidetik (0.993 detik). Pengujian performa kecepatan layanan web frekuansi tinggi berstatus lolos uji dengan maksimal kecepatan respon 72 milidetik (0.072 detik).
d. Pengujian Keamanan
   Pengujian keamanan seluruh layanan web berstatus lolos uji layanan web terbebas dari kerentaan *SQL Injection* dan *Weak Authentication.*

4. Aplikasi *backend* sistem manajemen vendor Giyomi.id siap digunakan oleh aplikasi *frontend* sistem manajemen vendor Giyomi.id agar menjadi aplikasi sistem manajemen vendor yang dapat dimanfaatkan oleh Giyomi.id.
5. Aplikasi *backend* telah memenuhi kebutuhan fungsional berdasarkan prototipe sistem manajemen vendor Giyomi.id yang dirancang untuk mengurangi risiko keterlambatan proses produksi dan memunculkan kerugian finansial dikarenakan pemilihan vendor yang memiliki performa tidak baik. Oleh karena itu dengan digunakannya aplikasi *frontend* sistem manajemen vendor Giyomi.id yang didukung oleh aplikasi *backend* sistem manajemen vendor Giyomi.id, Giyomi.id akan dapat mengurangi risiko kerugian finansial.
6. Penelitian mengenai implementasi aplikasi sistem manajemen vendor tersebut juga diharapkan dapat mengukur peningkatan kinerja pengelolaan vendor Giyomi.id dengan adanya aplikasi sistem manajemen vendor Giyomi.id.

6.2. Saran

Bagian ini berisi saran untuk penelitian lebih lanjut yang dapat dilakukan demi kebermanfaatan aplikasi sistem manajemen vendor Giyomi.id. Dengan selesainya pengerjaan tugas akhir tentang perancangan dan pengembangan aplikasi *backend* sistem manajemen vendor Giyomi.id serta penelitian lanjutan berupa pengembangan aplikasi *frontend* sistem manajemen vendor Giyomi.id, aplikasi sistem manajemen vendor Giyomi.id telah dapat digunakan oleh Giyomi.id. Untuk penelitian selanjutnya dapat diteliti bagaimana pemanfaatan dari aplikasi tersebut dan mengukur apakah benar-benar dapat meningkatkan proses manajemen vendor. Demi keberhasilan implementasi Giyomi.id juga perlu mempersiapkan infrastruktur yang dibutuhkan, yaitu peladen dan domain.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] A. Kinasih, “Analisis dan Usulan Rancang Ulang Proses Bisnis Produksi pada Usaha Garmen Make to Stock untuk Meningkatkan Ketepatan Waktu Produksi (Studi Kasus: UMKM Giyomi.id),” 2019.

[2] E. M. Bening, “Perancangan Prototipe Sistem Manajemen Vendor dengan Metode Iterative Prototyping dan Moderated Usability Test. (Studi Kasus : Umkm Giyomi.Id),” Tugas Akhir Departemen Sistem Informasi Fakultas Teknologi Informasi dan Komunikasi ITS, Surabaya, 2019.

[3] G. Rodr’iguez, J. A. D’iaz-Pace, and Á. Soria, “A Case-Based Reasoning Approach to Reuse Quality-Driven Designs in Service-Oriented Architectures,” *Inf. Syst.*, vol. 77, pp. 167–189, Sep. 2018, doi: 10.1016/j.is.2018.06.003.

[4] D. Hoffman and P. Strooper, “API Documentation With Executable Examples,” *J. Syst. Softw.*, vol. 66, no. 2, pp. 143–156, 2003, doi: 10.1016/S0164-1212(02)00055-9.

[5] “Create Great API Documentation | Swagger.” https://swagger.io/solutions/api-documentation/ (accessed Nov. 06, 2019).

[6] T. Yan Ting Wong, *Software Development Life Cycle*, no. 1. 2016.

[7] O. Suryadi, “Rancang Bangun Web Service pada Sistem Informasi Akademik Berbasis Representational State Transfer Menggunakan Service-Repository Pattern,” 2019.

[8] H. Suryotrisongko, D. P. Jayanto, and A. Tjahyanto, “Design and Development of Backend Application for Public Complaint Systems Using Microservice Spring Boot,” in *Procedia Computer Science*, 2017, doi: 10.1016/j.procs.2017.12.212.

[9] A. Nunome, H. Hirata, M. Fukuzawa, and K. Shibayama, “Development of an E-learning back-end system for code assessment in elementary programming practice,” *Proc. ACM SIGUCCS User Serv. Conf.*, pp. 181–186, 2010, doi: 10.1145/1878335.1878381.

[10] K. L. Madhuri and T. R. G. Nair, “Authentication and Authorization in Server Systems for Bio-Informatics,” *10014136*, 2010, [Online]. Available: http://arxiv.org/abs/1001.4136.

[11] H. Chen and P. Mohapatra, “Using service brokers for accessing backend servers for web applications,” *J. Netw. Comput. Appl.*, vol. 28, no. 1, pp. 57–74, Jan. 2005, doi: 10.1016/j.jnca.2004.02.001.

[12] R. Roslidar and R. Dawood, “Rancang Bangun Layanan Web (Web Service) Untuk Aplikasi Rekam Medis Praktik Pribadi Dokter,” *J. Online Tek. Elektro*, no. October, 2017.

[13] S. Balaji and M. S. Murugaiyan, “Waterfall vs. V-Model vs. Agile: A comparative study on SDLC,” *Int. J. Inf. Technol. Bus. Manag.*, vol. 2, no. 1, pp. 26–30, 2012.

[14] S. Feitosa, R. Ribeiro, and A. Du Bois, “A type-directed algorithm to generate random well-typed Java 8 programs,” *Sci. Comput. Program.*, vol. 196, p. 102494, 2020, doi: 10.1016/j.scico.2020.102494.

[15] A. Hunt and D. Thomas, *Pragmatic unit testing in Java with JUnit*. The Pragmatic Bookshelf, 2003.

[16] “The All-In-One Automated API Testing Platform | ReadyAPI.” https://smartbear.com/product/ready-api/overview/ (accessed Jun. 06, 2020).

[17] “Testing the API Backend of your Mobile Solution.” https://smartbear.com/blog/test-and-monitor/testing-the-api-backend-of-your-mobile-solution/ (accessed Jun. 06, 2020).

[18] "API Security Testing and Scanning | ReadyAPI." https://smartbear.com/product/ready-api/secure/overview/ (accessed Jun. 06, 2020).

[19] "IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains." https://www.jetbrains.com/idea/ (accessed Jun. 06, 2020).

[20] J. L. Harrington, *Relational Database Design and Implementation*. 2016.

[21] P. M. Fischer, M. Deshmukh, V. Maiwald, D. Quantius, A. M. Gomez, and A. Gerndt, "Conceptual Data Model: A Foundation for Successful Concurrent Engineering," *Concurr. Eng. Res. Appl.*, vol. 26, no. 1, pp. 55–76, 2018, doi: 10.1177/1063293X17734592.

[22] "DataGrip: The Cross-Platform IDE for Databases & SQL by JetBrains." https://www.jetbrains.com/datagrip/ (accessed Nov. 05, 2019).

[23] J. G. J. W. T. Clark, *Logical Database Design Principles*, vol. 66. 2012.

[24] P. Giessler, M. Gebhart, D. Sarancin, R. Steinegger, and S. Abeck, "Best Practices for the Design of RESTful Web Services," *Proc. Tenth Int. Conf. Softw. Eng. Adv. (ICSEA 2015)*, no. November 2015, pp. 392–398, 2016.

[25] S. Tumin and S. Encheva, *A Closer Look at Authentication and Authorization Mechanisms for Web-based Applications*. 2012.

[26] "Getting Started · Accessing Data with JPA." https://spring.io/guides/gs/accessing-data-jpa/ (accessed Nov. 05, 2019).

[27] "Getting Started · Building a RESTful Web Service." https://spring.io/guides/gs/rest-service/ (accessed Nov. 05, 2019).

[28] “Spring Security OAuth.” https://projects.spring.io/spring-security-oauth/docs/oauth2.html (accessed Nov. 06, 2019).

[29] “Advanced Java and Spring Microservices | Advanced Java and Spring Microservices - Experfy.” https://www.experfy.com/training/courses/advanced-java-and-spring-microservices (accessed Jun. 03, 2020).

BIODATA PENULIS

Penulis lahir di Pasuruan pada tanggal 21 Mei 1998, dengan nama Etika Pahang Krisdyan. Penulis merupakan anak terakhir dari orang tua bernama Budi Asmoro dan Endang Setiana. Penulis menempuh Pendidikan dasar di SDN 1 Wonokitri pada tahun 2004 hingga 2010. Kemudian penulis melanjutkan pendidikan sekolah menengah pertama di SMPN 7 Pasuruan dan lulus pada tahun 2013. Pada tahun yang sama, penulis melanjutkan pendidikan formal sekolah menengah atas di SMAN 10 Malang dan lulus pada tahun 2016. Selanjutnya, pada tahun 2016 penulis melanjutkan pendidikan tinggi di Departemen Sistem Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember. Penulis masuk pendidikan tinggi melalui jalur Seleksi Nasional Masuk Perguruan Tinggi Negeri.

Di Departemen Sistem Informasi ITS ini, penulis mengambil bidang minat Laboratorium Sistem Enterprise. Selama menjalani masa perkuliahan di ITS, penulis aktif dalam berbagai kegiatan seperti kepanitiaan dan organisasi. Penulis juga aktif mengikuti lomba-lomba yang ada serta mengikuti beberapa pelatihan. Selain itu, penulis juga pernah menjalani kerja praktik dan mencari pengalaman kerja dengan mengerjakan beberapa *project*.

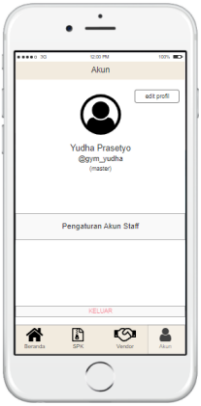
Informasi lebih detail terkait penulis, dapat menghubungi melalui email etika.pahang.k.1998@gmail.com

Halaman ini sengaja dikosongkan

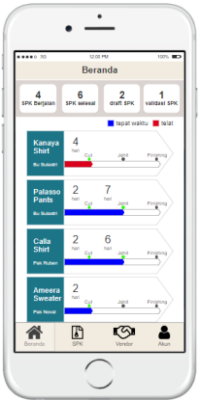
LAMPIRAN A
Keterangan Giyomi.id

Giyomi.id merupakan UMKM yang memproduksi pakaian wanita untuk dijual kepada pelanggan melalui daring maupun luring. Giyomi.id menggunakan *marketplace* dan media sosial sebagai wahana menjual secara daring sedangkan Giyomi.id menjual produknya secara luring di pasar dan bazar. Giyomi.id melakukan proses produksi mulai dari perancangan ide desain produk-produk baru, membuat sampel produk, menugaskan vendor untuk memproduksi produk, menerima produk jadi dari vendor sampai pemasaran produk [2].

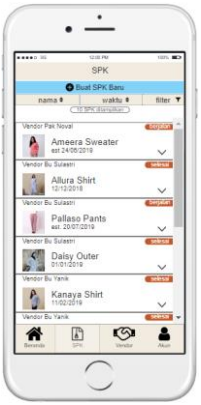
LAMPIRAN B
Sampel Prototipe Sistem Manajemen Vendor Giyomi.id oleh Bening



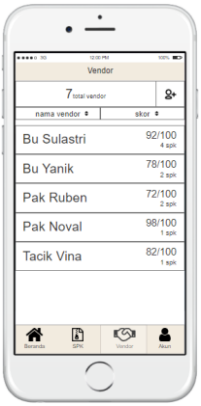
(a) Tampilan Halaman Profil Pengguna



(b) Tampilan Halaman Beranda



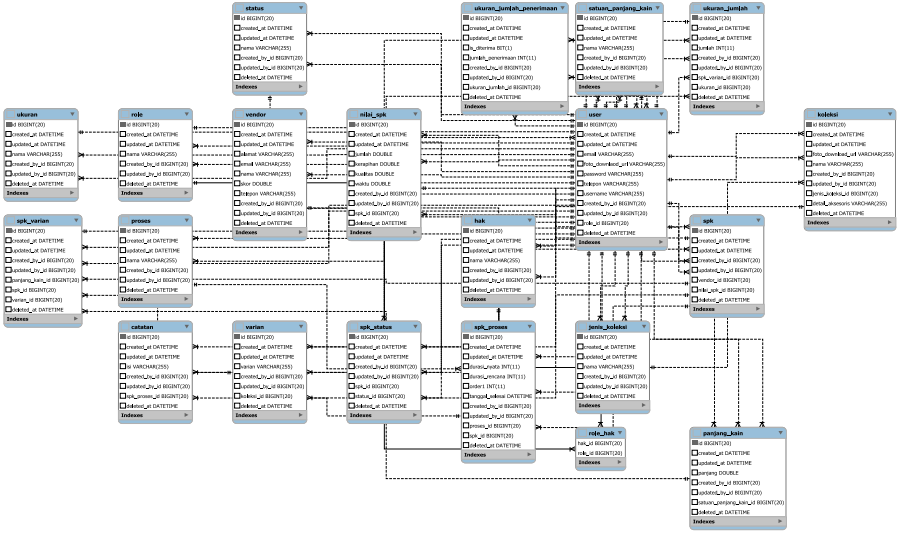
(c) Tampilan Halaman SPK



(d) Tampilan Halaman Vendor

Gambar B.1 (a)(b)(c)(d) Sampel Desain Prototipe Sistem Manajemen Vendor

LAMPIRAN C
Diagram Model Data Logis



Gambar C.1 Rancangan Model Data Logis

LAMPIRAN D
DOKUMENTASI PENGUJIAN

D.1. Fungsional Manajemen Vendor

SMARTBEAR ReadyAPI

TestSuite Results Report for Vendor Management

TestSuite Metrics

Overview

Project	Giyomi VMS Backend
Test suite name	Vendor Management
Description	
Environment	Default environment

Test Case Summary

- Create A Vendor
- Update A Vendor
- Get Vendors
- Delete A Vendor

Base Metrics

Number of test cases	4
Number of test steps	4
Number of assertions	15
Number of load tests	0

TestSuite Properties

Name	Value

created with ReadyAPI on Sun Jun 07 13:02:03 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for Vendor Management

TestCases

TestCase Name	TestCase Description
Create A Vendor	
Update A Vendor	
Get Vendors	
Delete A Vendor	

created with ReadyAPI on Sun Jun 07 13:02:03 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for Vendor Management

Result Metrics

Result Metrics	
Start Time	Sun Jun 07 12:53:18 WIB 2020
End Time	Sun Jun 07 12:53:19 WIB 2020
Time Taken	891 ms
TestCase Count	4
Failed TestCase Count	0
TestStep Count	4
Failed TestStep Count	0
Assertion Count	15
Failed Assertion Count	0

TestCase Results

TestCase	Status	Start Time	Time Taken	Reason
Create A Vendor	PASS	12:53:18	179 ms	
Update A Vendor	PASS	12:53:18	45 ms	
Get Vendors	PASS	12:53:18	30 ms	
Delete A Vendor	PASS	12:53:18	28 ms	

created with ReadyAPI on Sun Jun 07 13:02:03 WIB 2020

SMARTBEAR ReadyAPI

TestSuite Results Report for Vendor Management

Create A Vendor Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:18	179 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:18.202	Step 1 [Vendor Pak Nopal] PASS: took 179 ms

Test Step Parameters

Test steps

Vendor Pak Nopal

Endpoint	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:04 WIB 2020

SMARTBEAR ReadyAPI

TestSuite Results Report for Vendor Management

Update A Vendor Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:18	45 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:18.725	Step 1 [Update A Vendor] PASS: took 45 ms

Test Step Parameters

Test steps

Update A Vendor

Endpoint	http://localhost:8080

SMARTBEAR ReadyAPI

TestSuite Results Report for Vendor Management

Get Vendors Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:18	30 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:18.812	Step 1 [Get All Vendors] PASS: took 30 ms

Test Step Parameters

Test steps

Get All Vendors

Endpoint	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:04 WIB 2020

SMARTBEAR ReadyAPI

TestSuite Results Report for Vendor Management

Delete A Vendor Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:18	28 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:18.882	Step 1 [Delete A Vendor] PASS: took 28 ms

Test Step Parameters

Test steps

Delete A Vendor

Endpoint	http://localhost:8080

D.2. Fungsional Manajemen Pengguna

SMARTBEAR ReadyAPI

TestSuite Results Report for User Management

TestSuite Metrics

Overview

Project	Giyomi VMS Backend
Test suite name	User Management
Description	
Environment	Default environment

Test Case Summary

- Create A User
- Update A User
- Get Users Role=Staff
- Delete A User

Base Metrics

Number of test cases	4
Number of test steps	4
Number of assertions	8
Number of load tests	0

TestSuite Properties

Name	Value

created with ReadyAPI on Sun Jun 07 13:01:26 WIB 2020

SMARTBEAR ReadyAPI

TestSuite Results Report for User Management

TestCases

TestCase Name	TestCase Description
Create A User	
Update A User	
Get Users Role=Staff	
Delete A User	

created with ReadyAPI on Sun Jun 07 13:01:26 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for User Management

Result Metrics

Result Metrics

Start Time	Sun Jun 07 12:47:10 WIB 2020
End Time	Sun Jun 07 12:47:11 WIB 2020
Time Taken	901 ms
TestCase Count	4
Failed TestCase Count	0
TestStep Count	4
Failed TestStep Count	0
Assertion Count	8
Failed Assertion Count	0

TestCase Results

TestCase	Status	Start Time	Time Taken	Reason
Create A User	PASS	12:47:10	287 ms	
Update A User	PASS	12:47:11	39 ms	
Get Users Role=Staff	PASS	12:47:11	38 ms	
Delete A User	PASS	12:47:11	57 ms	

created with ReadyAPI on Sun Jun 07 13:01:26 WIB 2020

SMARTBEAR ReadyAPI

TestSuite Results Report for User Management

Create A User Summary

Status	Start Time	Time Taken	Reason
PASS	12:47:10	287 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:47:10.699	Step 1 [Staff 1] PASS: took 287 ms

Test Step Parameters

Test steps

Staff 1

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:01:26 WIB 2020

SMARTBEAR ReadyAPI

TestSuite Results Report for User Management

Update A User Summary

Status	Start Time	Time Taken	Reason
PASS	12:47:11	39 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:47:11.301	Step 1 [Staff 1] PASS: took 39 ms

Test Step Parameters

Test steps

Staff 1

Endpoint:	http://localhost:8080

SMARTBEAR ReadyAPI.

TestSuite Results Report for User Management

Get Users Role=Staff Summary

Status	Start Time	Time Taken	Reason
PASS	12:47:11	38 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:47:11.380	Step 1 [Get All Users] PASS: took 38 ms

Test Step Parameters

Test steps

Get All Users

Endpoint:	http://localhost:8080

SMARTBEAR ReadyAPI

TestSuite Results Report for User Management

Delete A User Summary

Status	Start Time	Time Taken	Reason
PASS	12:47:11	57 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:47:11.460	Step 1 [Delete A User] PASS: took 57 ms

Test Step Parameters

Test steps

Delete A User

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:01:26 WIB 2020

D.3. Fungsional Manajemen SPK

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

TestSuite Metrics

Overview

Project	Giyomi VMS Backend
Test suite name	SPK Management
Description	
Environment	Default environment

Test Case Summary

- Create A Vendor
- Create A Jenis Koleksi
- Create A Koleksi
- Create Ukurans
- Create Satuan Panjang
- Create Processes
- Create Statuses
- Create Varians
- Create A SPK
- Change SPK Status to REQ_VAL
- Change SPK Status to ON_GOING
- Submit Process Cutting Done
- Submit Process Jahit Done
- Submit Process Finishing Done
- Submit Penerimaan
- Get SPK Penilaian Kuantitatif
- Submit SPK Penilaian Kualitatif
- Get Vendor Performance

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

TestSuite Results Report for SPK Management

TestSuite Metrics

Base Metrics	
Number of test cases	18
Number of test steps	44
Number of assertions	82
Number of load tests	0

TestSuite Properties

Name	Value

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

TestCases

TestCase Name	TestCase Description
Create A Vendor	
Create A Jenis Koleksi	
Create A Koleksi	
Create Ukurans	
Create Satuan Panjang	
Create Processes	
Create Statuses	
Create Varians	
Create A SPK	
Change SPK Status to REQ_VAL	
Change SPK Status to ON_GOING	
Submit Process Cutting Done	
Submit Process Jahit Done	
Submit Process Finishing Done	
Submit Penerimaan	
Get SPK Penilaian Kuantitatif	
Submit SPK Penilaian Kualitatif	
Get Vendor Performance	

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

Result Metrics

Result Metrics	
Start Time	Sun Jun 07 12:53:21 WIB 2020
End Time	Sun Jun 07 12:53:26 WIB 2020
Time Taken	4486 ms
TestCase Count	18
Failed TestCase Count	0
TestStep Count	45
Failed TestStep Count	0
Assertion Count	83
Failed Assertion Count	0

TestCase Results

TestCase	Status	Start Time	Time Taken	Reason
Create A Vendor	PASS	12:53:21	24 ms	
Create A Jenis Koleksi	PASS	12:53:21	33 ms	
Create A Koleksi	PASS	12:53:21	33 ms	
Create Ukurans	PASS	12:53:21	114 ms	
Create Satuan Panjang	PASS	12:53:21	39 ms	
Create Processes	PASS	12:53:22	66 ms	
Create Statuses	PASS	12:53:22	109 ms	
Create Varians	PASS	12:53:22	76 ms	
Create A SPK	PASS	12:53:22	238 ms	
Change SPK Status to REQ_VAL	PASS	12:53:22	86 ms	
Change SPK Status to ON_GOING	PASS	12:53:22	118 ms	
Submit Process Cutting Done	PASS	12:53:22	83 ms	
Submit Process Jahit Done	PASS	12:53:23	76 ms	
Submit Process Finishing	PASS	12:53:23	67 ms	

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

TestCase Results

TestCase	Status	Start Time	Time Taken	Reason
Done				
Submit Penerimaan	PASS	12:53:23	1619 ms	
Get SPK Penilaian Kuantitatif	PASS	12:53:25	178 ms	
Submit SPK Penilaian Kualitatif	PASS	12:53:25	184 ms	
Get Vendor Performance	PASS	12:53:25	78 ms	

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI

TestSuite Results Report for SPK Management

Create A Vendor Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:21	24 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:21.562	Step 1 [Vendor Pak Nopal] PASS: took 24 ms

Test Step Parameters

Test steps

Vendor Pak Nopal

Endpoint:	http://localhost:8080

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

Create A Jenis Koleksi Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:21	33 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:21.629	Step 1 [Jenis Koleksi Atasan] PASS: took 33 ms

Test Step Parameters

Test steps

Jenis Koleksi Atasan

Endpoint:	http://localhost:8080

SMARTBEAR ReadyAPI

TestSuite Results Report for SPK Management

Create A Koleksi Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:21	33 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:21.699	Step 1 [Koleksi Ameraa Sweater] PASS: took 33 ms

Test Step Parameters

Test steps

Koleksi Ameraa Sweater

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI

TestSuite Results Report for SPK Management

Create Ukurans Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:21	114 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:21.776	Step 1 [Ukuran S] PASS: took 31 ms
PASS	12:53:21.813	Step 2 [Ukuran M] PASS: took 22 ms
PASS	12:53:21.840	Step 3 [Ukuran L] PASS: took 20 ms
PASS	12:53:21.864	Step 4 [Ukuran XL] PASS: took 22 ms
PASS	12:53:21.891	Step 5 [Ukuran XXL] PASS: took 19 ms

Test Step Parameters

Test steps

Ukuran S

Endpoint:	http://localhost:8080

Ukuran M

Endpoint:	http://localhost:8080

Ukuran L

Endpoint:	http://localhost:8080

Ukuran XL

Endpoint:	http://localhost:8080

Ukuran XXL

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI

TestSuite Results Report for SPK Management

Create Satuan Panjang Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:21	39 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:21.944	Step 1 [Satuan Panjang Meter] PASS: took 39 ms

Test Step Parameters

Test steps

Satuan Panjang Meter

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI

TestSuite Results Report for SPK Management

Create Processes Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:22	66 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:22.028	Step 1 [Process Cutting] PASS: took 27 ms
PASS	12:53:22.061	Step 2 [Process Jahit] PASS: took 19 ms
PASS	12:53:22.082	Step 3 [Process Finishing] PASS: took 20 ms

Test Step Parameters

Test steps

Process Cutting

Endpoint:	http://localhost:8080

Process Jahit

Endpoint:	http://localhost:8080

Process Finishing

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

Create Statuses Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:22	109 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:22.138	Step 1 [Draft] PASS: took 28 ms
PASS	12:53:22.172	Step 2 [Request Validation] PASS: took 19 ms
PASS	12:53:22.195	Step 3 [On Going] PASS: took 20 ms
PASS	12:53:22.230	Step 4 [Completed] PASS: took 17 ms
PASS	12:53:22.251	Step 5 [Pending] PASS: took 25 ms

Test Step Parameters

Test steps

Draft

Endpoint:	http://localhost:8080

Request Validation

Endpoint:	http://localhost:8080

On Going

Endpoint:	http://localhost:8080

Completed

Endpoint:	http://localhost:8080

Pending

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

Create Varians Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:22	76 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:22.314	Step 1 [Amera Maroon] PASS: took 33 ms
PASS	12:53:22.355	Step 2 [Amera Biru] PASS: took 22 ms
PASS	12:53:22.381	Step 3 [Amera Hitam] PASS: took 21 ms

Test Step Parameters

Test steps

Amera Maroon

Endpoint:	http://localhost:8080

Amera Biru

Endpoint:	http://localhost:8080

Amera Hitam

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

Create A SPK Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:22	238 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:22.437	Step 1 [SPK 1] PASS: took 238 ms

Test Step Parameters

Test steps

SPK 1

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

Change SPK Status to REQ_VAL Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:22	86 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:22.711	Step 1 [REST Request] PASS: took 86 ms

Test Step Parameters

Test steps

REST Request

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

Change SPK Status to REQ_VAL Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:22	86 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:22.711	Step 1 [REST Request] PASS: took 86 ms

Test Step Parameters

Test steps

REST Request

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

Change SPK Status to ON_GOING Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:22	118 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:22.832	Step 1 [REST Request] PASS: took 118 ms

Test Step Parameters

Test steps

REST Request

Endpoint:	http://localhost:8080

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

Submit Process Cutting Done Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:22	83 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:22.982	Step 1 [REST Request] PASS: took 83 ms

Test Step Parameters

Test steps

REST Request

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

Submit Process Jahit Done Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:23	76 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:23.100	Step 1 [REST Request] PASS: took 76 ms

Test Step Parameters

Test steps

REST Request

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

Submit Process Finishing Done Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:23	67 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:23.211	Step 1 [REST Request] PASS: took 67 ms

Test Step Parameters

Test steps

REST Request

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

Submit Penerimaan Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:23	1619 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:23.337	Step 1 [Penerimaan #1] PASS: took 76 ms
PASS	12:53:23.420	Step 2 [Penerimaan #2] PASS: took 78 ms
PASS	12:53:23.502	Step 3 [Penerimaan #3] PASS: took 78 ms
PASS	12:53:23.585	Step 4 [Penerimaan #4] PASS: took 69 ms
PASS	12:53:23.957	Step 5 [Penerimaan #5] PASS: took 94 ms
PASS	12:53:24.057	Step 6 [Penerimaan #6] PASS: took 95 ms
PASS	12:53:24.158	Step 7 [Penerimaan #7] PASS: took 82 ms
PASS	12:53:24.252	Step 8 [Penerimaan #8] PASS: took 75 ms
PASS	12:53:24.333	Step 9 [Penerimaan #9] PASS: took 81 ms
PASS	12:53:24.419	Step 10 [Penerimaan #10] PASS: took 69 ms
PASS	12:53:24.493	Step 11 [Penerimaan #11] PASS: took 92 ms
PASS	12:53:24.600	Step 12 [Penerimaan #12] PASS: took 101 ms
PASS	12:53:24.707	Step 13 [Penerimaan #13] PASS: took 89 ms
PASS	12:53:24.801	Step 14 [Penerimaan #14] PASS: took 85 ms
PASS	12:53:24.892	Step 15 [Penerimaan #15] PASS: took 300 ms
PASS	12:53:25.197	Step 16 [ChangeStatusToCompleted] PASS: took 155 ms

Test Step Parameters

Test steps

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

Test Step Parameters

Test steps

Penerimaan #1	
Endpoint:	http://localhost:8080
Penerimaan #2	
Endpoint:	http://localhost:8080
Penerimaan #3	
Endpoint:	http://localhost:8080
Penerimaan #4	
Endpoint:	http://localhost:8080
Penerimaan #5	
Endpoint:	http://localhost:8080
Penerimaan #6	
Endpoint:	http://localhost:8080
Penerimaan #7	
Endpoint:	http://localhost:8080
Penerimaan #8	
Endpoint:	http://localhost:8080
Penerimaan #9	
Endpoint:	http://localhost:8080
Penerimaan #10	
Endpoint:	http://localhost:8080
Penerimaan #11	
Endpoint:	http://localhost:8080
Penerimaan #12	
Endpoint:	http://localhost:8080
Penerimaan #13	
Endpoint:	http://localhost:8080
Penerimaan #14	
Endpoint:	http://localhost:8080
Penerimaan #15	
Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

TestSuite Results Report for SPK Management

Test Step Parameters

Test steps

ChangeStatusToCompleted

Endpoint:	http://localhost:8080

SMARTBEAR ReadyAPI

TestSuite Results Report for SPK Management

Get SPK Penilaian Kuantitatif Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:25	178 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:25.448	Step 1 [REST Request] PASS: took 178 ms

Test Step Parameters

Test steps

REST Request

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

SMARTBEAR ReadyAPI.

TestSuite Results Report for SPK Management

Submit SPK Penilaian Kualitatif Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:25	184 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:25.667	Step 1 [REST Request] PASS: took 184 ms

Test Step Parameters

Test steps

REST Request

Endpoint:	http://localhost:8080

SMARTBEAR ReadyAPI

TestSuite Results Report for SPK Management

Get Vendor Performance Summary

Status	Start Time	Time Taken	Reason
PASS	12:53:25	78 ms	

TestCase Properties

Name	Value

Test Step Results

Status	Timestamp	Message
PASS	12:53:25.898	Step 1 [REST Request] PASS: took 78 ms

Test Step Parameters

Test steps

REST Request

Endpoint:	http://localhost:8080

created with ReadyAPI on Sun Jun 07 13:02:21 WIB 2020

D.4. Performa Kecepatan Layanan Web API11

SMARTBEAR ReadyAPI.

Load Test Results for API Kompleks

Test properties

Project name	Giyomi VMS Backend
Project description	
Test name	API Kompleks
Test duration	0:30
Warm-up time	5 sec

Load allocation

Load type	VUs
Allocation type	Relative
Allocation parameters	Percentage distribution of 1 VU(s)

Load test overview

Scenarios in the test	1
Load agents used	0
Cloud agents used	0
Servers monitored	0
Test assertions (all levels)	2

Scenario details

Scenario

Load profile	Fixed
Base VUs	1
Wait time	1 second(s)
Target test case(s)	
SPK Management:Create A SPK	1 enabled test step(s)

created with ReadyAPI on Sun Jun 07 14:11:10 WIB 2020

SMARTBEAR ReadyAPI.

Test Execution Metrics

Test results

Test run result	PASS
Duration	00:00:35
Start time	14:10:29
End time	14:11:04
Maximum simultaneous virtual users	1

Information on target test cases

Test case runs, including loops	19
Failed test cases	0
Failure ratio	0%

Load test assertions summary

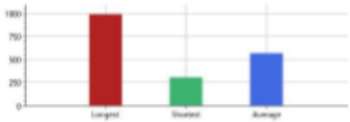
Assertion checks	53
Failed assertions	0
Failure ratio	0%

Executed test step(s) summary

Test step runs, including loops	19
Failed test steps	0
Failure ratio	0%

Response time

Longest	993 ms
Shortest	307 ms
Average	570 ms



A great difference between response time values may indicate server issues.

created with ReadyAPI on Sun Jun 07 14:11:10 WIB 2020

SMARTBEAR ReadyAPI

Scenarios Summary

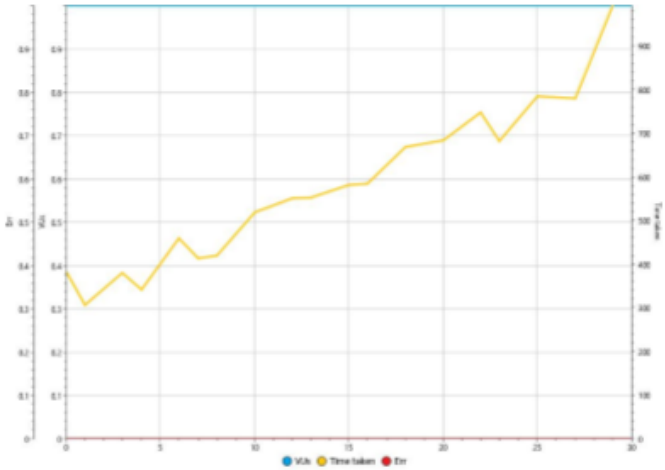
This section provides information about target test cases and assertions for each scenario in the load test.

Scenario name	Duration	Targets	Assertions checked	Assertions failed
Scenario	00:00:35	19	34	0

created with ReadyAPI on Sun Jun 07 14:11:10 WIB 2020

SMARTBEAR ReadyAPI.

Overview Chart



Test Step Metrics

Create A SPK

Test step	Min (ms)	Max (ms)	Median (ms)	Last (ms)	Count	TPS	Err	Err %
SPK 1	307	993	567	993	18	0	0	0
Test Case Level	307	993	553	993	22	1	0	0

Response time details

Test Step	Avg (ms)	75%	90%	95%	98%
SPK 1	580	700.00	804.90	993.00	993.00
Test Case Level	570	684.00	784.00	993.00	993.00

created with ReadyAPI on Sun Jun 07 14:11:10 WIB 2020

SMARTBEAR ReadyAPI

Top 5 Test Cases

Test case	Time taken (ms)	Start time	Size (bytes)
SPK Management:Create A SPK	307	14:10:36	1826
SPK Management:Create A SPK	342	14:10:38	1841
SPK Management:Create A SPK	380	14:10:37	1832
SPK Management:Create A SPK	383	14:10:34	1826
SPK Management:Create A SPK	414	14:10:41	1842

Bottom 5 Test Cases

Test case	Time taken (ms)	Start time	Size (bytes)
SPK Management:Create A SPK	993	14:11:03	1842
SPK Management:Create A SPK	784	14:10:59	1842
SPK Management:Create A SPK	780	14:11:01	1842
SPK Management:Create A SPK	748	14:10:55	1842
SPK Management:Create A SPK	684	14:10:54	1842

Top 5 Test Steps

Test case	Time taken (ms)	Start time	Size (bytes)
SPK Management:Create A SPK->SPK 1	307	14:10:36	1826
SPK Management:Create A SPK->SPK 1	342	14:10:39	1841
SPK Management:Create A SPK->SPK 1	380	14:10:37	1832
SPK Management:Create A SPK->SPK 1	414	14:10:42	1842
SPK Management:Create A SPK->SPK 1	420	14:10:43	1842

Bottom 5 Test Steps

Test case	Time taken (ms)	Start time	Size (bytes)
SPK Management:Create A SPK->SPK 1	993	14:11:04	1842
SPK Management:Create A SPK->SPK 1	784	14:11:00	1842
SPK Management:Create A SPK->SPK 1	780	14:11:02	1842
SPK Management:Create A SPK->SPK 1	748	14:10:56	1842
SPK Management:Create A SPK->SPK 1	684	14:10:54	1842

SMARTBEAR ReadyAPI

Load Test Results for Scenario Scenario

Scenario overview

Load profile	Fixed
Scenario targets	1
Simulated users	1

Information on target test cases

Test case runs, including loops	19
Failed test cases	0
Failure ratio	0%

Load test assertions summary

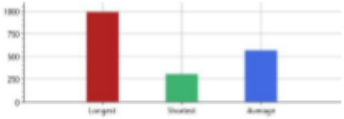
Assertion checks	34
Failed assertions	0
Failure ratio	0%

Executed test step(s) summary

Test step runs, including loops	19
Failed test steps	0
Failure ratio	0%

Response time

Longest	993 ms
Shortest	307 ms
Average	570 ms



A great difference between response time values may indicate server issues.

Targets Metrics

Target	Requests	Min time (ms)	Max time (ms)	Median time (ms)	Std dev (ms)	Err	Err %

created with ReadyAPI on Sun Jun 07 14:11:10 WIB 2020

SMARTBEAR ReadyAPI.

Targets Metrics

Target	Requests	Min time (ms)	Max time (ms)	Median time (ms)	Std dev (ms)	Err	Err %
SPK Management:Create A SPK	22	307	993	553	261	0	0

Response time details

Target	Avg (ms)	75%	90%	95%	98%
SPK Management:Create A SPK	570	684.00	784.00	993.00	993.00

Top 5 Test Case Executions

Test case	Time taken (ms)	Start time	Size (bytes)
SPK Management:Create A SPK	307	14:10:36	1826
SPK Management:Create A SPK	342	14:10:38	1841
SPK Management:Create A SPK	380	14:10:37	1832
SPK Management:Create A SPK	383	14:10:34	1826
SPK Management:Create A SPK	414	14:10:41	1842

Bottom 5 Test Case Executions

Test case	Time taken (ms)	Start time	Size (bytes)
SPK Management:Create A SPK	993	14:11:03	1842
SPK Management:Create A SPK	784	14:10:59	1842
SPK Management:Create A SPK	780	14:11:01	1842
SPK Management:Create A SPK	748	14:10:55	1842
SPK Management:Create A SPK	684	14:10:54	1842

Top 5 Test Step Executions

Test case	Time taken (ms)	Start time	Size (bytes)
SPK Management:Create A SPK->SPK 1	307	14:10:36	1826
SPK Management:Create A SPK->SPK 1	342	14:10:39	1841
SPK Management:Create A SPK->SPK 1	380	14:10:37	1832
SPK Management:Create A SPK->SPK 1	414	14:10:42	1842
SPK Management:Create A SPK->SPK 1	420	14:10:43	1842

created with ReadyAPI on Sun Jun 07 14:11:10 WIB 2020

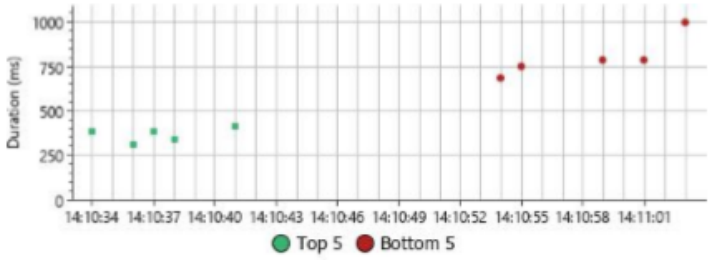
SMARTBEAR ReadyAPI.

Bottom 5 Test Step Executions

Test case	Time taken (ms)	Start time	Size (bytes)
SPK Management:Create A SPK->SPK 1	993	14:11:04	1842
SPK Management:Create A SPK->SPK 1	784	14:11:00	1842
SPK Management:Create A SPK->SPK 1	780	14:11:02	1842
SPK Management:Create A SPK->SPK 1	748	14:10:56	1842
SPK Management:Create A SPK->SPK 1	684	14:10:54	1842

Execution dispersion

This graph shows when the 5 fastest and 5 slowest test case executions happened. The clustered values show the time when the issues occurred during the test run. For example, a lot of values at the beginning of the test might mean that you need a longer warm-up time for the server. Ideally, the values should be distributed evenly throughout the test duration.



D.5. Performa Kecepatan Layanan Web API8

SMARTBEAR ReadyAPI.

Load Test Results for Load Test 2

Test properties

Project name	Giyomi VMS Backend
Project description	
Test name	Load Test 2
Test duration	0:30
Warm-up time	5 sec

Load allocation

Load type	VUs
Allocation type	Relative
Allocation parameters	Percentage distribution of 10 VU(s)

Load test overview

Scenarios in the test	1
Load agents used	0
Cloud agents used	0
Servers monitored	0
Test assertions (all levels)	1

Scenario details

New Scenario

Load profile	Fixed
Base VUs	10
Wait time	1 second(s)
Target test case(s)	
Single Request Test Cases:LoadUI Single	1 enabled test step(s)

created with ReadyAPI on Sun Jun 07 13:49:48 WIB 2020

SMARTBEAR ReadyAPI

Test Execution Metrics

Test results

Test run result	PASS
Duration	00:00:35
Start time	13:49:01
End time	13:49:36
Maximum simultaneous virtual users	10

Information on target test cases

Test case runs, including loops	290
Failed test cases	0
Failure ratio	0%

Load test assertions summary

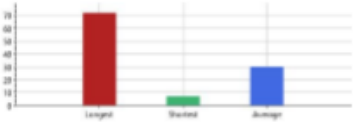
Assertion checks	30
Failed assertions	0
Failure ratio	0%

Executed test step(s) summary

Test step runs, including loops	290
Failed test steps	0
Failure ratio	0%

Response time

Longest	72 ms
Shortest	7 ms
Average	30 ms



A great difference between response time values may indicate server issues.

created with ReadyAPI on Sun Jun 07 13:49:48 WIB 2020

SMARTBEAR ReadyAPI

Scenarios Summary

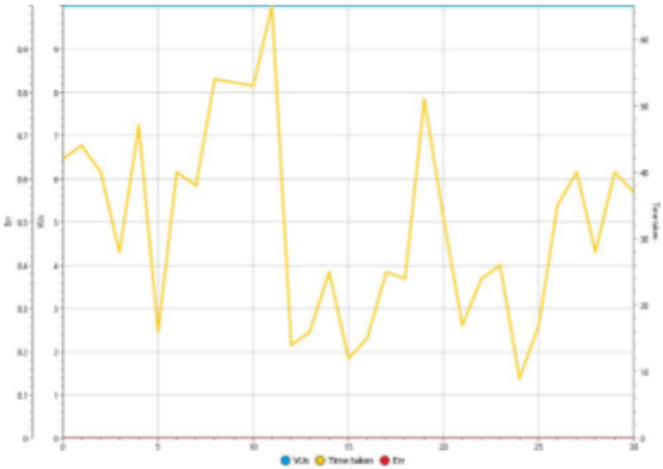
This section provides information about target test cases and assertions for each scenario in the load test.

Scenario name	Duration	Targets	Assertions checked	Assertions failed
New Scenario	00:00:35	290	0	0

created with ReadyAPI on Sun Jun 07 13:49:48 WIB 2020

SMARTBEAR ReadyAPI

Overview Chart



Test Step Metrics

LoadUI Single Request Test Case 2

Test step	Min (ms)	Max (ms)	Median (ms)	Last (ms)	Count	TPS	Err	Err %
Request 1	0	72	26	43	280	7	0	0
Test Case Level	7	72	28	43	330	9	0	0

Response time details

Test Step	Avg (ms)	75%	90%	95%	98%
Request 1	29	40.00	52.00	57.95	65.38
Test Case Level	30	41.00	53.00	58.45	65.18

created with ReadyAPI on Sun Jun 07 13:49:48 WIB 2020

SMARTBEAR ReadyAPI.

Top 5 Test Cases

Test case	Time taken (ms)	Start time	Size (bytes)
Single Request Test Cases:LoadUI Single Request Test Case 2	7	13:49:23	2
Single Request Test Cases:LoadUI Single Request Test Case 2	8	13:49:18	2
Single Request Test Cases:LoadUI Single Request Test Case 2	8	13:49:20	2
Single Request Test Cases:LoadUI Single Request Test Case 2	8	13:49:21	2
Single Request Test Cases:LoadUI Single Request Test Case 2	8	13:49:22	2

Bottom 5 Test Cases

Test case	Time taken (ms)	Start time	Size (bytes)
Single Request Test Cases:LoadUI Single Request Test Case 2	72	13:49:16	2
Single Request Test Cases:LoadUI Single Request Test Case 2	68	13:49:16	2
Single Request Test Cases:LoadUI Single Request Test Case 2	67	13:49:16	2
Single Request Test Cases:LoadUI Single Request Test Case 2	66	13:49:16	2
Single Request Test Cases:LoadUI Single Request Test Case 2	66	13:49:16	2

Top 5 Test Steps

Test case	Time taken (ms)	Start time	Size (bytes)
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	0	13:49:36	0
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	0	13:49:36	0
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	0	13:49:36	0
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	0	13:49:36	0
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	0	13:49:36	0

Bottom 5 Test Steps

Test case	Time taken (ms)	Start time	Size (bytes)
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	72	13:49:16	2
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	68	13:49:16	2
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	67	13:49:16	2

created with ReadyAPI on Sun Jun 07 13:49:48 WIB 2020

SMARTBEAR ReadyAPI

Bottom 5 Test Steps

Test case	Time taken (ms)	Start time	Size (bytes)
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	66	13:49:16	2
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	66	13:49:16	2

SMARTBEAR ReadyAPI.

Load Test Results for New Scenario Scenario

Scenario overview

Load profile	Fixed
Scenario targets	1
Simulated users	10

Information on target test cases

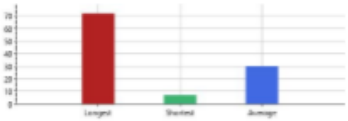
Test case runs, including loops	290
Failed test cases	0
Failure ratio	0%

Executed test step(s) summary

Test step runs, including loops	290
Failed test steps	0
Failure ratio	0%

Response time

Longest	72 ms
Shortest	7 ms
Average	30 ms



A great difference between response time values may indicate server issues.

Targets Metrics

Target	Requests	Min time (ms)	Max time (ms)	Median time (ms)	Std dev (ms)	Err	Err %
Single Request Test Cases:LoadUI Single Request Test Case 2	330	7	72	28	17	0	0

Response time details

Target	Avg (ms)	75%	90%	95%	98%
Single Request Test Cases:LoadUI Single Request Test Case 2	30	41.00	53.00	58.45	65.18

created with ReadyAPI on Sun Jun 07 13:49:49 WIB 2020

SMARTBEAR ReadyAPI.

Top 5 Test Case Executions

Test case	Time taken (ms)	Start time	Size (bytes)
Single Request Test Cases:LoadUI Single Request Test Case 2	7	13:49:23	2
Single Request Test Cases:LoadUI Single Request Test Case 2	8	13:49:18	2
Single Request Test Cases:LoadUI Single Request Test Case 2	8	13:49:20	2
Single Request Test Cases:LoadUI Single Request Test Case 2	8	13:49:21	2
Single Request Test Cases:LoadUI Single Request Test Case 2	8	13:49:22	2

Bottom 5 Test Case Executions

Test case	Time taken (ms)	Start time	Size (bytes)
Single Request Test Cases:LoadUI Single Request Test Case 2	72	13:49:16	2
Single Request Test Cases:LoadUI Single Request Test Case 2	68	13:49:16	2
Single Request Test Cases:LoadUI Single Request Test Case 2	67	13:49:16	2
Single Request Test Cases:LoadUI Single Request Test Case 2	66	13:49:16	2
Single Request Test Cases:LoadUI Single Request Test Case 2	66	13:49:16	2

Top 5 Test Step Executions

Test case	Time taken (ms)	Start time	Size (bytes)
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	0	13:49:36	0
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	0	13:49:36	0
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	0	13:49:36	0
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	0	13:49:36	0
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	0	13:49:36	0

Bottom 5 Test Step Executions

Test case	Time taken (ms)	Start time	Size (bytes)
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	72	13:49:16	2
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	68	13:49:16	2
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	67	13:49:16	2
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	66	13:49:16	2

created with ReadyAPI on Sun Jun 07 13:49:49 WIB 2020

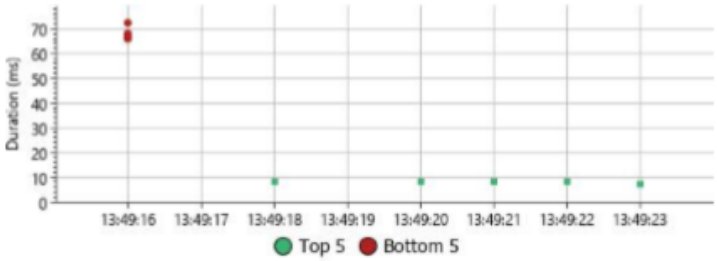
SMARTBEAR ReadyAPI.

Bottom 5 Test Step Executions

Test case	Time taken (ms)	Start time	Size (bytes)
Single Request Test Cases:LoadUI Single Request Test Case 2->Request 1	66	13:49:16	2

Execution dispersion

This graph shows when the 5 fastest and 5 slowest test case executions happened. The clustered values show the time when the issues occurred during the test run. For example, a lot of values at the beginning of the test might mean that you need a longer warm-up time for the server. Ideally, the values should be distributed evenly throughout the test duration.



D.6. Keamanan

SMARTBEAR ReadyAPI.

SecurityTest Results for Security Test 1

SecurityTest Metrics

Overview

Project	Giyomi VMS Backend
TestSuite	Security Test Suite 1
TestCase	Giyomi Vendor Management Service TestCase
Security Test Name	Security Test 1
Description	

Base Metrics

Number of TestSteps	105
Number of SecurityScans	210

SMARTBEAR ReadyAPI.

SecurityTest TestSteps

TestStep Name	Security Scan List
GET	SQL Injection Weak Authentication
POST	SQL Injection Weak Authentication
PUT	SQL Injection Weak Authentication
DELETE	SQL Injection Weak Authentication
PATCH	SQL Injection Weak Authentication
OPTIONS	SQL Injection Weak Authentication
POST 2	SQL Injection Weak Authentication
GET 2	SQL Injection Weak Authentication
POST 3	SQL Injection Weak Authentication
GET 3	SQL Injection Weak Authentication
GET 4	SQL Injection Weak Authentication
POST 4	SQL Injection Weak Authentication
GET 5	SQL Injection Weak Authentication
POST 5	SQL Injection Weak Authentication
GET 6	SQL Injection Weak Authentication
GET 7	SQL Injection Weak Authentication
POST 6	SQL Injection Weak Authentication
GET 8	SQL Injection Weak Authentication
PUT 2	SQL Injection Weak Authentication
DELETE 2	SQL Injection Weak Authentication
GET 9	SQL Injection Weak Authentication

SMARTBEAR ReadyAPI

SecurityTest TestSteps

TestStep Name	Security Scan List
POST 7	SQL Injection Weak Authentication
GET 10	SQL Injection Weak Authentication
PUT 3	SQL Injection Weak Authentication
DELETE 3	SQL Injection Weak Authentication
GET 11	SQL Injection Weak Authentication
POST 8	SQL Injection Weak Authentication
GET 12	SQL Injection Weak Authentication
PUT 4	SQL Injection Weak Authentication
DELETE 4	SQL Injection Weak Authentication
GET 13	SQL Injection Weak Authentication
POST 9	SQL Injection Weak Authentication
GET 14	SQL Injection Weak Authentication
PUT 5	SQL Injection Weak Authentication
DELETE 5	SQL Injection Weak Authentication
GET 15	SQL Injection Weak Authentication
POST 10	SQL Injection Weak Authentication
GET 16	SQL Injection Weak Authentication
PUT 6	SQL Injection Weak Authentication
DELETE 6	SQL Injection Weak Authentication
GET 17	SQL Injection Weak Authentication
POST 11	SQL Injection Weak Authentication

SMARTBEAR ReadyAPI.

SecurityTest TestSteps

TestStep Name	Security Scan List
GET 18	SQL Injection Weak Authentication
PUT 7	SQL Injection Weak Authentication
DELETE 7	SQL Injection Weak Authentication
GET 19	SQL Injection Weak Authentication
POST 12	SQL Injection Weak Authentication
GET 20	SQL Injection Weak Authentication
PUT 8	SQL Injection Weak Authentication
DELETE 8	SQL Injection Weak Authentication
GET 21	SQL Injection Weak Authentication
POST 13	SQL Injection Weak Authentication
GET 22	SQL Injection Weak Authentication
PUT 9	SQL Injection Weak Authentication
DELETE 9	SQL Injection Weak Authentication
GET 23	SQL Injection Weak Authentication
POST 14	SQL Injection Weak Authentication
GET 24	SQL Injection Weak Authentication
PUT 10	SQL Injection Weak Authentication
DELETE 10	SQL Injection Weak Authentication
GET 25	SQL Injection Weak Authentication
POST 15	SQL Injection Weak Authentication
GET 26	SQL Injection Weak Authentication

SMARTBEAR ReadyAPI.

SecurityTest TestSteps

TestStep Name	Security Scan List
PUT 11	SQL Injection Weak Authentication
DELETE 11	SQL Injection Weak Authentication
GET 27	SQL Injection Weak Authentication
POST 16	SQL Injection Weak Authentication
GET 28	SQL Injection Weak Authentication
PUT 12	SQL Injection Weak Authentication
DELETE 12	SQL Injection Weak Authentication
GET 29	SQL Injection Weak Authentication
POST 17	SQL Injection Weak Authentication
GET 30	SQL Injection Weak Authentication
PUT 13	SQL Injection Weak Authentication
DELETE 13	SQL Injection Weak Authentication
GET 31	SQL Injection Weak Authentication
POST 18	SQL Injection Weak Authentication
GET 32	SQL Injection Weak Authentication
PUT 14	SQL Injection Weak Authentication
DELETE 14	SQL Injection Weak Authentication
GET 33	SQL Injection Weak Authentication
POST 19	SQL Injection Weak Authentication
GET 34	SQL Injection Weak Authentication
PUT 15	SQL Injection Weak Authentication

SMARTBEAR ReadyAPI.

SecurityTest TestSteps

TestStep Name	Security Scan List
DELETE 15	SQL Injection Weak Authentication
GET 35	SQL Injection Weak Authentication
POST 20	SQL Injection Weak Authentication
GET 36	SQL Injection Weak Authentication
PUT 16	SQL Injection Weak Authentication
DELETE 16	SQL Injection Weak Authentication
GET 37	SQL Injection Weak Authentication
POST 21	SQL Injection Weak Authentication
GET 38	SQL Injection Weak Authentication
PUT 17	SQL Injection Weak Authentication
DELETE 17	SQL Injection Weak Authentication
GET 39	SQL Injection Weak Authentication
POST 22	SQL Injection Weak Authentication
GET 40	SQL Injection Weak Authentication
PUT 18	SQL Injection Weak Authentication
DELETE 18	SQL Injection Weak Authentication
GET 41	SQL Injection Weak Authentication
POST 23	SQL Injection Weak Authentication
GET 42	SQL Injection Weak Authentication
PUT 19	SQL Injection Weak Authentication
DELETE 19	SQL Injection Weak Authentication

SMARTBEAR ReadyAPI.

SecurityTest Properties

Name	Value

Result Metrics

Result Metrics	
Start Time	Sun Jun 07 13:30:47 WIB 2020
End Time	Sun Jun 07 13:43:56 WIB 2020
Time Taken	789300 ms
TestStep Count	105
Scan Count	210
ScanRequests Count	5033
Alerts Count	0
Total Alert Rate	0
Request Alert Rate	0

SecurityTestStep Overview

GET	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts

SMARTBEAR ReadyAPI

SecurityTestStep Overview

PATCH	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
OPTIONS	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 2	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 2	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 3	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 3	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 4	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 4	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 5	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts

SMARTBEAR ReadyAPI

SecurityTestStep Overview

POST 5	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 6	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 7	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 6	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 8	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 2	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 2	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 9	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 7	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts

SMARTBEAR ReadyAPI.

SecurityTestStep Overview

GET 10	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 3	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 3	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 11	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 8	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 12	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 4	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 4	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 13	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts

SMARTBEAR ReadyAPI

SecurityTestStep Overview

POST 9	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 14	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 5	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 5	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 15	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 10	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 16	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 6	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 6	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts

SMARTBEAR ReadyAPI

SecurityTestStep Overview

GET 17	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 11	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 18	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 7	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 7	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 19	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 12	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 20	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 8	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts

SMARTBEAR ReadyAPI

SecurityTestStep Overview

DELETE 8	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 21	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 13	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 22	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 9	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 9	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 23	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 14	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 24	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts

SMARTBEAR ReadyAPI

SecurityTestStep Overview

PUT 10	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 10	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 25	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 15	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 26	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 11	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 11	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 27	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 16	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts

SMARTBEAR ReadyAPI

SecurityTestStep Overview

GET 28	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 12	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 12	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 29	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 17	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 30	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 13	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 13	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 31	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts

SMARTBEAR ReadyAPI

SecurityTestStep Overview

POST 18	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 32	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 14	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 14	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 33	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 19	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 34	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 15	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 15	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts

SMARTBEAR ReadyAPI.

SecurityTestStep Overview

GET 35	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 20	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 36	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 16	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 16	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 37	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 21	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 38	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 17	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts

SMARTBEAR ReadyAPI.

SecurityTestStep Overview

DELETE 17	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 39	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 22	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 40	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
PUT 18	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 18	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 41	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
POST 23	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
GET 42	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts

SMARTBEAR ReadyAPI.

SecurityTestStep Overview

PUT 19	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts
DELETE 19	No Alerts
SQL Injection	No Alerts
Weak Authentication	No Alerts

SecurityTestStep Results

Status	Message
PASS	Step [GET] PASS: took 200 ms SecurityScan [SQL Injection] PASS: took 176 ms [SQL Injection] Request 1 - PASS: took 26 ms [SQL Injection] Request 2 - PASS: took 10 ms [SQL Injection] Request 3 - PASS: took 14 ms [SQL Injection] Request 4 - PASS: took 12 ms [SQL Injection] Request 5 - PASS: took 14 ms [SQL Injection] Request 6 - PASS: took 11 ms [SQL Injection] Request 7 - PASS: took 12 ms [SQL Injection] Request 8 - PASS: took 13 ms [SQL Injection] Request 9 - PASS: took 11 ms [SQL Injection] Request 10 - PASS: took 11 ms [SQL Injection] Request 11 - PASS: took 9 ms [SQL Injection] Request 12 - PASS: took 9 ms [SQL Injection] Request 13 - PASS: took 11 ms [SQL Injection] Request 14 - PASS: took 13 ms SecurityScan [Weak Authentication] PASS: took 31 ms [Weak Authentication] Request 1 - PASS - : took 31 ms
PASS	Step [POST] PASS: took 24 ms SecurityScan [SQL Injection] PASS: took 169 ms [SQL Injection] Request 1 - PASS: took 13 ms [SQL Injection] Request 2 - PASS: took 14 ms [SQL Injection] Request 3 - PASS: took 11 ms [SQL Injection] Request 4 - PASS: took 13 ms [SQL Injection] Request 5 - PASS: took 21 ms [SQL Injection] Request 6 - PASS: took 14 ms [SQL Injection] Request 7 - PASS: took 13 ms [SQL Injection] Request 8 - PASS: took 11 ms [SQL Injection] Request 9 - PASS: took 9 ms [SQL Injection] Request 10 - PASS: took 9 ms [SQL Injection] Request 11 - PASS: took 12 ms [SQL Injection] Request 12 - PASS: took 12 ms [SQL Injection] Request 13 - PASS: took 9 ms [SQL Injection] Request 14 - PASS: took 8 ms SecurityScan [Weak Authentication] PASS: took 22 ms [Weak Authentication] Request 1 - PASS - : took 22 ms

SMARTBEAR ReadyAPI.

SecurityTestStep Results

Status	Message
PASS	Step [PUT] PASS: took 15 ms SecurityScan [SQL Injection] PASS: took 132 ms [SQL Injection] Request 1 - PASS: took 11 ms [SQL Injection] Request 2 - PASS: took 10 ms [SQL Injection] Request 3 - PASS: took 7 ms [SQL Injection] Request 4 - PASS: took 12 ms [SQL Injection] Request 5 - PASS: took 7 ms [SQL Injection] Request 6 - PASS: took 8 ms [SQL Injection] Request 7 - PASS: took 8 ms [SQL Injection] Request 8 - PASS: took 7 ms [SQL Injection] Request 9 - PASS: took 7 ms [SQL Injection] Request 10 - PASS: took 7 ms [SQL Injection] Request 11 - PASS: took 8 ms [SQL Injection] Request 12 - PASS: took 9 ms [SQL Injection] Request 13 - PASS: took 24 ms [SQL Injection] Request 14 - PASS: took 7 ms SecurityScan [Weak Authentication] PASS: took 17 ms [Weak Authentication] Request 1 - PASS - : took 17 ms
PASS	Step [DELETE] PASS: took 14 ms SecurityScan [SQL Injection] PASS: took 169 ms [SQL Injection] Request 1 - PASS: took 9 ms [SQL Injection] Request 2 - PASS: took 14 ms [SQL Injection] Request 3 - PASS: took 8 ms [SQL Injection] Request 4 - PASS: took 22 ms [SQL Injection] Request 5 - PASS: took 20 ms [SQL Injection] Request 6 - PASS: took 20 ms [SQL Injection] Request 7 - PASS: took 8 ms [SQL Injection] Request 8 - PASS: took 13 ms [SQL Injection] Request 9 - PASS: took 8 ms [SQL Injection] Request 10 - PASS: took 14 ms [SQL Injection] Request 11 - PASS: took 8 ms [SQL Injection] Request 12 - PASS: took 7 ms [SQL Injection] Request 13 - PASS: took 10 ms [SQL Injection] Request 14 - PASS: took 8 ms SecurityScan [Weak Authentication] PASS: took 16 ms [Weak Authentication] Request 1 - PASS - : took 16 ms

SMARTBEAR ReadyAPI.

SecurityTestStep Results

Status	Message
PASS	Step [PATCH] PASS: took 17 ms SecurityScan [SQL Injection] PASS: took 167 ms [SQL Injection] Request 1 - PASS: took 9 ms [SQL Injection] Request 2 - PASS: took 6 ms [SQL Injection] Request 3 - PASS: took 7 ms [SQL Injection] Request 4 - PASS: took 20 ms [SQL Injection] Request 5 - PASS: took 7 ms [SQL Injection] Request 6 - PASS: took 20 ms [SQL Injection] Request 7 - PASS: took 15 ms [SQL Injection] Request 8 - PASS: took 11 ms [SQL Injection] Request 9 - PASS: took 17 ms [SQL Injection] Request 10 - PASS: took 8 ms [SQL Injection] Request 11 - PASS: took 8 ms [SQL Injection] Request 12 - PASS: took 7 ms [SQL Injection] Request 13 - PASS: took 20 ms [SQL Injection] Request 14 - PASS: took 12 ms SecurityScan [Weak Authentication] PASS: took 34 ms [Weak Authentication] Request 1 - PASS - : took 34 ms
PASS	Step [OPTIONS] PASS: took 22 ms SecurityScan [SQL Injection] PASS: took 164 ms [SQL Injection] Request 1 - PASS: took 7 ms [SQL Injection] Request 2 - PASS: took 5 ms [SQL Injection] Request 3 - PASS: took 17 ms [SQL Injection] Request 4 - PASS: took 15 ms [SQL Injection] Request 5 - PASS: took 15 ms [SQL Injection] Request 6 - PASS: took 14 ms [SQL Injection] Request 7 - PASS: took 10 ms [SQL Injection] Request 8 - PASS: took 20 ms [SQL Injection] Request 9 - PASS: took 7 ms [SQL Injection] Request 10 - PASS: took 9 ms [SQL Injection] Request 11 - PASS: took 10 ms [SQL Injection] Request 12 - PASS: took 9 ms [SQL Injection] Request 13 - PASS: took 12 ms [SQL Injection] Request 14 - PASS: took 14 ms SecurityScan [Weak Authentication] PASS: took 24 ms [Weak Authentication] Request 1 - PASS - : took 24 ms

SMARTBEAR ReadyAPI.

SecurityTestStep Results

Status	Message
PASS	Step [POST 2] PASS: took 50 ms SecurityScan [SQL Injection] PASS: took 848 ms [SQL Injection] Request 1 - PASS: took 27 ms [SQL Injection] Request 2 - PASS: took 17 ms [SQL Injection] Request 3 - PASS: took 15 ms [SQL Injection] Request 4 - PASS: took 16 ms [SQL Injection] Request 5 - PASS: took 14 ms [SQL Injection] Request 6 - PASS: took 12 ms [SQL Injection] Request 7 - PASS: took 11 ms [SQL Injection] Request 8 - PASS: took 10 ms [SQL Injection] Request 9 - PASS: took 19 ms [SQL Injection] Request 10 - PASS: took 22 ms [SQL Injection] Request 11 - PASS: took 27 ms [SQL Injection] Request 12 - PASS: took 30 ms [SQL Injection] Request 13 - PASS: took 28 ms [SQL Injection] Request 14 - PASS: took 25 ms [SQL Injection] Request 15 - PASS: took 80 ms [SQL Injection] Request 16 - PASS: took 41 ms [SQL Injection] Request 17 - PASS: took 24 ms [SQL Injection] Request 18 - PASS: took 30 ms [SQL Injection] Request 19 - PASS: took 20 ms [SQL Injection] Request 20 - PASS: took 13 ms [SQL Injection] Request 21 - PASS: took 14 ms [SQL Injection] Request 22 - PASS: took 12 ms [SQL Injection] Request 23 - PASS: took 15 ms [SQL Injection] Request 24 - PASS: took 13 ms [SQL Injection] Request 25 - PASS: took 14 ms [SQL Injection] Request 26 - PASS: took 21 ms [SQL Injection] Request 27 - PASS: took 12 ms [SQL Injection] Request 28 - PASS: took 35 ms [SQL Injection] Request 29 - PASS: took 31 ms [SQL Injection] Request 30 - PASS: took 16 ms [SQL Injection] Request 31 - PASS: took 17 ms [SQL Injection] Request 32 - PASS: took 12 ms [SQL Injection] Request 33 - PASS: took 12 ms [SQL Injection] Request 34 - PASS: took 12 ms [SQL Injection] Request 35 - PASS: took 12 ms [SQL Injection] Request 36 - PASS: took 32 ms [SQL Injection] Request 37 - PASS: took 13 ms [SQL Injection] Request 38 - PASS: took 12 ms [SQL Injection] Request 39 - PASS: took 15 ms [SQL Injection] Request 40 - PASS: took 25 ms [SQL Injection] Request 41 - PASS: took 11 ms [SQL Injection] Request 42 - PASS: took 11 ms SecurityScan [Weak Authentication] PASS: took 15 ms [Weak Authentication] Request 1 - PASS - : took 15 ms

SMARTBEAR ReadyAPI.

SecurityTestStep Results

Status	Message
PASS	Step [GET 2] PASS: took 28 ms SecurityScan [SQL Injection] PASS: took 220 ms [SQL Injection] Request 1 - PASS: took 11 ms [SQL Injection] Request 2 - PASS: took 11 ms [SQL Injection] Request 3 - PASS: took 10 ms [SQL Injection] Request 4 - PASS: took 26 ms [SQL Injection] Request 5 - PASS: took 25 ms [SQL Injection] Request 6 - PASS: took 10 ms [SQL Injection] Request 7 - PASS: took 25 ms [SQL Injection] Request 8 - PASS: took 17 ms [SQL Injection] Request 9 - PASS: took 11 ms [SQL Injection] Request 10 - PASS: took 18 ms [SQL Injection] Request 11 - PASS: took 10 ms [SQL Injection] Request 12 - PASS: took 21 ms [SQL Injection] Request 13 - PASS: took 15 ms [SQL Injection] Request 14 - PASS: took 10 ms SecurityScan [Weak Authentication] PASS: took 16 ms [Weak Authentication] Request 1 - PASS - : took 16 ms
PASS	Step [POST 3] PASS: took 23 ms SecurityScan [SQL Injection] PASS: took 193 ms [SQL Injection] Request 1 - PASS: took 9 ms [SQL Injection] Request 2 - PASS: took 9 ms [SQL Injection] Request 3 - PASS: took 29 ms [SQL Injection] Request 4 - PASS: took 11 ms [SQL Injection] Request 5 - PASS: took 15 ms [SQL Injection] Request 6 - PASS: took 10 ms [SQL Injection] Request 7 - PASS: took 9 ms [SQL Injection] Request 8 - PASS: took 9 ms [SQL Injection] Request 9 - PASS: took 8 ms [SQL Injection] Request 10 - PASS: took 8 ms [SQL Injection] Request 11 - PASS: took 16 ms [SQL Injection] Request 12 - PASS: took 14 ms [SQL Injection] Request 13 - PASS: took 19 ms [SQL Injection] Request 14 - PASS: took 27 ms SecurityScan [Weak Authentication] PASS: took 86 ms [Weak Authentication] Request 1 - PASS - : took 86 ms