PENGEMBANGAN ARSITEKTUR BERORIENTASI LAYANAN DENGAN RESTFUL API PADA SISTEM INFORMASI SEKOLAH MENENGAH PERTAMA MENGGUNAKAN LARAVEL

Disusun Oleh:

N a m a : Achmad Khodzim
NIM : 17523064

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2024

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN ARSITEKTUR BERORIENTASI LAYANAN DENGAN RESTFUL API PADA SISTEM INFORMASI SEKOLAH MENENGAH PERTAMA MENGGUNAKAN LARAVEL

TUGAS AKHIR

Disusun Oleh:

N a m a	: Achmad Khodzim
NIM	: 17523064

Yogyakarta, 11 Juli 2024

Pembimbing,

(Andhik Budi Cahyono, S.T, M.T)

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN ARSITEKTUR BERORIENTASI LAYANAN DENGAN RESTFUL API PADA SISTEM INFORMASI SEKOLAH MENENGAH PERTAMA MENGGUNAKAN LARAVEL

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 29 Agustus 2024

Tim Penguji

Andhik Budi Cahyono, S.T, M.T

Anggota 1

Sheila Nurul Huda, S.Kom., M.Cs.

Anggota 2

Irving Vitra Paputungan, S.T., M.Sc., Ph.D.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Ir. Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Achmad Khodzim

NIM : 17523064

Tugas akhir dengan judul:

PENGEMBANGAN ARSITEKTUR BERORIENTASI LAYANAN DENGAN RESTFUL API PADA SISTEM INFORMASI SEKOLAH MENENGAH PERTAMA MENGGUNAKAN LARAVEL

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 11 Juli 2024

(Achmad Khodzim)

HALAMAN PERSEMBAHAN

Saya persembahkan tugas akhir ini untuk:

Ibu saya tercinta **Hamsinah Hafid SE.** yang tiada hentinya memberikan semangat dan selalu mendoakan saya selama diperantauan jauh dari sanak keluarga. Berkat dukungan dari beliau yang sudah mengasuh saya dan kedua adik saya seorang diri semenjak meninggalnya Bapak 10 tahun lalu.

Tidak lupa saya mengucapkan terima kasih kepada:

Rizqi paser, Ocid, dan Mupid yang menjadi teman nongkrong dari awal dijogja sampai sekarang.

Dosen dan Staf prodi informatika uii yang telah memberikan support dan dorongan untuk mahasiswa tua seperti saya.

HALAMAN MOTTO

"Tidak ada kata terlambat untuk mencapai tujuan yang berharga. Selesaikan apa yang telah kamu mulai."

"Setiap detik yang kamu habiskan untuk berusaha membawa kamu lebih dekat pada impianmu. Teruslah berjuang hingga titik terakhir."

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Puji dan syukur penulis panjatkan ke hadirat Allah SWT atas rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul **"PENGEMBANGAN ARSITEKTUR BERORIENTASI LAYANAN DENGAN RESTFUL API PADA SISTEM INFORMASI SEKOLAH MENENGAH PERTAMA MENGGUNAKAN LARAVEL"**. Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi Teknik Informatika, Fakultas Teknik, Universitas Islam Indonesia.

Penulisan skripsi ini tidak terlepas dari bantuan, bimbingan, dan dukungan dari berbagai pihak. Oleh karena itu, pada kesempatan ini, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Bapak Andhik Budi Cahyono, S.T.,MT. , selaku dosen pembimbing yang telah memberikan bimbingan, arahan, dan motivasi kepada penulis dalam menyelesaikan skripsi ini.
2. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc. , selaku Ketua Program Studi Teknik Informatika, Fakultas Teknik, Universitas Islam Indonesia, yang telah memberikan kesempatan dan fasilitas dalam penyelesaian studi.
3. Bapak/Ibu Dosen dan Staf Pengajar di Program Studi Teknik Informatika, Fakultas Teknik, Universitas Islam Indonesia, yang telah memberikan ilmu dan pengetahuan selama masa studi.
4. Rekan-rekan Mahasiswa Teknik Informatika, yang telah memberikan bantuan dan semangat dalam proses penyusunan skripsi ini.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih terdapat kekurangan. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun untuk penyempurnaan skripsi ini di masa mendatang. Semoga skripsi ini dapat memberikan manfaat dan kontribusi positif bagi perkembangan ilmu pengetahuan, khususnya di bidang teknik informatika.

Yogyakarta, 11 Juli 2024

(Achmad Khodzim)

SARI

Penelitian ini bertujuan untuk mengembangkan sistem informasi akademik di tingkat Sekolah Menengah Pertama (SMP) menggunakan arsitektur berorientasi layanan (SOA) dengan framework Laravel. Sistem informasi akademik yang ada masih menggunakan pendekatan monolitik yang menyebabkan kesulitan dalam maintenance, scaling, dan pengembangan fitur baru. Pendekatan SOA dipilih karena memungkinkan layanan-layanan terkait dapat digunakan kembali dan diintegrasikan dengan mudah. Penelitian ini menghasilkan sepuluh layanan utama: TahunAjaran, Kurikulum, Kelas, MataPelajaran, Nilai, NilaiSiswa, Presensi, SiswaKelas, Siswa, dan Kaldik. Pengujian API dilakukan untuk memastikan bahwa layanan yang dikembangkan berfungsi dengan baik. Hasil penelitian menunjukkan bahwa sistem yang dikembangkan lebih fleksibel dan mudah di-maintenance dibandingkan sistem monolitik.

Kata Kunci: Service-Oriented Architecture, Laravel, Sistem Informasi Akademik, Sekolah Menengah Pertama, API Testing.

GLOSARIUM

API (Application Programming Interface)	Sekumpulan definisi dan protokol untuk membangun dan mengintegrasikan perangkat lunak aplikasi.
Arsitektur Berorientasi Layanan (SOA)	Pendekatan untuk merancang perangkat lunak dengan mengemas fungsionalitas ke dalam layanan yang terpisah dan mandiri.
Docker	Platform untuk mengembangkan, mengirimkan, dan menjalankan aplikasi dalam kontainer.
Framework Laravel	Framework PHP yang menggunakan konsep MVC untuk pengembangan aplikasi web.
Kurikulum	Rencana pembelajaran yang mencakup tujuan, isi, dan proses pendidikan yang harus dicapai oleh siswa.
Monolitik	Sistem perangkat lunak yang dirancang sebagai satu kesatuan yang tidak terpisahkan.
Persistence Service	Layanan yang tidak bergantung pada layanan lain tetapi diakses oleh layanan lain untuk menyimpan dan mengambil data.
Aggregate Service	Layanan yang tergantung pada layanan lain
Sail	Alat untuk menjalankan Laravel dalam lingkungan Docker.

DAFTAR ISI

HALAMAN PENGESAHAN DOSEN PEMBIMBING ii
HALAMAN PENGESAHAN DOSEN PENGUJI iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR iv
HALAMAN PERSEMBAHAN v
HALAMAN MOTTO vi
KATA PENGANTAR vii
SARI viii
GLOSARIUM ix
DAFTAR ISI x
DAFTAR TABEL xii
DAFTAR GAMBAR xiii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan Penelitian 2
1.5 Manfaat Penelitian 2
1.6 Metode Penelitian 2
1.7 Sistematika Penulisan 3
BAB II LANDASAN TEORI 5
2.1 *Service Oriented Architecture* (SOA) 5
2.2 Web Service 5
2.3 Laravel Framework 6
2.4 Api Testing 6
BAB III ANALISIS DAN PERANCANGAN 7
3.1 Analisis Sistem Monolitik 7
3.1.1 Analisis Domain 7
3.1.2 Analisis Database 9
3.2 Dekomposisi 10
3.2.1 Identifikasi Domain Bisnis 10
3.2.2 Pemecahan Layanan Berdasarkan Fungsi 10
3.2.3 Definisi Interaksi Antar Layanan 10
3.3 Desain Layanan 11
3.4 Perancangan *API Endpoint* Layanan 12
3.5 Rancangan Pengujian 19
BAB IV PENGEMBANGAN DAN PENGUJIAN 20
4.1 Pengembangan Layanan 20
4.1.1 Inisiasi Project Laravel Layanan 20
4.1.2 Melakukan Pengaturan Pada File Konfigurasi Docker 20
4.1.3 Pembuatan *Database* 21
4.1.4 Pembuatan *Controller* 25
4.2 Hasil Pengembangan 27
4.2.1 Daftar *API* TahunAjaranService 28
4.2.2 Daftar *API* KurikulumService 28
4.2.3 Daftar *API* KelasService 29
4.2.4 Daftar *API* MataPelajaranService 30
4.2.5 Daftar *API* NilaiService 31

4.2.6 Daftar *API* NilaiSiswaService 32
4.2.7 Daftar *API* PresensiService 33
4.2.8 Daftar *API* SiswaKelasRegulerBerjalanService 34
4.2.9 Daftar *API* SiswaService 35
4.2.10 Daftar *API* KaldikService 35
4.3 Pengujian Layanan 35
4.3.1 Hasil Skenario Pengujian 36
4.3.2 Kesimpulan Pengujian 41
BAB V KESIMPULAN 42
5.1 Kesimpulan 42
5.2 Saran 42
DAFTAR PUSTAKA 43
LAMPIRAN 44

DAFTAR TABEL

Tabel 3.1 Tabel Rancangan Layanan 11
Tabel 3.2 Tabel *Endpoint* TahunAjaranService 13
Tabel 3.3 Tabel *Endpoint* KurikulumService 14
Tabel 3.4 Tabel *Endpoint* KelasService modul kelas reguler 14
Tabel 3.5 Tabel *Endpoint* KelasService modul kelas reguler berjalan 14
Tabel 3.6 Tabel *Endpoint* MataPelajaranService modul nama mapel 15
Tabel 3.7 Tabel *Endpoint* MataPelajaranService modul MataPelajaran 15
Tabel 3.8 Tabel *Endpoint* NilaiService modul Deskripsi Nilai 16
Tabel 3.9 Tabel *Endpoint* NilaiService modul Kategori Nilai 16
Tabel 3.10 Tabel *Endpoint* NilaiService modul Jenis Nilai Akhir 16
Tabel 3.11 Tabel *Endpoint* NilaiSiswaService 17
Tabel 3.12 Tabel *Endpoint* PresensiService 17
Tabel 3.13 Tabel *Endpoint* SiswaKelasRegulerBerjalanService 18
Tabel 3. 14 Tabel *Endpoint* SiswaService 18
Tabel 3.15 Tabel *Endpoint* KaldikService 19
Tabel 4.1 Tabel Daftar *Port* Layanan 21
Tabel 4.2 Tabel daftar detail *controller* 25
Tabel 4.3 Tabel Daftar *API* TahunAjaranService 28
Tabel 4.4 Tabel Daftar *API* KurikulumService 28
Tabel 4.5 Tabel Daftar *API* KelasService 29
Tabel 4.6 Tabel Daftar *API* MataPelajaranService 30
Tabel 4.7 Tabel Daftar *API* NilaiService 31
Tabel 4.8 Tabel Daftar *API* NilaiSiswaService 32
Tabel 4.9 Tabel Daftar *API* PresensiService 33
Tabel 4.10 Tabel Daftar *API* SiswaKelasRegulerBerjalanService 34
Tabel 4.11 Tabel Daftar *API* SiswaService 35
Tabel 4.12 Tabel Daftar *API* KaldikService 35
Tabel 4.13 Tabel Daftar Hasil Skenario Pengujian 36

DAFTAR GAMBAR

Gambar 3.1 Tahapan Analisis.....7
Gambar 3.2 Desain Arsitektur Monolitik Sistem Informasi Akademik 8
Gambar 3.3 Relasi Tabel.....9
Gambar 3.4 Diagram Desain Layanan.....11
Gambar 4.1 Isi file docker-compose.yml.....21
Gambar 4.2 Kode model TahunAjaran. 22
Gambar 4.3 Hasil Pengujian Api Testing 41

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam era digital saat ini, teknologi informasi telah menjadi salah satu elemen kunci dalam mendukung berbagai aspek kehidupan, termasuk dalam bidang pendidikan. Penggunaan teknologi informasi yang tepat dapat meningkatkan efisiensi dan efektivitas dalam pengelolaan data, informasi, serta proses administrasi di lingkungan sekolah. Sekolah Menengah Pertama (SMP) sebagai salah satu jenjang pendidikan formal memiliki kebutuhan yang kompleks dalam hal pengelolaan data siswa, guru, kurikulum, nilai, presensi, dan berbagai aktivitas lainnya.

Namun, sistem informasi yang digunakan di banyak sekolah masih berbasis monolitik, di mana semua fungsi dan layanan terintegrasi dalam satu kesatuan yang sulit untuk diubah dan dikembangkan secara fleksibel. Pendekatan ini sering kali menyebabkan permasalahan seperti kesulitan dalam maintenance, skala, dan pengembangan fitur baru yang sesuai dengan kebutuhan yang berkembang.

Pengembangan *service* pada penelitian ini akan dibuat menggunakan *Service Oriented Architecture (SOA).* Salah satu pertimbangan penting dalam pemilihan SOA adalah bagaimana layanan terkait dapat digunakan kembali dalam sistem lain(Giao et al., 2022), sebagai contoh modul pendaftaran dapat memberikan akses data calon siswa untuk digunakan pada modul atau service akademik seperti presensi, pemilihan kelas, dan sebagainya. Kemudahan integrasi inilah yang menjadi pertimbangan penting dalam pemilihan SOA dalam penelitian ini.

Dalam penerapan *service* pada penelitian ini menggunakan framework Laravel, Laravel merupakan framework yang berbasis bahasa pemrograman PHP yang bersifat Open-source yang ditujukan untuk mengembangkan aplikasi berbasis web. Penggunaan Laravel memberikan pengembang kemampuan untuk membangun aplikasi yang kompleks karena menyediakan banyak peralatan dan komponen yang sering di gunakan dan mudah dipahami oleh pengembang(Chen et al., 2017).

Pada penelitian kali ini, penulis akan meneruskan penelitian sebelumnya yang sudah dikerjakan dari tahap perancangan sampai menjadi sebuah sistem informasi akademik sekolah menengah pertama oleh (SIDDIQ, 2018).

1.2 Rumusan Masalah

Berdasarkan uraian masalah yang telah ada pada latar belakang tersebut maka rumusan masalah dalam penelitian ini adalah bagaimana cara merancang dan mengembangkan arsitektur berorientasi layanan menggunakan RESTful API dengan Laravel dari sistem informasi akademik SMP yang masih menggunakan sistem monolitik?

1.3 Batasan Masalah

a. Penelitian ini ini hanya berfokus dalam pembuatan RESTful API untuk sistem informasi akademik Sekolah Menengah Pertama (SMP) berdasarkan data penelitian sebelumnya.
b. Fokus pengembangan adalah pada layanan utama seperti pengelolaan data siswa, kelas, kurikulum, nilai, dan presensi.
c. *Framework* yang digunakan untuk pengembangan adalah Laravel, yang berbasis PHP.

1.4 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah sebagai berikut:

a. Mengembangkan sistem informasi akademik SMP dari Sistem monolitik menjadi sistem berbasis layanan.
b. Mengubah sistem informasi akademik SMP yang sebelumnya berbasis monolitik menjadi berbasis layanan yang modular dan fleksibel.
c. Menganalisis dan mengevaluasi kinerja RESTful API yang dihasilkan dari pengembangan menggunakan framework Laravel.

1.5 Manfaat Penelitian

Dengan adanya penelitian ini penulis berharap akan memberikan manfaat bagi pengembang untuk menjadi contoh implementasi SOA menggunakan Laravel yang dapat digunakan sebagai referensi dan menjadi acuan untuk penelitian lanjutan terkait pengembangan sistem informasi sekolah menengah pertama berbasis layanan ini.

1.6 Metode Penelitian

a. Analisis

Melalui rancangan yang sebelumnya telah dibuat oleh Muhammad Hafiz Siddiq (SIDDIQ, 2018) berupa hasil implementasi sistem informasi sekolah menengah pertama, penulis akan

mempelajari terkait struktur dan modul-modul yang telah diterapkan pada penelitian tersebut.

b. Perancangan

Tahapan perancangan adalah pemberian hasil dari Analisis kebutuhan terhadap layanan-layanan yang telah ditentukan pada tahap sebelumnya. Pada tahap ini juga akan ditentukan terkait *library* dan *tools* yang akan digunakan dalam perancangan setiap layanan yang akan dibangun.

c. Penulisan Kode Program

Berdasarkan perancangan yang sudah ada, maka pada tahap ini akan diimplementasikan menggunakan *framework* Laravel serta penggunaan *library* dan *tools* yang telah ditentukan.

d. Pengujian

Pada tahap ini akan menggunakan pengujian *API Testing* untuk mengetahui apakah layanan telah dapat berjalan dengan baik.

1.7 Sistematika Penulisan

Sistematika penulisan dalam penyusunan laporan tugas akhir ini terdiri dari beberapa bab, yang mencakup gambaran dari keseluruhan masalah dan penyelesaiannya. Berikut sistematika penulisan yang terbagi dalam 5 bab :

a. BAB I PENDAHULUAN

Pada bab ini menerangkan latar belakang, perumusan masalah, tujuan, batasan masalah, manfaat, metodologi, dan sistematika penulisan.

b. BAB II LANDASAN TEORI

Pada bab ini berisikan teori teori yang relevan yang melengkapi latar belakang sekaligus memberi review tentang pustaka yang telah dibaca selama pencarian solusi terhadap masalah yang diangkat dalam penelitian ini.

c. BAB III ANALISIS DAN PERANCANGAN

Bab ini berisi uraian tentang analisis, desain dan rancangan *service* dari aplikasi monolitik pada penelitian sebelumnya.

d. BAB IV PENGEMBANGAN DAN PENGUJIAN

Bagian ini akan memuat hasil pengembangan *service* yang dibuat beserta hasil dari pengujian.

e. BAB V KESIMPULAN DAN SARAN

Bab ini merupakan bab terakhir yang akan membahas kesimpulan dan saran terhadap penelitian yang telah dilakukan pada tugas akhir.

BAB II
LANDASAN TEORI

2.1 *Service Oriented Architecture* (SOA)

Service-Oriented Architecture (SOA) adalah sebuah pendekatan untuk mengembangkan sistem komputer yang terdiri dari beberapa layanan yang terpisah namun saling terintegrasi. Layanan-layanan tersebut dapat diakses melalui protokol standar seperti HTTP atau *Web Services Description Language* (WSDL)(Warkim & Sensuse, 2017). Arsitektur ini menekankan pada pembagian tugas antar layanan yang terpisah, sehingga setiap layanan dapat dikembangkan, diperbaiki, atau dikembalikan tanpa mempengaruhi layanan lain. Ini memungkinkan pengembangan sistem yang lebih cepat dan mudah diubah sesuai dengan kebutuhan, oleh karena itu arsitektur ini sering digunakan untuk mengembangkan aplikasi web, karena memungkinkan pembagian tugas antar tim yang lebih efektif dan integrasi dengan sistem lain yang lebih mudah.

2.2 Web Service

Web service adalah sebuah layanan yang disediakan oleh suatu sistem atau aplikasi yang dapat diakses melalui jaringan internet. *Web service* menggunakan protokol standar seperti HTTP (*Hypertext Transfer Protocol*) atau HTTPS (*Hypertext Transfer Protocol Secure*) untuk bertukar informasi dan data (Maulidiansyah et al., 2017). *Web service* biasanya digunakan untuk menghubungkan sistem atau aplikasi yang berbeda dan dapat diintegrasikan dengan mudah. Keuntungan dari penggunaan *web service* dalam pengembangan sistem informasi sekolah menengah pertama adalah:

a. Fleksibilitas: *Web service* memungkinkan aplikasi yang terpisah untuk saling berkomunikasi, sehingga sistem informasi sekolah menengah pertama dapat diintegrasikan dengan sistem lain yang ada di sekolah, seperti sistem pembayaran, sistem penjadwalan, atau sistem lain yang terkait dengan sistem informasi sekolah menengah pertama.
b. Interoperabilitas: *Web service* menggunakan protokol standar yang dapat digunakan oleh aplikasi apa pun, sehingga aplikasi pendaftaran sekolah menengah pertama dapat diakses oleh sistem lain yang menggunakan protokol yang sama.

c. *Scalability*: *Web service* dapat diakses oleh banyak aplikasi secara bersamaan, sehingga sistem informasi sekolah menengah pertama dapat menangani banyak permintaan secara efisien.

2.3 Laravel Framework

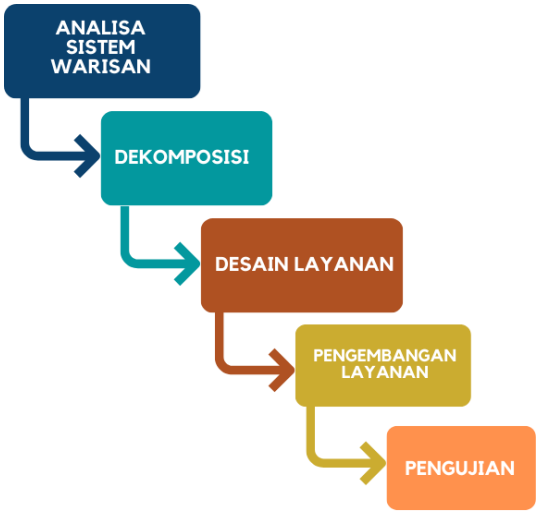
Laravel adalah sebuah framework PHP open-source yang menggunakan konsep MVC (*Model-View-Controller*). Dikembangkan oleh Taylor Otwell, Laravel bertujuan untuk membuat pengembangan aplikasi web lebih cepat dan efisien dengan menyediakan berbagai fitur seperti Eloquent ORM untuk interaksi database, sistem routing yang fleksibel, Blade templating engine, middleware, dan Artisan CLI untuk tugas rutin. Dengan mengadopsi arsitektur MVC, Laravel memisahkan logika bisnis, tampilan, dan kontrol alur aplikasi, sehingga memudahkan pengembang dalam membangun dan memelihara aplikasi web yang kompleks dan terstruktur.

2.4 Api Testing

API testing adalah proses pengujian *Application Programming Interfaces* (API) untuk memastikan bahwa mereka berfungsi sebagaimana mestinya. Pengujian ini melibatkan pengiriman permintaan (*request*) ke API dan memeriksa respons (*response*) untuk validasi fungsionalitas, keandalan, kinerja, dan keamanan.

BAB III
ANALISIS DAN PERANCANGAN

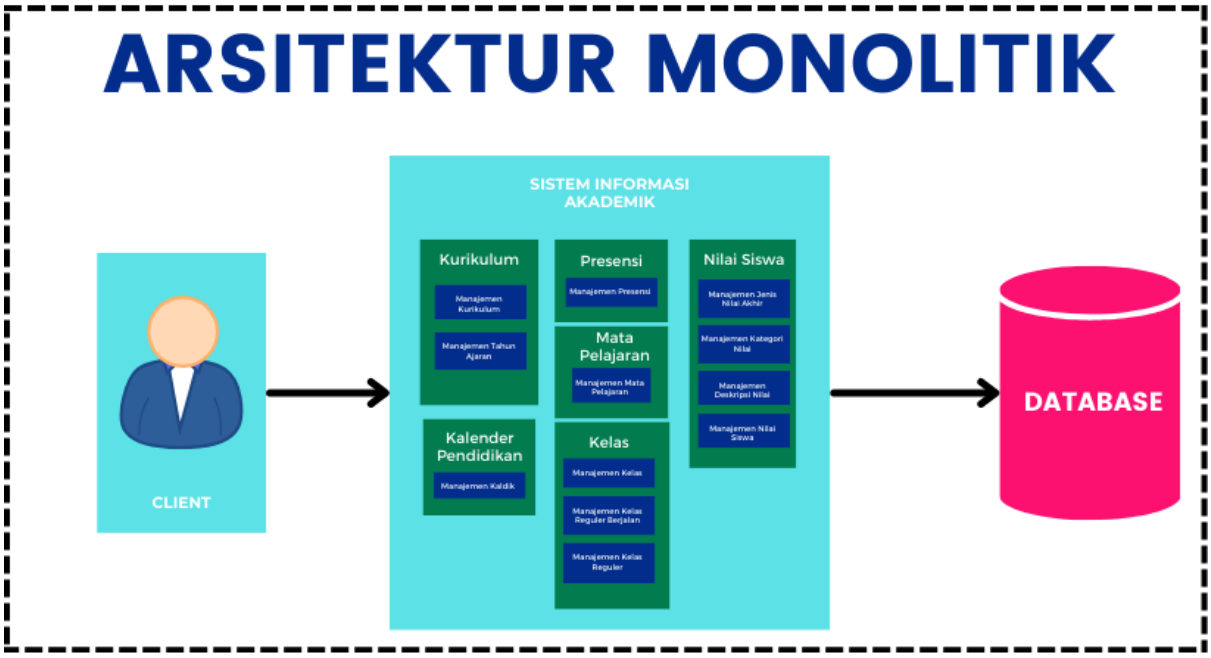
3.1 Analisis Sistem Monolitik



Gambar 3.1 Tahapan Analisis

3.1.1 Analisis Domain

Langkah ini merupakan tahapan awal yang krusial dalam proses transisi dari sistem monolitik ke arsitektur berbasis layanan. Analisis sistem monolitik dilakukan untuk memahami secara mendalam proses bisnis yang berjalan pada sistem lama serta bagaimana data dan informasi dikelola dan didistribusikan dalam sistem tersebut. Hasil dari analisis ini akan menjadi dasar penting dalam merancang dan membangun sistem berbasis layanan pada penelitian ini.



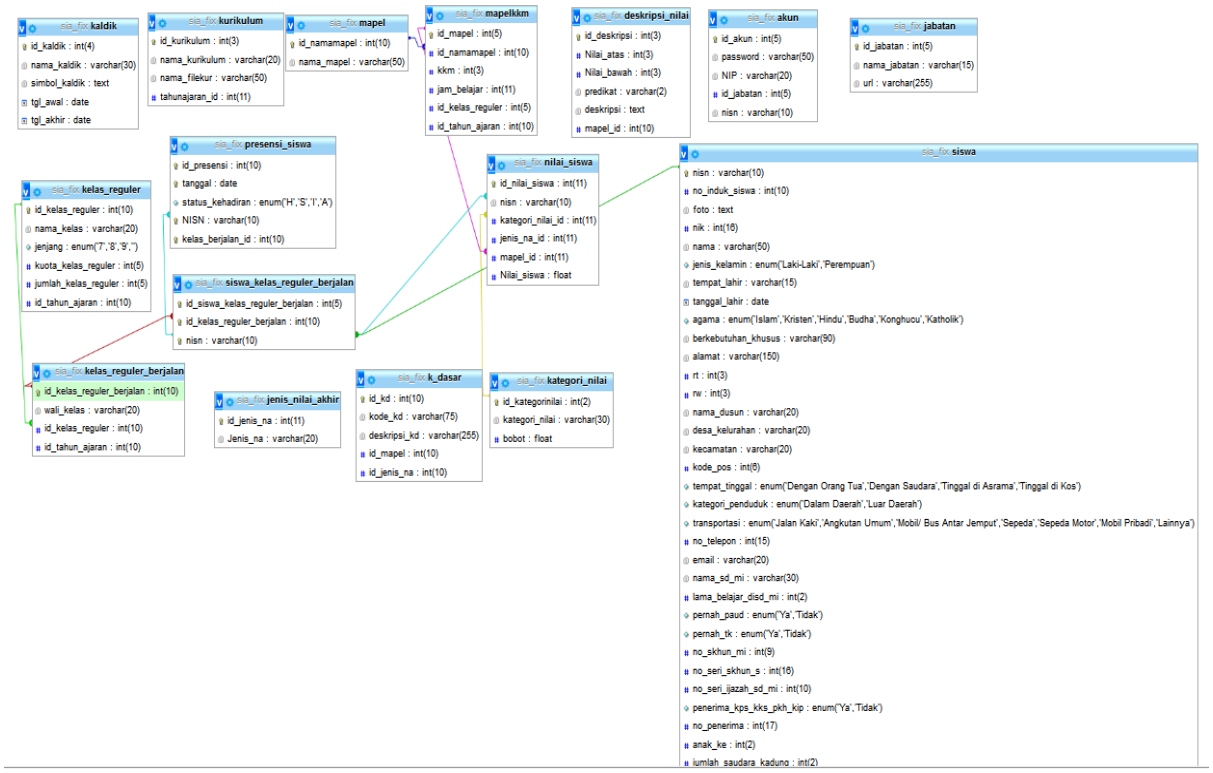
Gambar 3.2 Desain Arsitektur Monolitik Sistem Informasi Akademik

Pada Gambar 3.2 dapat dilihat bahwa seluruh proses dibuat dalam sebuah aplikasi tunggal dan beroperasi sebagai satu kesatuan yang tidak dapat dipisahkan. Artinya jika terjadi sebuah kesalahan pada salah satu fungsionalitas maka hal tersebut akan mengganggu atau berpengaruh pada fungsionalitas lainnya.

Pada sistem monolitik dibagi menjadi 6 domain utama yang memiliki subdomain masing-masing, domain-domain tersebut adalah:

a. Domain Kurikulum : manajemen kurikulum, manajemen tahun ajaran.
b. Domain Presensi: manajemen presensi siswa
c. Domain Nilai Siswa: Manajemen nilai akhir, manajemen kategori nilai, manajemen deskripsi nilai, manajemen nilai siswa.
d. Domain Mata Pelajaran: manajemen mata pelajaran.
e. Domain Kalender Pendidikan: manajemen kalender Pendidikan.
f. Domain Kelas: manajemen kelas, manajemen kelas reguler, manajemen kelas reguler berjalan.

3.1.2 Analisis Database



Gambar 3.3 Relasi Tabel

Sumber: (SIDDIQ, 2018)

Berdasarkan hasil Analisis dari database sistem monolitik terdapat 16 tabel dan beberapa memiliki relasi antar tabel. Pada Analisis ini peneliti akan mengelompokkan tabel berdasarkan domain yang menggunakan tabelnya.

- Domain Kurikulum :tabel kurikulum dan tabel kaldik.
- Domain Presensi: tabel presensi_siswa.
- Domain Nilai Siswa: tabel kategori_nilai, tabel jenis_nilai_akhir, tabel deskripsi_nilai dan tabel nilai_siswa.
- Domain Mata Pelajaran: tabel mapel dan tabel mapel_kkm.
- Domain Kalender Pendidikan: tabel kaldik.
- Domain Kelas: tabel kelas_reguler,tabel kelas_reguler_berjalan, dan tabel siswa_kelas_reguler_berjalan.

Terdapat 13 tabel yang telah dikelompokkan berdasar domainnya dan untuk 3 yang belum termasuk pada pengelompokan yaitu tabel siswa, tabel jabatan, dan tabel akun. Untuk tabel siswa belum terdapat domain yang mengelola tabel ini dan pada sistem sebelumnya tabel ini hanya digunakan sebagai tabel yang menyimpan data *dummy* dan digunakan sebagai relasi untuk domain presensi dan domain nilai siswa. Sedangkan untuk tabel akun dan

jabatan tidak akan digunakan untuk penelitian ini karna berada di luar cakupan fungsionalitas yang akan dibahas pada penelitian ini.

3.2 Dekomposisi

Dekomposisi merupakan tahapan penting dalam proses transisi dari sistem monolitik ke arsitektur berbasis layanan. Berdasarkan hasil analisis sistem monolitik, dekomposisi dilakukan dengan tujuan memecah sistem monolitik menjadi layanan-layanan kecil yang independen, tetapi tetap saling terintegrasi. Pendekatan *database-centric* digunakan sebagai dasar dalam melakukan dekomposisi ini, di mana setiap layanan yang dihasilkan akan berfokus pada pengelolaan bagian spesifik dari database sistem monolitik.

3.2.1 Identifikasi Domain Bisnis

Langkah pertama dalam dekomposisi adalah mengidentifikasi domain bisnis utama yang ada dalam sistem monolitik. Berdasarkan diagram sistem monolitik yang dihasilkan dari analisis, domain-domain bisnis seperti manajemen siswa, manajemen nilai, manajemen kurikulum, dan manajemen presensi dapat diidentifikasi.

Setiap domain bisnis ini akan dipecah menjadi satu atau beberapa layanan, tergantung pada kompleksitas dan fungsi yang dimiliki. Domain bisnis ini menjadi fondasi dalam menentukan batasan setiap layanan.

3.2.2 Pemecahan Layanan Berdasarkan Fungsi

Setelah domain bisnis diidentifikasi, langkah berikutnya adalah memecah fungsi-fungsi yang ada dalam sistem monolitik menjadi layanan-layanan yang lebih kecil.

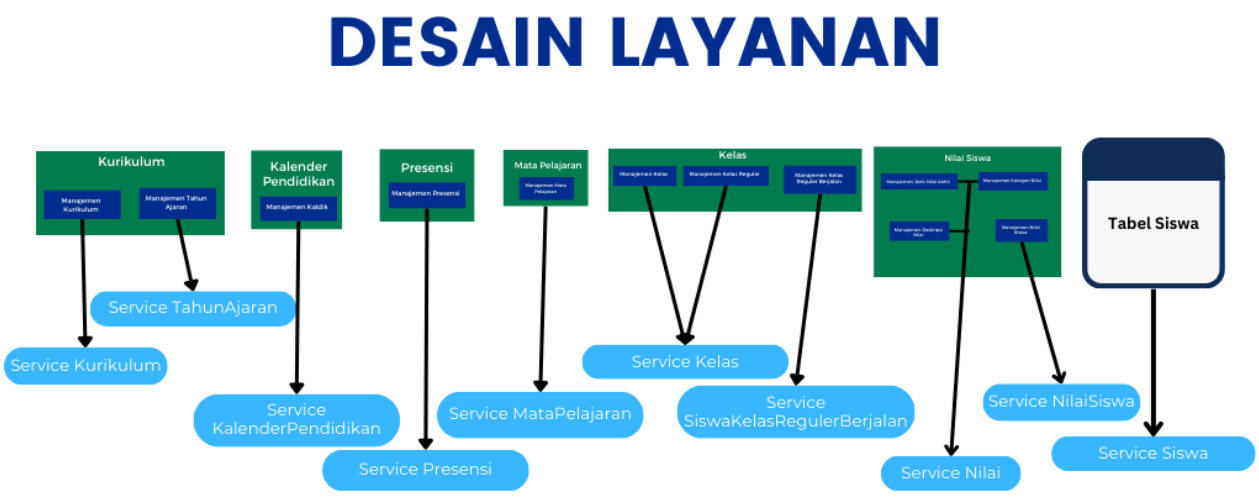
Berdasarkan diagram desain database dari sistem monolitik, setiap layanan akan dirancang untuk memiliki tanggung jawab yang jelas terhadap satu atau beberapa tabel dalam database, sesuai dengan prinsip database-centric.

3.2.3 Definisi Interaksi Antar Layanan

Dalam arsitektur layanan, layanan-layanan yang dihasilkan dari dekomposisi harus dapat berinteraksi satu sama lain untuk membentuk ekosistem yang fungsional.

Berdasarkan pemetaan proses bisnis dari sistem monolitik, interaksi antar domain ini didefinisikan dengan jelas dengan bentuk relasi antar tabel. Relasi relasi tersebut dapat dilihat pada tiap tabel yang mereferensi id dari tabel lain dengan contoh id_tahun_ajaran yang di referensi pada tabel kurikulum, tabel mapel_kkm, tabel kelas_reguler, dan tabel kelas_reguler_berjalan. Dan relasi-relasi lain yang dapat dilihat lebih detail pada Gambar 3.3.

3.3 Desain Layanan



Gambar 3.4 Diagram Desain Layanan

Berdasarkan hasil dari dekomposisi yang telah dilakukan maka layanan akan dibagi berdasarkan kapabilitas bisnisnya, layanan-layanan tersebut dapat dilihat pada

Tabel 3.1. Pembagian layanan-layanan ini berdasarkan kapabilitas bisnis ini tidak hanya meningkatkan efisiensi operasional tetapi juga memudahkan dalam pemeliharaan, pengembangan, dan integrasi sistem di masa depan. Setiap layanan dapat dikembangkan dan dikelola secara independen, sesuai dengan kebutuhan spesifiknya, yang pada akhirnya memberikan fleksibilitas dan ketangguhan yang lebih besar pada sistem secara keseluruhan.

Tabel 3.1 Tabel Rancangan Layanan

Domain Asal	Domain Asal	Cakupan Subdomain	Penamaan	Keterangan
Service Kalender Pendidikan	Kalender Pendidikan	Manajemen Kalender Pendidikan	KaldikService	fungsinya terfokus dan tidak memerlukan interaksi yang kompleks dengan *subdomain* lain, sehingga bisa dikelola secara mandiri.
Service Kelas	Kelas	Manajemen Kelas, Manajemen Kelas Reguler	KelasService	keterkaitan antar subdomain membuat penggabungan dalam satu layanan meningkatkan efisiensi dan integrasi proses.
Service Kurikulum	Kurikulum	Manajemen Kurikulum	KurikulumService	fungsinya terfokus dan tidak memerlukan interaksi yang kompleks dengan subdomain lain, sehingga bisa dikelola secara mandiri.

Service MataPelajaran	Mata Pelajaran	Manajemen Nama Mata Pelajaran, Manajemen Mata Pelajaran	MataPelajaranService	Kedua subdomain memiliki keterikatan langsung berdasarkan bentuk relasi database.
Service Nilai	Nilai Siswa	Manajemen Jenis Nilai Akhir, Manajemen Kategori Nilai, Manajemen Deskripsi Nilai	NilaiService	fungsionalitas ketiga subdomain ini sangat terkait dan sering digunakan bersama, sehingga menggabungkannya meningkatkan konsistensi data dan integrasi.
Service Nilai Siswa	Nilai Siswa	Manajemen Nilai Siswa	NilaiSiswaService	subdomain ini spesifik untuk pengelolaan nilai individu siswa, yang memerlukan fokus dan kemandirian dalam pengelolaannya.
Service Presensi	Presensi	Manajemen Presensi	PresensiService	Membutuhkan kemandirian fungsional.
Service Siswa Kelas Reguler Berjalan	Kelas	Manajemen Siswa Kelas Reguler Berjalan	SiswaKelasService	Layanan ini menangani pengelolaan siswa yang sedang mengikuti kelas reguler. Dengan memisahkan layanan ini, kita dapat lebih fokus pada pemantauan dan administrasi siswa yang terdaftar dalam kelas yang sedang berjalan.
Service Siswa	Tabel Siswa	Manajemen Siswa	SiswaService	menangani pengelolaan data siswa secara keseluruhan, yang membutuhkan pengelolaan mandiri untuk menjaga keamanan dan integritas data.
Service Tahun Ajaran	Kurikulum	Manajemen Tahun Ajaran	TahunAjaranService	pengelolaan tahun ajaran adalah fungsi spesifik yang memerlukan fokus tersendiri, terpisah dari aspek kurikulum lainnya.

3.4 Perancangan *API Endpoint* Layanan

Perancangan RESTful API merupakan langkah penting dalam pengembangan arsitektur berorientasi layanan, khususnya dalam konteks sistem informasi akademik. RESTful API (Representational State Transfer) adalah sebuah standar arsitektur yang digunakan dalam pengembangan layanan web, yang memungkinkan berbagai aplikasi untuk saling berkomunikasi dan bertukar data secara efisien melalui HTTP (Hypertext Transfer Protocol).

Dalam merancang RESTful API, terdapat beberapa komponen kunci yang harus dipertimbangkan, antara lain desain endpoint, pengelolaan sumber daya, serta penerapan

metode HTTP yang sesuai dengan operasi yang dilakukan. Tujuan utama dari perancangan ini adalah untuk memastikan API yang dihasilkan memiliki performa yang baik, mudah diakses, serta mematuhi prinsip-prinsip REST.

RESTful API menggunakan empat metode HTTP utama yang sering digunakan dalam pengelolaan sumber daya, yaitu:

a. *GET* digunakan untuk mengambil atau membaca data dari server tanpa mengubah sumber daya tersebut.
b. *POST* digunakan untuk membuat atau menambahkan sumber daya baru di *server*.
c. *PUT* digunakan untuk memperbarui atau mengubah data yang sudah ada di *server*.
d. *DELETE* digunakan untuk menghapus sumber daya dari *server*.

Dalam proses perancangan, penting untuk memastikan bahwa setiap endpoint didokumentasikan dengan baik dan *API* yang dikembangkan dapat dengan mudah diuji dan diintegrasikan dengan berbagai platform. Hal ini akan meningkatkan kualitas sistem secara keseluruhan dan memastikan bahwa sistem informasi akademik dapat diakses dan digunakan oleh semua pemangku kepentingan dengan optimal.

Adapun rancangan *API* dari layanan yang akan dikembangkan adalah sebagai berikut:

a. Rancangan *API* TahunAjaranService

Tabel 3.2 Tabel *Endpoint* TahunAjaranService

Metode	***Endpoint***	**Kebutuhan**
POST	*/api/tahun-ajarans*	Menambah tahun ajaran baru.
GET	*/api/tahun-ajarans*	Mendapatkan daftar semua tahun ajaran.
GET	*/api/tahun-ajarans/:id*	Mendapatkan tahun ajaran berdasarkan ID.
PUT	*/api/tahun-ajarans/:id*	Memperbarui tahun ajaran berdasarkan ID.
DELETE	*/api/tahun-ajarans/:id*	Menghapus tahun ajaran

b. Rancangan *API* KurikulumService

Tabel 3.3 Tabel *Endpoint* KurikulumService

Method	***Endpoint***	**Kebutuhan**
POST	*/api/kurikulums*	Menambah kurikulum baru.
GET	*/api/kurikulums*	Mendapatkan daftar semua kurikulum.
GET	*/api/kurikulums/:id*	Mendapatkan kurikulum berdasarkan ID.
PUT	*/api/kurikulums/:id*	Memperbarui kurikulum berdasarkan ID.
DELETE	*/api/kurikulums/:id*	Menghapus kurikulum berdasarkan ID.

c. Rancangan *API* KelasService

Tabel 3.4 Tabel *Endpoint* KelasService modul kelas reguler

Metode	***Endpoint***	**Kebutuhan**
POST	/api/kelas-regulers	Menambah kelas reguler baru.
GET	/api/kelas-regulers	Mendapatkan daftar semua kelas reguler.
GET	/api/kelas-regulers/:id	Mendapatkan kelas reguler berdasarkan ID.
PUT	/api/kelas-regulers/:id	Memperbarui kelas reguler berdasarkan ID.
DELETE	/api/kelas-regulers/:id	Menghapus kelas reguler berdasarkan ID.

Tabel 3.5 Tabel *Endpoint* KelasService modul kelas reguler berjalan

Metode	***Endpoint***	**Kebutuhan**
POST	/api/kelas-reguler-berjalans	Menambah kelas reguler berjalan baru.
GET	/api/kelas-reguler-berjalans	Mendapatkan daftar semua kelas reguler berjalan.
GET	/api/kelas-reguler-berjalans/:id	Mendapatkan kelas reguler berjalan berdasarkan ID.

PUT	/api/kelas- reguler-berjalans/:id	Memperbarui kelas reguler berjalan berdasarkan ID.
DELETE	/api/kelas- reguler-berjalans/:id	Menghapus kelas reguler berjalan berdasarkan ID.

d. Rancangan *API* MataPelajaranService

Tabel 3.6 Tabel *Endpoint* MataPelajaranService modul nama mapel

Metode	***Endpoint***	**Kebutuhan**
POST	*/api/nama-mapels*	Menambah nama mata pelajaran baru.
GET	*/api/nama-mapels*	Mendapatkan daftar semua nama mata pelajaran.
GET	*/api/nama-mapels /:id*	Mendapatkan nama mata pelajaran berdasarkan ID.
PUT	*/api/nama-mapels /:id*	Memperbarui nama mata pelajaran berdasarkan ID.
DELETE	*/api/nama-mapels /:id*	Menghapus nama mata pelajaran berdasarkan ID.

Tabel 3.7 Tabel *Endpoint* MataPelajaranService modul MataPelajaran

Metode	***Endpoint***	**Kebutuhan**
POST	*/api/mata-pelajarans*	Menambah mata pelajaran baru.
GET	*/api/mata-pelajarans*	Mendapatkan daftar semua mata pelajaran.
GET	*/api/mata-pelajarans/:id*	Mendapatkan mata pelajaran berdasarkan ID.
PUT	*/api/mata-pelajarans/:id*	Memperbarui mata pelajaran berdasarkan ID.

DELETE	*/api/mata-pelajarans/:id*	Menghapus mata pelajaran berdasarkan ID.

e. Rancangan *API* NilaiService

Tabel 3.8 Tabel *Endpoint* NilaiService modul Deskripsi Nilai

Metode	***Endpoint***	**Kebutuhan**
POST	*/api/deskripsi-nilais*	Menambah deskripsi nilai baru.
GET	*/api/deskripsi-nilais*	Mendapatkan daftar semua deskripsi nilai.
GET	*/api/deskripsi-nilais/:id*	Mendapatkan deskripsi nilai berdasarkan ID.
PUT	*/api/deskripsi-nilais/:id*	Memperbarui deskripsi nilai berdasarkan ID.
DELETE	*/api/deskripsi-nilais/:id*	Menghapus deskripsi nilai berdasarkan ID.

Tabel 3.9 Tabel *Endpoint* NilaiService modul Kategori Nilai

Metode	***Endpoint***	**Kebutuhan**
POST	*/api/kategori-nilais*	Menambah Kategori nilai baru.
GET	*/api/ kategori-nilais*	Mendapatkan daftar semua Kategori nilai.
GET	*/api/ kategori-nilais/:id*	Mendapatkan Kategori nilai berdasarkan ID.
PUT	*/api/ kategori-nilais/:id*	Memperbarui Kategori nilai berdasarkan ID.
DELETE	*/api/ kategori-nilais/:id*	Menghapus Kategori nilai berdasarkan ID.

Tabel 3.10 Tabel *Endpoint* NilaiService modul Jenis Nilai Akhir

Metode	***Endpoint***	**Kebutuhan**

POST	*/api/jenis-nilai-akhirs*	Menambah deskripsi jenis nilai akhir baru.
GET	*/api/jenis-nilai-akhirs*	Mendapatkan daftar semua jenis nilai akhir.
GET	*/api/jenis-nilai-akhirs/:id*	Mendapatkan jenis nilai akhir berdasarkan ID.
PUT	*/api/deskripsi-nilais/:id*	Memperbarui jenis nilai akhir berdasarkan ID.
DELETE	*/api/jenis-nilai-akhirs/:id*	Menghapus jenis nilai akhir berdasarkan ID.

f. Rancangan *API* NilaiSiswaService

Tabel 3.11 Tabel *Endpoint* NilaiSiswaService

Metode	***Endpoint***	**Kebutuhan**
POST	*/api/nilai-siswas*	Menambah nilai siswa baru.
GET	*/api/nilai-siswas*	Mendapatkan daftar semua nilai siswa.
GET	*/api/nilai-siswas/:id*	Mendapatkan nilai siswa berdasarkan ID.
PUT	*/api/nilai-siswas/:id*	Memperbarui nilai siswa berdasarkan ID.
DELETE	*/api/nilai-siswas/:id*	Menghapus nilai siswa berdasarkan ID.

g. Rancangan *API* PresensiService

Tabel 3.12 Tabel *Endpoint* PresensiService

Metode	***Endpoint***	**Kebutuhan**
POST	*/api/presensis*	Menambah data presensi baru.
GET	*/api/presensis*	Mendapatkan daftar semua data presensi.
GET	*/api/presensis/:id*	Mendapatkan data presensi berdasarkan ID.

PUT	*/api/presensis/:id*	Memperbarui data presensi berdasarkan ID.
DELETE	*/api/presensis/:id*	Menghapus data presensi berdasarkan ID.

h. Rancangan *API* SiswaKelasRegulerBerjalanService

Tabel 3.13 Tabel *Endpoint* SiswaKelasRegulerBerjalanService

Metode	***Endpoint***	**Kebutuhan**
POST	*/api/siswa-kelas-reguler-berjalans*	Menambah siswa ke kelas reguler berjalan baru.
GET	*/api/siswa-kelas-reguler-berjalans*	Mendapatkan daftar semua siswa dalam kelas reguler berjalan.
GET	*/api/siswa-kelas-reguler-berjalans/:id*	Mendapatkan data siswa dalam kelas reguler berjalan berdasarkan ID.
PUT	*/api/siswa-kelas-reguler-berjalans/:id*	Memperbarui data siswa dalam kelas reguler berjalan berdasarkan ID.
DELETE	*/api/siswa-kelas-reguler-berjalans/:id*	Menghapus data siswa dalam kelas reguler berjalan berdasarkan ID.

i. Rancangan *API* SiswaService

Tabel 3. 14 Tabel *Endpoint* SiswaService

Metode	***Endpoint***	**Kebutuhan**
POST	*/api/siswa*	Menambah siswa baru.
GET	*/api/siswa*	Mendapatkan daftar semua siswa.
GET	*/api/siswa/:nisn*	Mendapatkan data siswa berdasarkan NISN.
PUT	*/api/siswa/:nisn*	Memperbarui data siswa berdasarkan NISN.

DELETE	*/api/siswa/:nisn*	Menghapus data siswa berdasarkan NISN.

j. Rancangan *API* KaldikService

Tabel 3.15 Tabel *Endpoint* KaldikService

Metode	***Endpoint***	**Kebutuhan**
POST	*/api/kaldiks*	Menambah data kalender pendidikan baru.
GET	*/api/kaldiks*	Mendapatkan daftar semua kalender pendidikan.
GET	*/api/kaldik /:id*	Mendapatkan kalender pendidikan berdasarkan ID.
PUT	*/api/kaldiks/:id*	Memperbarui kalender pendidikan berdasarkan ID.
DELETE	*/api/kaldiks/:id*	Menghapus kalender pendidikan

3.5 Rancangan Pengujian

Pengujian dalam penelitian ini akan dilaksanakan menggunakan metode pengujian fungsional. Pengujian ini akan difokuskan pada pengujian respons *API* untuk memastikan bahwa setiap permintaan yang dikirim ke API menghasilkan respons yang sesuai, yaitu kode status HTTP 200 (OK), bukan kode *status* 500 (*Internal Server Error*). Kode status 200 menunjukkan bahwa *API* berfungsi dengan baik dan permintaan diproses sesuai dengan yang diharapkan. Sebaliknya, kode *status* 500 mengindikasikan adanya kesalahan pada sisi *server* yang perlu diperbaiki. Dengan pengujian ini, diharapkan semua endpoint *API* dalam sistem dapat beroperasi secara andal dan memberikan respons yang benar dalam berbagai skenario penggunaan.

BAB IV
PENGEMBANGAN DAN PENGUJIAN

4.1 Pengembangan Layanan

Pada tahap ini akan memberikan detail dan langkah-langkah yang dilakukan untuk pengembangan layanan dari desain yang telah diuraikan pada Bab 3 .

4.1.1 Inisiasi Project Laravel Layanan

Setelah berhasil membuat rancangan untuk tiap layanan, selanjutnya dilakukan proses inisiasi untuk setiap servis dengan menginstall proyek Laravel dan *library* yang akan digunakan oleh seluruh servis.

4.1.2 Melakukan Pengaturan Pada File Konfigurasi Docker

Tahapan ini akan sangat terbantu oleh *library* sail dari Laravel karna akan langsung membuat file docker-compose.yml di setiap proyek yang telah di inisiasi sebelumnya.

```
services:
    Laravel.test:
        build:
            context: ./vendor/laravel/sail/runtimes/8.3
            dockerfile: Dockerfile
            args:
                WWWGROUP: '${WWWGROUP}'
        image: sail-8.3/app
        extra_hosts:
            - 'host.docker.internal:host-gateway'
        ports:
            - '${APP_PORT:-80}:80'
            - '${VITE_PORT:-5177}:${VITE_PORT:-5173}'
        environment:
            WWWUSER: '${WWWUSER}'
            LARAVEL_SAIL: 1
            XDEBUG_MODE: '${SAIL_XDEBUG_MODE:-off}'
            XDEBUG_CONFIG:                          '${SAIL_XDEBUG_CONFIG:-
client_host=host.docker.internal}'
            IGNITION_LOCAL_SITES_PATH: '${PWD}'
        volumes:
            - '.:/var/www/html'
        networks:
            - sail
        depends_on:
            - mysql
    mysql:
        image: 'mysql/mysql-server:8.0'
        ports:
            - '${FORWARD_DB_PORT:-3310}:3306'
        environment:
            MYSQL_ROOT_PASSWORD: '${DB_PASSWORD}'
            MYSQL_ROOT_HOST: '%'
            MYSQL_DATABASE: '${DB_DATABASE}'
```

```
            MYSQL_USER: '${DB_USERNAME}'
            MYSQL_PASSWORD: '${DB_PASSWORD}'
            MYSQL_ALLOW_EMPTY_PASSWORD: 1
        volumes:
            - 'sail-mysql:/var/lib/mysql'
            -         './vendor/laravel/sail/database/mysql/create-testing-
database.sh:/docker-entrypoint-initdb.d/10-create-testing-database.sh'
        networks:
            - sail
        healthcheck:
            test:
                - CMD
                - mysqladmin
                - ping
                - '-p${DB_PASSWORD}'
            retries: 3
            timeout: 5s
networks:
    sail:
        driver: bridge
volumes:
    sail-mysql:
        driver: local
```

Gambar 4.1 Isi file docker-compose.yml.

Pada tahapan ini hanya perlu dilakukan penggantian pada nomor APP_PORT:-80 sesuaikan dengan daftar port pada tabel.

Tabel 4.1 Tabel Daftar *Port* Layanan

Nama Layanan	*PORT*
KurikulumService	80
NilaiService	81
MataPelajaranService	82
TahunAjaranService	83
KelasService	84
SiswaService	85
SiswaKelasService	86
NilaiSiswaService	87
KaldikService	88
PresensiService	89

4.1.3 Pembuatan *Database*

Pada tahapan ini akan dilakukan pembuatan database dari setiap layanan mulai dari pembuatan file migrasi dan model pada setiap layanan.

```
<?php
```

```
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class TahunAjaran extends Model
{
    use HasFactory;

    protected $fillable = [
        'tahun_ajaran',
        'semester',
        'tanggal_mulai',
        'tanggal_selesai',
    ];

    public static function rules()
    {
        return [
            'tahun_ajaran' => 'required|string|max:255',
            'semester' => 'required|in:ganjil,genap',
            'tanggal_mulai' => 'required|date',
            'tanggal_selesai'                                                    =>
'required|date|after_or_equal:tanggal_mulai',
        ];
    }
}
```

Gambar 4.2 Kode model TahunAjaran.

Dapat dilihat pada Gambar 4.2 pada bagian model hanya perlu mengisi *field* kedalam variabel "fillable" dan menentukan validasi aturan dari setiap *field* pada fungsi *rules()*. Untuk daftar *field* yang akan dibuat setiap *service* adalah sebagai berikut:

a. *Field* KurikulumService

   Pada layanan ini akan ada 3 *field* yaitu:

   1. id_tahun_ajaran
   2. nama_kurikulum
   3. file_kurikulum.

b. *Field* TahunAjaranService

   Pada layanan ini akan ada 4 *field* yaitu:

   1. tahun_ajaran
   2. semester
   3. tanggal_mulai
   4. tanggal_selesai.

c. *Field* KelasService

   Pada layanan ini terdapat 2 *model* yaitu:

   1. KelasReguler

Field pada *model* KelasReguler adalah nama_kelas, jenjang, kuota_kelas_reguler, jumlah_kelas_reguler

2. KelasRegulerBerjalan

   Field pada KelasRegulerBerjalan adalah id_kelas_reguler, id_tahun_ajaran, dan walikelas.

d. *Field* MataPelajaranService

Pada layanan ini terdapat 2 *model* yaitu:

1. NamaMapel

   Field pada *model* NamaMapel adalah nama dan warna

2. MataPelajaran

   Field pada MataPelajaran adalah id_kelas_reguler, id_tahun_ajaran, id_nama_mapel, kkm, jam_belajar.

e. *Field* NilaiService

Pada layanan ini terdapat 3 *model* yaitu:

1. DeskripsiNilai

   Field DeskripsiNilai adalah id_mata_pelajaran, nilai_atas, nilai_bawah, predikat, deskripsi_nilai

2. JenisNilai

   Field JenisNilai adalah jenis_na dan kode_na

3. KategoriNilai

   Field pada KategoriNilai adalah kategori_nilai dan bobot.

f. *Field* NilaiSiswaService

Field pada layanan ini yaitu:

1. id_siswa_kelas_reguler_berjalan
2. id_kategori_nilai
3. id_jenis_nilai
4. id_mapel
5. nilai_siswa

g. *Field* PresensiService

Field pada layanan ini yaitu:

1. id_siswa_kelas_reguler_berjalan
2. tanggal
3. status_kehadiran

4. nisn

h. *Field* SiswaKelasRegulerBerjalanService

Field pada layanan ini yaitu:

1. id_kelas_reguler_berjalan
2. nisn

i. *Field* Siswa

Field pada layanan ini yaitu:

1. no_induk_siswa
2. nisn
3. foto
4. nik
5. nama
6. jenis_kelamin
7. tempat_lahir
8. tanggal_lahir
9. agama
10. berkebutuhan_khusus
11. alamat
12. RT
13. RW
14. kategori_penduduk
15. nama_dusun
16. desa_kelurahan, kecamatan
17. kode_pos
18. tempat_tinggal
19. transportasi
20. no_telepon
21. email
22. nama_sd_mi
23. lama_belajar_disd_mi
24. pernah_paud
25. pernah_tk
26. no_skhun_mi

27. no_seri_skhun_s
28. no_seri_ijazah_sd_mi
29. penerima_kps_kks_kip
30. no_penerima
31. anak_ke
32. jumlah_saudara_kandung
33. jumlah_saudara_tiri
34. jumlah_saudara_angkat
35. status_dalam_keluarga
36. pernah_menderit_sakit
37. pernah_mengaji
38. keterangan_mengaji
39. anak_yatim_piatu
40. bahasa_seharihari_dirumah
41. prestasi_disekolah
42. status_siswa
43. tinggi_badan
44. berat_badan

j. *Field* KaldikService

Field pada layanan ini yaitu:

1. nama_kaldik
2. symbol_kaldik
3. tgl_awal
4. tgl_akhir.

4.1.4 Pembuatan *Controller*

Controller merupakan kumpulan *method* yang dapat diakses melalui *endpoint*. Setiap *endpoint* akan terhubung dengan sebuah *function*. Adapun *controller* dari setiap *service* dijelaskan pada Tabel 4.2.

Tabel 4.2 Tabel daftar detail *controller*

No	**Nama *Controller***	**Layanan**	***Function***
1	KaldikController	KaldikService	• *Index*

			• *Show* • *Update* • *Delete*
2	KelasRegulerController	KelasService	• *Index* • *Show* • *Update* • *Delete*
3	KelasRegulerBerjalanController	KelasService	• *Index* • *Show* • *Update* • *Delete*
4	KurikulumController	KurikulumService	• *Index* • *Show* • *Update* • *Delete*
5	MataPelajaranController	MataPelajaranService	• *Index* • *Show* • *Update* • *Delete*
6	NamaMapelController	MataPelajaranService	• *Index* • *Show* • *Update* • *Delete*
7	DeskripsiNilaiController	NilaiService	• *Index* • *Show* • *Update* • *Delete*
8	JenisNilaiAkhirController	NilaiService	• *Index* • *Show* • *Update* • *Delete*
9	KategoriNilaiController	NilaiService	• *Index* • *Show* • *Update*

			• *Delete*
10	NilaiSiswaController	NilaiSiswaService	• *Index* • *Show* • *Update* • *Delete*
11	PresensiController	PresensiService	• *Index* • *Show* • *Update* • *Delete* • *showByKelas* • *showByNisn*
12	SiswaKelasRegulerBerjalanController	SiswaKelasRegulerBerjalanService	• *Index* • *Show* • *Update* • *Delete* • *showByKelasBerjalan*
13	SiswaController	SiswaService	• *Index* • *Show* • *Update* • *Delete*
14	TahunAjaranController	TahunAjaranService	• *Index* • *Show* • *Update* • *Delete*

4.2 Hasil Pengembangan

Hasil pengembangan sistem informasi akademik berbasis layanan ini menghasilkan serangkaian *Application Programming Interface (API)* yang telah disusun untuk mendukung fungsionalitas utama dari setiap layanan yang ada. Setiap layanan atau *service* dalam sistem ini memiliki *API* yang dirancang untuk mempermudah interaksi antar modul serta integrasi dengan sistem lain. Daftar *API* dari setiap layanan mencakup endpoint yang memungkinkan operasi

seperti pembuatan, pembaruan, penghapusan, dan pengambilan data yang terkait dengan domain tertentu.

4.2.1 Daftar *API* TahunAjaranService

Tabel 4.3 Tabel Daftar *API* TahunAjaranService

No	***Endpoint* Name**	***Method***	***URL***	***Description***
1	*Create* Tahun Ajaran	*POST*	*http://localhost:83/api/tahun-ajarans*	Menambah tahun ajaran baru.
2	*Get All* Tahun Ajaran	*GET*	*http://localhost:83/api/tahun-ajarans*	Mendapatkan daftar semua tahun ajaran.
3	*Get* Tahun Ajaran by ID	*GET*	*http://localhost:83/api/tahun-ajarans/:id*	Mendapatkan tahun ajaran berdasarkan ID.
4	*Update* Tahun Ajaran	*PUT*	*http://localhost:83/api/tahun-ajarans/:id*	Memperbarui tahun ajaran berdasarkan ID.
5	*Delete* Tahun Ajaran	*DELETE*	*http://localhost:83/api/tahun-ajarans/:id*	Menghapus tahun ajaran

4.2.2 Daftar *API* KurikulumService

Tabel 4.4 Tabel Daftar *API* KurikulumService

No	Endpoint Name	***Method***	***URL***	***Description***
1	*Create* Kurikulum	*POST*	*http://localhost:80/api/kurikulums*	Menambah kurikulum baru.
2	*Get All* Kurikulums	*GET*	*http://localhost:80/api/kurikulums*	Mendapatkan daftar semua kurikulum.
3	*Get Kurikulum by ID*	*GET*	*http://localhost:80/api/kurikulums/:id*	Mendapatkan kurikulum berdasarkan ID.
4	*Update* Kurikulum	*PUT*	*http://localhost:80/api/kurikulums/:id*	Memperbarui kurikulum berdasarkan ID.
5	*Delete* Kurikulum	*DELETE*	*http://localhost:80/api/kurikulums/:id*	Menghapus kurikulum berdasarkan ID.

4.2.3 Daftar *API* KelasService

Tabel 4.5 Tabel Daftar *API* KelasService

No	Endpoint Name	*Method*	*URL*	*Description*
1	*Create* Kelas Reguler	*POST*	*http://localhost:84/api/kelas-regulers*	Menambah kelas reguler baru.
2	*Get All* Kelas Reguler	*GET*	*http://localhost:84/api/kelas-regulers*	Mendapatkan daftar semua kelas reguler.
3	*Get Kelas Reguler by ID*	*GET*	*http://localhost:84/api/kelas-regulers/:id*	Mendapatkan kelas reguler berdasarkan ID.
4	*Update* Kelas Reguler	*PUT*	*http://localhost:84/api/kelas-regulers/:id*	Memperbarui kelas reguler berdasarkan ID.
5	*Delete* Kelas Reguler	*DELETE*	*http://localhost:84/api/kelas-regulers/:id*	Menghapus kelas reguler berdasarkan ID.
6	*Create* Kelas Reguler Berjalan	*POST*	*http://localhost:84/api/kelas-reguler-berjalans*	Menambah kelas reguler berjalan baru.
7	*Get All* Kelas Reguler Berjalan	*GET*	*http://localhost:84/api/kelas-reguler-berjalans*	Mendapatkan daftar semua kelas reguler berjalan.
8	*Get* Kelas Reguler Berjalan by ID	*GET*	*http://localhost:84/api/kelas-reguler-berjalans /:id*	Mendapatkan kelas reguler berjalan berdasarkan ID.
9	*Update* Kelas Reguler Berjalan	*PUT*	*http://localhost:84/api/kelas-reguler-berjalans /:id*	Memperbarui kelas reguler berjalan berdasarkan ID.
10	*Delete* Kelas Reguler Berjalan	*DELETE*	*http://localhost:84/api/kelas-reguler-berjalans /:id*	Menghapus kelas reguler berjalan berdasarkan ID.

4.2.4 Daftar *API* MataPelajaranService

Tabel 4.6 Tabel Daftar *API* MataPelajaranService

No	***Endpoint Name***	***Method***	***URL***	***Description***
1	*Create* Mata Pelajaran	*POST*	*http://localhost:82/api/mata-pelajarans*	Menambah mata pelajaran baru.
2	*Get All* Mata Pelajaran	*GET*	*http://localhost:82/api/mata-pelajarans*	Mendapatkan daftar semua mata pelajaran.
3	*Get* Mata Pelajaran by ID	*GET*	*http://localhost:82/api/mata-pelajarans/:id*	Mendapatkan mata pelajaran berdasarkan ID.
4	*Update* Mata Pelajaran	*PUT*	*http://localhost:82/api/mata-pelajarans/:id*	Memperbarui mata pelajaran berdasarkan ID.
5	*Delete* Mata Pelajaran	*DELETE*	*http://localhost:82/api/mata-pelajarans/:id*	Menghapus mata pelajaran berdasarkan ID.
6	*Create* Nama Mata Pelajaran	*POST*	*http://localhost:82/api/nama-mapels*	Menambah nama mata pelajaran baru.
7	*Get All* Nama Mata Pelajaran	*GET*	*http://localhost:82/api/nama-mapels*	Mendapatkan daftar semua nama mata pelajaran.
8	*Get* Nama Mata Pelajaran by ID	*GET*	*http://localhost:82/api/nama-mapels /:id*	Mendapatkan nama mata pelajaran berdasarkan ID.
9	*Update* Nama Mata Pelajaran	*PUT*	*http://localhost:82/api/nama-mapels /:id*	Memperbarui nama mata pelajaran berdasarkan ID.
10	*Delete* Nama Mata Pelajaran	*DELETE*	*http://localhost:82/api/nama-mapels/:id*	Menghapus nama mata pelajaran berdasarkan ID.

4.2.5 Daftar *API* NilaiService

Tabel 4.7 Tabel Daftar *API* NilaiService

No	***Endpoint Name***	***Method***	***URL***	***Description***
1	*Create* Deskripsi Nilai	*POST*	*http://localhost:81/api/deskripsi-nilais*	Menambah deskripsi nilai baru.
2	*Get All* Deskripsi Nilai	*GET*	*http://localhost:81/api/deskripsi-nilais*	Mendapatkan daftar semua deskripsi nilai.
3	*Get* Deskripsi Nilai by ID	*GET*	*http://localhost:81/api/deskripsi-nilais/:id*	Mendapatkan deskripsi nilai berdasarkan ID.
4	*Update* Deskripsi Nilai	*PUT*	*http://localhost:81/api/deskripsi-nilais/:id*	Memperbarui deskripsi nilai berdasarkan ID.
5	*Delete* Deskripsi Nilai	*DELETE*	*http://localhost:81/api/deskripsi-nilais/:id*	Menghapus deskripsi nilai berdasarkan ID.
6	*Create* Kategori Nilai	*POST*	*http://localhost:81/api/kategori-nilais*	Menambah Kategori nilai baru.
7	*Get All* Kategori Nilai	*GET*	*http://localhost:81/api/kategori-nilais*	Mendapatkan daftar semua Kategori nilai.
8	*Get* Kategori Nilai by ID	*GET*	*http://localhost:81/api/kategori-nilais/:id*	Mendapatkan Kategori nilai berdasarkan ID.
9	*Update* Kategori Nilai	*PUT*	*http://localhost:81/api/kategori-nilais/:id*	Memperbarui Kategori nilai berdasarkan ID.
10	*Delete* Kategori Nilai	*DELETE*	*http://localhost:81/api/kategori-nilais/:id*	Menghapus Kategori nilai berdasarkan ID.
11	*Create* Deskripsi Nilai	*POST*	*http://localhost:81/api/jenis-nilais*	Menambah jenis nilai akhir baru.
12	*Get All* Deskripsi Nilai	*GET*	*http://localhost:81/api/jenis-nilais*	Mendapatkan daftar semua jenis nilai akhir.
13	*Get* Deskripsi Nilai by ID	*GET*	*http://localhost:81/api/jenis-nilais/:id*	Mendapatkan jenis nilai akhir berdasarkan ID.
14	Update *Deskripsi* Nilai	*PUT*	*http://localhost:81/api/jenis-nilais/:id*	Memperbarui jenis nilai akhir berdasarkan ID.
15	*Delete* Deskripsi Nilai	*DELETE*	*http://localhost:81/api/jenis-nilais/:id*	Menghapus jenis nilai akhir berdasarkan ID.

4.2.6 Daftar *API* NilaiSiswaService

Tabel 4.8 Tabel Daftar *API* NilaiSiswaService

No	Endpoint Name	Method	URL	Description
1	Create Nilai Siswa	POST	http://localhost:87/api/nilai-siswas	Menambah nilai siswa baru.
2	Get All Nilai Siswa	GET	http://localhost:87/api/nilai-siswas	Mendapatkan daftar semua nilai siswa.
3	Get Nilai Siswa by ID	GET	http://localhost:87/api/nilai-siswas/:id	Mendapatkan nilai siswa berdasarkan ID.
4	Update Nilai Siswa	PUT	http://localhost:87/api/nilai-siswas/:id	Memperbarui nilai siswa berdasarkan ID.
5	Delete Nilai Siswa	DELETE	http://localhost:87/api/nilai-siswas/:id	Menghapus nilai siswa berdasarkan ID.

4.2.7 Daftar *API* PresensiService

Tabel 4.9 Tabel Daftar *API* PresensiService

No	Endpoint Name	*Method*	*URL*	*Description*
1	*Create* Presensi	*POST*	*http://localhost:89/api/presensis*	Menambah data presensi baru.
2	*Get All* Presensi	*GET*	*http://localhost:89/api/presensis*	Mendapatkan daftar semua data presensi.
3	*Get* Presensi by ID	*GET*	*http://localhost:89/api/presensis/:id*	Mendapatkan data presensi berdasarkan ID.
4	*Update* Presensi	*PUT*	*http://localhost:89/api/presensis/:id*	Memperbarui data presensi berdasarkan ID.
5	*Delete* Presensi	*DELETE*	*http://localhost:89/api/presensis/:id*	Menghapus data presensi berdasarkan ID.
6	*Get* Presensi by NISN	*GET*	*http://localhost:89/api/presensis/nisn/:nisn*	Mengambil data presensi berdasarkan NISN
7	*Get* Presensi by Kelas	*GET*	*http://localhost:89/api/ presensis/kelas/:id_siswa_kelas_reguler_berjalan*	Mengambil Data Presensi berdasarkan Kelas Reguler Berjalan

4.2.8 Daftar *API* SiswaKelasRegulerBerjalanService

Tabel 4.10 Tabel Daftar *API* SiswaKelasRegulerBerjalanService

No	*Endpoint Name*	*Method*	*URL*	Description
1	Create Siswa Kelas Reguler Berjalan	*POST*	*http://localhost:86/api/siswa-kelas-reguler-berjalans*	Menambah siswa ke kelas reguler berjalan baru.
2	Get All Siswa Kelas Reguler Berjalan	*GET*	*http://localhost:86/api/siswa-kelas-reguler-berjalans*	Mendapatkan daftar semua siswa dalam kelas reguler berjalan.
3	Get Siswa Kelas Reguler Berjalan by ID	*GET*	*http://localhost:86/api/siswa-kelas-reguler-berjalans/:id*	Mendapatkan data kelas reguler berjalan berdasarkan ID.
4	Update Siswa Kelas Reguler Berjalan	*PUT*	*http://localhost:86/api/siswa-kelas-reguler-berjalans/:id*	Memperbarui data siswa dalam kelas reguler berjalan berdasarkan ID.
5	Delete Siswa Kelas Reguler Berjalan	*DELETE*	*http://localhost:86/api/siswa-kelas-reguler-berjalans/:id*	Menghapus data siswa dalam kelas reguler berjalan berdasarkan ID.
6	Get Siswa Kelas Reguler Berjalan by Kelas Berjalan	*GET*	*http://localhost:86/api/siswa-kelas-reguler-berjalans/ kelasberjalan/:id_kelas_reguler_berjalan*	Mengambil data siswa yang terdaftar pada kelas reguler berjalan

4.2.9 Daftar *API* SiswaService

Tabel 4.11 Tabel Daftar *API* SiswaService

No	*Endpoint Name*	*Method*	*URL*	*Description*
1	Create Siswa	*POST*	*http://localhost:85/api/siswa*	Menambah siswa baru.
2	Get All Siswa	*GET*	*http://localhost:85/api/siswa*	Mendapatkan daftar semua siswa.
3	Get Siswa by NISN	*GET*	*http://localhost:85/api/siswa/:nisn*	Mendapatkan data siswa berdasarkan NISN.
4	Update Siswa	*PUT*	*http://localhost:85/api/siswa/:nisn*	Memperbarui data siswa berdasarkan NISN.
5	Delete Siswa	*DELETE*	*http://localhost:85/api/siswa/:nisn*	Menghapus data siswa berdasarkan NISN.

4.2.10 Daftar *API* KaldikService

Tabel 4.12 Tabel Daftar *API* KaldikService

No	*Endpoint Name*	*Method*	*URL*	*Description*
1	Create Tahun Ajaran	*POST*	*http://localhost:83/api/tahun-ajarans*	Menambah tahun ajaran baru.
2	Get All Tahun Ajaran	*GET*	*http://localhost:83/api/tahun-ajarans*	Mendapatkan daftar semua tahun ajaran.
3	Get Tahun Ajaran by ID	*GET*	*http://localhost:83/api/tahun-ajarans/:id*	Mendapatkan tahun ajaran berdasarkan ID.
4	Update Tahun Ajaran	*PUT*	*http://localhost:83/api/tahun-ajarans/:id*	Memperbarui tahun ajaran berdasarkan ID.
5	Delete Tahun Ajaran	*DELETE*	*http://localhost:83/api/tahun-ajarans/:id*	Menghapus tahun ajaran

4.3 Pengujian Layanan

Pengujian dilakukan sesuai dengan rancangan pengujian yang ada pada Bab 3 dimana pengujian akan dilakukan dengan metode pengujian fungsional.

4.3.1 Hasil Skenario Pengujian

Tabel 4.13 Tabel Daftar Hasil Skenario Pengujian

Name	**Result**	**Response Code**	**Response Time (ms)**
Get All Tahun Ajaran	Pass	200	144
Create Tahun Ajaran	Pass	201	49
Get Tahun Ajaran by ID	Pass	200	18
Update Tahun Ajaran	Pass	200	26
Get All Kurikulums	Pass	200	100
Create Kurikulum	Pass	201	67
Get Kurikulum by ID	Pass	200	16
Update Kurikulum	Pass	200	35
Get All KelasRegulers	Pass	200	125
Create KelasReguler	Pass	201	35
Get Single KelasReguler	Pass	200	18

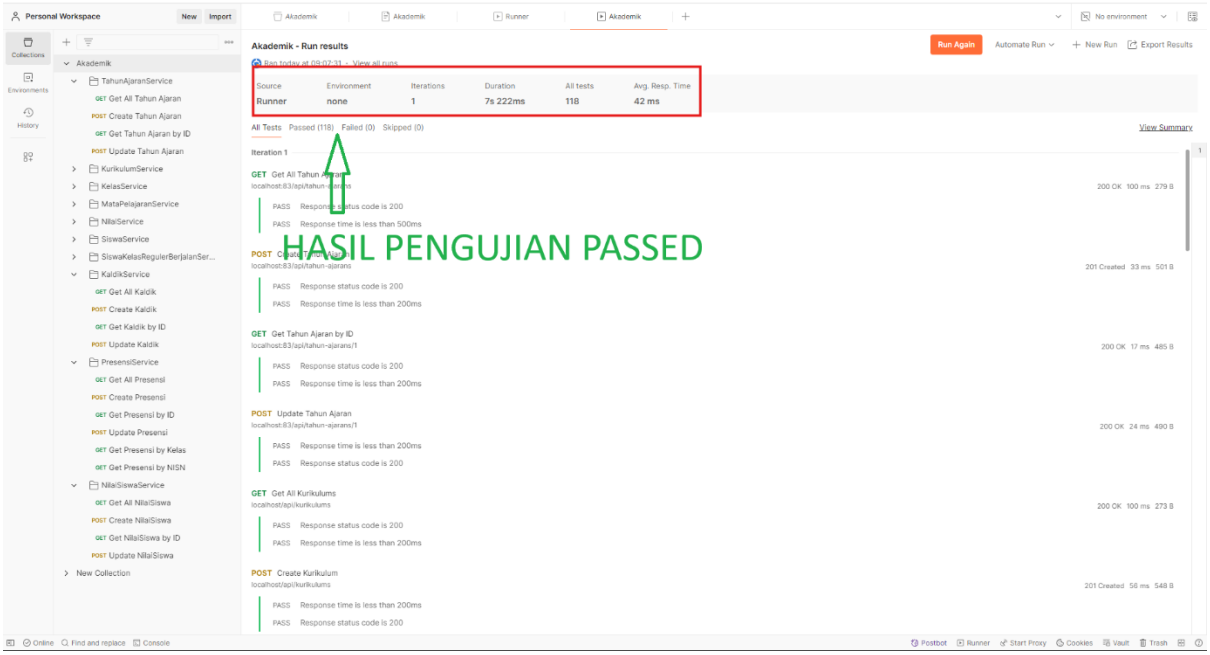
Update KelasReguler	Pass	200	16
Get All KelasRegulerBerjalans	Pass	200	15
Create KelasRegulerBerjalan	Pass	201	53
Get Single KelasRegulerBerjalan	Pass	200	19
Update KelasRegulerBerjalan	Pass	200	27
Get All NamaMapels	Pass	200	116
Create NamaMapel	Pass	201	31
Get Single NamaMapel	Pass	200	16
Update NamaMapel	Pass	200	16
Get All Mata Pelajaran	Pass	200	17
Create Mata Pelajaran	Pass	201	57

Get Single Mata Pelajaran	Pass	200	16
Update Mata Pelajaran	Pass	200	46
Get All DeskripsiNilai	Pass	200	109
Create DeskripsiNilai	Pass	201	58
Get DeskripsiNilai by ID	Pass	200	16
Update DeskripsiNilai	Pass	200	23
Get All Kategori Nilai	Pass	200	28
Create Kategori Nilai	Pass	201	23
Get Kategori Nilai by ID	Pass	200	16
Update Kategori Nilai	Pass	200	25
Get All Jenis Nilai Akhir	Pass	200	18
Create Jenis Nilai Akhir	Pass	201	21

Get Jenis Nilai Akhir by ID	Pass	200	17
Update Jenis Nilai Akhir	Pass	200	22
Get All Siswa	Pass	200	134
Create Siswa	Pass	201	41
Get Siswa by ID	Pass	200	15
Update Siswa	Pass	200	32
Get All Siswa Kelas Reguler Berjalan	Pass	200	126
Create Siswa Kelas Reguler Berjalan	Pass	201	61
Get Siswa Kelas Reguler Berjalan by ID	Pass	200	22
Update Siswa Kelas Reguler Berjalan	Pass	200	39
Get Siswa Kelas Reguler Berjalan byKelas	Pass	200	13

Get All Kaldik	Pass	200	119
Create Kaldik	Pass	201	31
Get Kaldik by ID	Pass	200	17
Update Kaldik	Pass	200	21
Get All Presensi	Pass	200	137
Create Presensi	Pass	201	67
Get Presensi by ID	Pass	200	19
Update Presensi	Pass	200	38
Get Presensi by Kelas	Pass	200	15
Get Presensi by NISN	Pass	200	17
Get All NilaiSiswa	Pass	200	134
Create NilaiSiswa	Pass	201	98
Get NilaiSiswa by ID	Pass	200	20
Update NilaiSiswa	Pass	200	78

4.3.2 Kesimpulan Pengujian



Gambar 4.3 Hasil Pengujian Api Testing

Berdasarkan hasil pengujian fungsional yang telah dilakukan pada seluruh *API* dapat dilihat kasus uji berhasil dijalankan dengan hasil "Pass". Hal ini menunjukkan bahwa setiap endpoint API yang diuji mampu merespons permintaan dengan baik dan sesuai dengan ekspektasi. Rincian hasil pengujian menunjukkan bahwa semua endpoint API memberikan kode respons yang sesuai dengan standar HTTP, seperti kode 200 (OK) untuk permintaan GET, PUT, dan PATCH yang berhasil, serta kode 201 (Created) untuk permintaan POST yang berhasil. Selain itu, waktu respons setiap permintaan berada dalam rentang yang diharapkan, yang umumnya di bawah 200ms hingga 500ms. Hal ini menunjukkan bahwa performa API cukup responsif dan efisien dalam memproses permintaan.

Keseluruhan hasil pengujian ini mencerminkan bahwa sistem API yang dikembangkan telah berfungsi sesuai dengan spesifikasi yang ditentukan dan memenuhi kriteria kualitas perangkat lunak dalam hal ketepatan respons dan efisiensi waktu respons. Dengan demikian, API ini dinilai layak digunakan untuk mendukung kebutuhan aplikasi yang memerlukan komunikasi antar sistem melalui layanan web.

BAB V
KESIMPULAN

5.1 Kesimpulan

Penelitian ini berhasil mengembangkan sistem informasi akademik Sekolah Menengah Pertama (SMP) dengan menggunakan arsitektur berorientasi layanan (SOA) dan framework Laravel. Hasil penelitian menunjukkan bahwa sistem yang dihasilkan lebih fleksibel dan mudah di-maintenance dibandingkan sistem monolitik yang digunakan sebelumnya. Pembagian sistem menjadi sepuluh layanan utama memungkinkan pengembangan, perbaikan, dan pemeliharaan dilakukan secara terpisah tanpa mempengaruhi layanan lain. Selain itu, penggunaan API testing memastikan bahwa setiap layanan berfungsi dengan baik dan dapat diintegrasikan dengan sistem lain.

5.2 Saran

Untuk pengembangan lebih lanjut, beberapa saran yang dapat diberikan antara lain:

a. Menambahkan fitur-fitur tambahan seperti manajemen keuangan dan pengelolaan kegiatan ekstrakurikuler untuk memperluas cakupan sistem.
b. Meningkatkan aspek keamanan sistem dengan menerapkan autentikasi dan otorisasi yang lebih ketat serta enkripsi data.
c. Melakukan optimasi performa sistem untuk menangani jumlah pengguna dan data yang lebih besar.
d. Menggunakan *API Gateway* atau *Message Broker* untuk komunikasi antar layanan.

DAFTAR PUSTAKA

Chen, X., Ji, Z., Fan, Y., & Zhan, Y. (2017). Restful API Architecture Based on Laravel Framework. *Journal of Physics: Conference Series*, *910*, 012016. https://doi.org/10.1088/1742-6596/910/1/012016

Giao, J., Nazarenko, A. A., Luis-Ferreira, F., Gonçalves, D., & Sarraipa, J. (2022). A Framework for Service-Oriented Architecture (SOA)-Based IoT Application Development. *Processes*, *10*(9), 1782. https://doi.org/10.3390/pr10091782

Maulidiansyah, R., Rakhman, D. F., & Ramdhani, M. A. (2017). APLIKASI PELAPORAN KERUSAKAN JALAN TOL MENGGUNAKAN LAYANAN WEB SERVICE BERBASIS ANDROID. *JURNAL ISTEK*, *10*(No 1).

SIDDIQ, M. H. (2018). *IMPLEMENTASI SISTEM INFORMASI PENILAIAN AKADEMIK SEKOLAH MENENGAH PERTAMA*. UNIVERSITAS ISLAM INDONESIA.

Warkim, & Sensuse, D. I. (2017). Model Integrasi Sistem dengan Pendekatan Metode Service Oriented Architecture dan Model View Controller pada Pusat Penelitian Perkembangan Iptek Lembaga Ilmu Pengetahuan Indonesia. *Jurnal Teknik Informatika Dan Sistem Informasi*, *3*(1). https://doi.org/10.28932/jutisi.v8i2.4411

LAMPIRAN

LAMPIRAN KODE PROGRAM dapat diakses pada link berikut https://gitlab.com/tugasakhir9113629.

LAMPIRAN DOKUMENTASI API dapat diakses pada link berikut https://documenter.getpostman.com/view/5405141/2sA3e5eTsH