PERANCANGAN DAN IMPLEMENTASI *BACKEND RESTFUL API* UNTUK MANAJEMEN *APPOINTMENT* DAN DATA LAYANAN KESEHATAN DI PT ADAMLABS

KERJA PRAKTIK

Program Studi

S1 Sistem Informasi

Oleh:

AXEL BELLIANDRI ADRIANSAH

22410100010

FAKULTAS TEKNOLOGI DAN INFORMATIKA

UNIVERSITAS DINAMIKA

2025

PERANCANGAN DAN IMPLEMENTASI *BACKEND RESTFUL API* UNTUK MANAJEMEN *APPOINTMENT* DAN DATA LAYANAN KESEHATAN DI PT ADAMLABS

Diajukan sebagai salah satu syarat untuk menyelesaikan

Program Sarjana

Disusun Oleh:

Nama : Axel Belliandri Adriansah

NIM : 22410100010

Program : S1 (Strata Satu)

Jurusan : Sistem Informasi

FAKULTAS TEKNOLOGI DAN INFORMATIKA

UNIVERSITAS DINAMIKA

2025

"We can't change what's done, we can only move on. "

- Arthur Morgan

LEMBAR PENGESAHAN

PERANCANGAN DAN IMPLEMENTASI *BACKEND RESTFUL API* UNTUK MANAJEMEN *APPOINTMENT* DAN DATA LAYANAN KESEHATAN DI PT ADAMLABS

Laporan Kerja Praktik

oleh:

Axel Belliandri Adriansah

NIM. 22410100010

Telah diperiksa, diuji, dan disetujui

Surabaya, 27 Mei 2025

Disetujui

Dosen Pembimbing	Penyelia,
Tan Amelia, S.Kom., M.MT.	**Arif Rahman Susetyo**
NIDN. 0728017602	NIP. 005-06151018

Mengetahui,

Ketua Program Studi S1 Sistem Informasi

Digitally signed by Endra Rahmawati
Date: 2025.06.27 10:48:02 +07'00'

Endra Rahmawati, M.Kom.

NIDN. 0712108701

PERNYATAAN

PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH

Sebagai mahasiswa **Universitas Dinamika**, Saya :

Nama	: **Axel Belliandri Adriansah**
NIM	: **22410100010**
Program Studi	: **S1 Sistem Informasi**
Fakultas	: **Fakultas Teknologi dan Informatika**
Jenis Karya	: **Laporan Kerja Praktik**
Judul Karya	: **PERANCANGAN DAN IMPLEMENTASI BACKEND RESTFUL API UNTUK MANAJEMEN APPOINTMENT DAN DATA LAYANAN KESEHATAN DI PT ADAMLABS**

Menyatakan dengan sesungguhnya bahwa :

1. Demi pengembangan Ilmu Pengetahuan, Teknologi dan Seni, Saya menyetujui memberikan kepada **Universitas Dinamika** Hak Bebas Royalti Non-Eksklusif (***Non-Exclusive Royalty Free Right***) atas seluruh isi/sebagian karya ilmiah Saya tersebut diatas untuk disimpan, dialihmediakan, dan dikelola dalam bentuk pangkalan data (***database***) untuk selanjutnya didistribusikan atau dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama Saya sebagai penulis atau pencipta dan sebagai pemilik Hak Cipta.
2. Karya tersebut diatas adalah hasil karya asli Saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya, atau pendapat orang lain yang ada dalam karya ilmiah ini semata-mata hanya sebagai rujukan yang dicantumkan dalam Daftar Pustaka Saya.
3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiasi pada karya ilmiah ini, maka Saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada Saya.

Demikan surat pernyataan ini Saya buat dengan sebenar-benarnya.

Surabaya, 10 Juni 2025

Axel Belliandri Adriansah
NIM : 22410100010

ABSTRAK

Banyak lembaga kesehatan telah beralih ke sistem pengelolaan data digital sebagai akibat dari kemajuan teknologi informasi. Sistem *backend* yang dapat menangani manajemen data secara sistematis merupakan kebutuhan utama. Tujuan dari penelitian ini adalah untuk membuat dan menerapkan *backend RESTful API* yang dapat digunakan untuk mengelola data *appointment*, riwayat, konten layanan (iklan dan berita), dan informasi *homepage* di sistem aplikasi PT ADAMLABS. Sistem ini menggunakan framework *Express.js*, basis data *PostgreSQL*, dan pengelola struktur data *ORM Sequelize*. Untuk memastikan bahwa setiap *endpoint* dapat merespon sesuai spesifikasi, pengujian dilakukan menggunakan *Swagger UI* dan *Postman*. Setiap modul diuji menggunakan proses *CRUD*, yang mencakup penanganan kesalahan, validasi data, dan pengunduhan riwayat dalam format PDF. Hasil perancangan ini menunjukkan bahwa *API* yang dibangun dapat menyediakan layanan data yang terstruktur dan konsisten untuk aplikasi mobile yang digunakan pasien dan tenaga medis. Dua format dokumentasi *API* tersedia: *Postman* Collection untuk kebutuhan integrasi tim pengembang dan *Swagger UI* untuk dokumentasi interaktif. Diharapkan implementasi sistem ini akan mendukung digitalisasi layanan di PT ADAMLABS dan meningkatkan efisiensi pengelolaan data layanan kesehatan.

Kata Kunci: *Express.js*, Layanan Kesehatan, Manajemen *Appointment*, *PostgreSQL*, *RESTful API,*

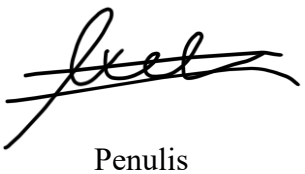
KATA PENGANTAR

Penulis berterima kasih kepada Allah SWT atas karunia-Nya, yang telah memungkinkan penulis menyelesaikan laporan kerja praktik ini dengan judul "PERANCANGAN DAN IMPLEMENTASI *BACKEND RESTFUL API* UNTUK MANAJEMEN *APPOINTMENT* DAN DATA LAYANAN KESEHATAN DI PT ADAMLABS." Laporan ini diperlukan untuk menyelesaikan program kerja praktik di Program Studi Sistem Informasi Universitas Dinamika. Penulis memperoleh banyak pengalaman yang berharga dalam pengembangan sistem *backend RESTful API*. Oleh karena itu, penulis ingin menyampaikan penghargaan yang sebesar-besarnya kepada:

1. Ayah dan Ibu tercinta yang telah memberikan dukungan penuh kepada saya.
2. Ibu Tan Amelia, S.Kom., M.MT., Selaku Dosen Pembimbing, yang telah memberikan saran, dan pengarahan dalam penyusunan laporan ini.
3. Bapak Arif Rahman Susetyo, selaku penyelia kerja praktik di PT ADAMLABS.
4. Seluruh teman-teman di Universitas Dinamika dan PT ADAMLABS.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna secara keseluruhan. Oleh karena itu, penulis terbuka untuk kritik dan saran yang bermanfaat untuk perbaikan di masa depan.

Surabaya, 10 Juni 2025



Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

BAB I

PENDAHULUAN

1.1 Latar Belakang

Di era modern, layanan kesehatan semakin bergantung pada teknologi untuk meningkatkan efisiensi dan kualitas pelayanan. Banyak lembaga kesehatan menghadapi masalah besar dengan pengelolaan data layanan kesehatan yang efektif serta sistem manajemen janji temu (*appointment*). Sistem yang tidak terdigitalisasi seringkali menyebabkan pencatatan janji temu yang tertunda, kehilangan data pasien, dan kurangnya koordinasi antara tenaga medis dan pasien (Grand View Research, 2023).

Sebagai penyedia layanan kesehatan, PT ADAMLABS menghadapi masalah dalam mengelola layanan karena proses pengelolaan masih manual dan belum terintegrasi dengan baik. Akibatnya, pencatatan janji temu seringkali tidak sesuai dengan jadwal tenaga medis, yang dapat menyebabkan kesalahan administrasi dan ketidakpuasan pasien (Santikarama, 2023). Solusi yang memungkinkan pencatatan, pengelolaan, dan pemantauan layanan kesehatan serta janji temu pasien secara otomatis dan terstruktur diperlukan untuk mengatasi masalah ini.

Pengembangan *backend RESTful API* adalah salah satu solusi yang dapat digunakan untuk memastikan sistem dapat diakses dari berbagai platform, seperti aplikasi mobile dan website. *RESTful API* memungkinkan pertukaran data yang efisien, kompatibilitas yang baik dengan sistem sebelumnya, dan proses pencatatan dan pengolahan data layanan kesehatan yang cepat (Bogner, Kotstein, & Pfaff, 2023). *Backend RESTful API* adalah bagian sistem yang berjalan di sisi *server* dan

berfungsi sebagai penghubung antara antarmuka pengguna dan basis data. *API* ini menangani permintaan aplikasi klien, mengelola data dari *database*, dan mengembalikannya dalam *JSON* yang mudah dibaca, sehingga fitur seperti jadwal layanan dan riwayat layanan dapat digunakan dengan baik dalam aplikasi.

Penelitian ini akan melihat perancangan dan implementasi *backend RESTful API* untuk sistem manajemen janji temu dan layanan kesehatan PT ADAMLABS. *API* akan diuji dengan metode black-box *testing* untuk mengetahui seberapa andal itu dan apakah platform yang digunakan dapat digunakan dengan baik. Diharapkan *API* ini dapat meningkatkan efisiensi administrasi layanan kesehatan dan memudahkan pasien serta tenaga medis dalam mengakses informasi yang mereka butuhkan secara real-time (Santikarama, 2023).

1.2 Rumusan Masalah

Beberapa masalah utama yang akan dibahas dalam kerja praktik ini, berdasarkan latar belakang yang telah dijelaskan, yaitu bagaimana membuat *backend RESTful API* yang dapat mengelola data *appointment* dan layanan kesehatan secara efisien, bagaimana memastikan bahwa *API* yang telah dikembangkan dapat diintegrasikan dengan sistem yang ada, serta bagaimana mengoptimalkan kinerja *API* agar dapat menangani permintaan data yang besar.

1.3 Batasan Masalah

Terdapat beberapa batasan yang diterapkan agar penelitian ini tetap terarah dan sesuai dengan ruang lingkup kerja praktik, yaitu:

1. Pengembangan difokuskan pada *backend RESTful API* tanpa mencakup frontend atau antarmuka pengguna.

2. *API* yang dibuat hanya mencakup manajemen *appointment*, riwayat layanan pasien, dan informasi layanan kesehatan.
3. Sistem yang dikembangkan menggunakan *Express.js* sebagai framework *backend*, serta *PostgreSQL* sebagai basis data utama.
4. Pengujian *API* dilakukan menggunakan *Postman* dan dokumentasi *API* disusun menggunakan *Swagger*.

1.4 Tujuan

Tujuan khusus dari kerja praktik ini adalah untuk mengembangkan sistem *backend RESTful API* yang dapat membantu pengelolaan layanan kesehatan PT ADAMLABS:

1. Merancang dan mengembangkan *backend RESTful API* untuk mengelola data *appointment* dan layanan kesehatan.
2. Mengintegrasikan *API* dengan sistem atau platform aplikasi mobile PT ADAMLABS.
3. Mengoptimalkan kinerja *API* untuk menangani permintaan data yang besar.
4. Memberikan dokumentasi teknis untuk membantu tim pengembang frontend menggunakan *API* yang telah dibuat.

1.5 Manfaat

1. Bagi PT ADAMLABS: Memperoleh sistem *backend* yang efisien dalam mengelola data *appointment* dan layanan kesehatan.
2. Bagi Pengguna (Pasien dan Tenaga Medis): Memudahkan akses terhadap informasi layanan kesehatan serta proses janji temu secara daring.
3. Bagi Pengembang Frontend: Memiliki *API* yang terdokumentasi dengan baik

untuk mempermudah pengembangan antarmuka aplikasi.

4. Bagi Akademisi: Menambah wawasan dan pengalaman dalam pengembangan sistem berbasis *RESTful API* menggunakan *Express.js* dan *PostgreSQL*.

Dengan adanya sistem yang dirancang dalam kerja praktik ini, diharapkan PT ADAMLABS dapat meningkatkan efisiensi operasional dalam pengelolaan data layanan kesehatan serta memberikan pengalaman yang baik bagi pasien dan tenag

BAB II

GAMBARAN UMUM

2.1 Latar Belakang Perusahaan

PT ADAMLABS adalah startup teknologi kesehatan (Health-Tech) yang didirikan pada tahun 2019 dan berbasis di Surabaya, Jawa Timur. Untuk meningkatkan efisiensi manajemen data dan layanan kesehatan, perusahaan mengembangkan layanan berbasis web yang digitalisasi sektor kesehatan, seperti Sistem Informasi Laboratorium (LIS), Sistem Informasi Klinik (CIS), dan Sistem Informasi Hospital (HIS). Lokasi kantor PT ADAMLABS adalah di Eastern Park AB, yang terletak di Jl. Raya Sukolilo Mulia No.Ruko B23, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur 60111. Logo dari PT ADAMLABS dapat dilihat pada Gambar 2.1.

Gambar 2.1 Logo PT ADAMLABS

2.2 Identitas Perusahaan

Nama Instansi	:	PT ADAMLABS
Alamat	:	Eastern Park AB, Jl. Raya Sukolilo Mulia No.Ruko B23, Keputih, Kec. Sukolilo, Surabaya, Jawa Timur 60111
No. Telepon	:	0811-3190-408
Website	:	adamlabs.id
Email	:	support.marketing@adamlabs.id

2.3 Visi Perusahaan

PT ADAMLABS berkomitmen untuk meningkatkan layanan kesehatan melalui integrasi teknologi yang baik dan berkelanjutan dengan menggunakan pendekatan Ekosistem Teknologi Kesehatan (*Health-Tech*).

2.4 Misi Perusahaan

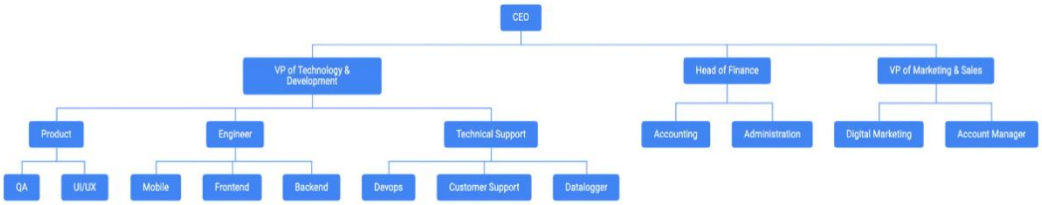
Misi dari PT ADAMLABS adalah sebagai berikut:

1. Memungkinkan Rumah Sakit, Klinik, Laboratorium, dan Radiologi menggunakan sistem informasi kesehatan yang terintegrasi.
2. Bekerja sama dengan berbagai pihak untuk meningkatkan kualitas layanan kesehatan di Indonesia.
3. Mengembangkan sistem berbasis teknologi untuk meningkatkan efisiensi operasional fasilitas kesehatan.

2.5 Struktur Organisasi

Struktur organisasi PT ADAMLABS ditunjukkan dengan posisi tertinggi dijabat oleh *Chief Executive Officer* (*CEO*). Di bawah *CEO*, terdapat tiga divisi utama yaitu *VP of Technology & Development*, *Head of Finance*, dan *VP of Marketing & Sales*. *VP of Technology & Development* membawahi tim *Product* (*UI/UX*), *Engineer* (*Mobile*, *Frontend*, *Backend*, dan *DevOps*), serta *Technical Support* (*Customer Support* dan *Datalogger*). *Head of Finance* bertanggung jawab atas divisi *Accounting* dan *Administration*. Sementara itu, *VP of Marketing & Sales* membawahi *Digital Marketing* dan *Account Manager*. Setiap divisi memiliki peran penting dalam mendukung operasional dan pengembangan perusahaan secara sinergis. Struktur ini memungkinkan koordinasi yang efisien dan responsif terhadap

kebutuhan teknologi dan pasar. Lebih jelasnya dapat dilihat pada gambar 2.2.



Gambar 2.2 Struktur Organisasi PT ADAMLABS

BAB III

LANDASAN TEORI

3.1 *Node.js*

Node.js adalah platform runtime *JavaScript* yang berjalan di sisi *server* dan memungkinkan pengembang menjalankan *JavaScript* di luar lingkungan browser. Arsitekturnya yang bersifat *asynchronous* dan berbasis *event* membuat *Node.js* efisien dalam menangani permintaan secara bersamaan. Hal ini menjadikannya pilihan yang tepat untuk pengembangan *backend API* yang cepat dan responsif, khususnya dalam sistem layanan kesehatan yang membutuhkan performa tinggi dan pengolahan data real-time (Santikarama, 2023).

Dibangun di atas *Node.js*, sistem *backend RESTful API* proyek ini berfungsi untuk menangani alur logika permintaan pengguna untuk aplikasi mobile, seperti membuat janji temu dan melihat riwayat layanan. Karena banyak modul yang dapat digunakan langsung untuk memenuhi kebutuhan sistem, ekosistem modul *NPM* memungkinkan pengembangan *API* menjadi cepat dan fleksibel (Garg & Gupta, 2023).

3.2 *Express.js*

Express.js merupakan framework minimalis berbasis *Node.js* yang digunakan untuk membangun aplikasi web dan *RESTful API*. Framework ini menawarkan fitur-fitur seperti *routing*, pengelolaan HTTP *request*, dan modularitas struktur proyek. *Express.js* membantu menyederhanakan penulisan kode dalam pengembangan *backend RESTful API* dan memungkinkan pemisahan antara logika bisnis dan konfigurasi *server* (Bogner, Kotstein, & Pfaff, 2023).

Dalam sistem yang dikembangkan di PT ADAMLABS, *Express.js* digunakan untuk menangani permintaan-permintaan dari aplikasi mobile, seperti mengatur janji temu atau mengambil data layanan. *Express.js* mempercepat pengujian dan debugging karena struktur kodenya yang sederhana dan mudah dikembangkan.

3.3 *RESTful API* dalam Integrasi Sistem

RESTful API (Representational State Transfer) adalah pendekatan arsitektural yang digunakan dalam pengembangan layanan web berbasis HTTP. *API* ini menggunakan metode standar seperti *GET*, *POST*, *PUT*, dan *DELETE* untuk mengakses dan memanipulasi data, serta menggunakan format data ringan seperti *JSON* yang mudah dibaca dan diproses oleh berbagai platform. *RESTful API* bersifat stateless, yang artinya setiap permintaan dari klien diproses secara independen, sehingga membuatnya efisien untuk diintegrasikan ke sistem aplikasi.

RESTful API berfungsi sebagai penghubung antara frontend (antarmuka pengguna) dan *backend* (*server* dan *database*). Aplikasi mobile yang dibuat oleh PT ADAMLABS dapat melakukan banyak hal, seperti membuat janji temu, mengakses data layanan kesehatan, dan menampilkan riwayat layanan secara real-time. *RESTful API* membuat integrasi sistem fleksibel, efektif, dan mudah dipelihara (Grand View Research, 2023).

3.4 *PostgreSQL*

PostgreSQL adalah sistem manajemen basis data relasional open-source yang terkenal karena kestabilannya, skalabilitasnya, dan kemampuan untuk menangani operasi yang kompleks. *PostgreSQL* mendukung standar *ACID*

(Atomitas, Konsistensi, Isolasi, Kekuatan), yang menjamin integritas dan konsistensi data, terutama di bidang penting seperti layanan kesehatan. Selain itu, sistem ini memiliki relasi antar tabel, indexing, dan dukungan untuk berbagai tipe data, membuatnya pilihan yang bagus untuk sistem berskala besar (Garg & Gupta, 2023).

PostgreSQL digunakan sebagai basis data utama dalam proyek *backend RESTful API* PT ADAMLABS untuk menyimpan data penting seperti data pasien, jadwal janji temu, histori layanan kesehatan, serta konten berita dan iklan. Dengan *PostgreSQL*, pengelolaan data menjadi terorganisir, akurat, dan mudah diakses melalui *API* yang berinteraksi dengan aplikasi mobile. Hal ini memungkinkan aplikasi menawarkan layanan terpercaya dan real-time kepada pasien dan tenaga medis.

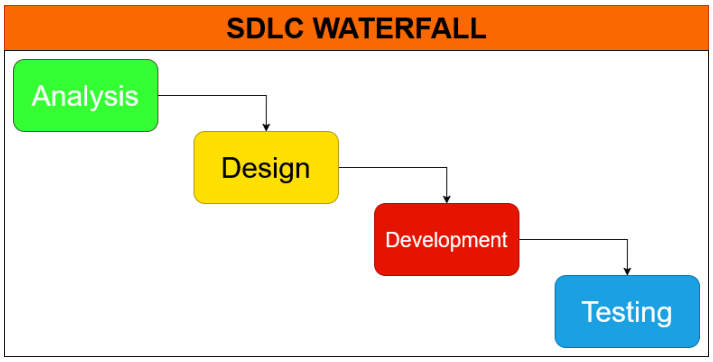
3.5 *Sequelize* (*ORM*)

Sequelize adalah Object-Relational Mapping (*ORM*) berbasis *Node.js* yang digunakan untuk menghubungkan aplikasi dengan basis data relasional seperti *PostgreSQL*. *ORM* ini memungkinkan pengembang menulis query *database* menggunakan sintaks *JavaScript* alih-alih *SQL*, yang membuat proses pengembangan cepat dan mudah.

Sequelize digunakan untuk membuat model data seperti riwayat layanan, tabel *appointment*, dan data pengguna untuk sistem yang dikembangkan di PT ADAMLABS. *Sequelize* membantu mengelola struktur data dan memungkinkan pengaturan relasi antar tabel yang terorganisir. Selain itu, *Sequelize* memungkinkan transaksi data, validasi, dan migrasi skema sesuai dengan persyaratan aplikasi layanan kesehatan (Garg & Gupta, 2023).

3.6 *SDLC Waterfall*

Mengenai *SDLC Waterfall* dapat dilihat pada Gambar 3.1



Gambar 3.1 *SDLC Waterfall*

Sumber: (Saravanos & Curinga, 2023).

Salah satu pendekatan linear dan sistematis untuk pengembangan perangkat lunak adalah *Waterfall*. Pengembangan sistem berskala besar yang membutuhkan dokumentasi lengkap dan pelacakan proses yang terstruktur dan berurutan akan menggunakan model ini. Salah satu ciri utama model ini adalah bahwa setiap tahapan harus diselesaikan terlebih dahulu sebelum melanjutkan ke tahap berikutnya (Santikarama, 2023).

Pendekatan *SDLC Waterfall* dipilih untuk proyek kerja praktik ini karena memberikan alur kerja yang terstruktur dan memudahkan pengembangan *backend RESTful API* secara bertahap dan terdokumentasi. Proses berikut digunakan:

1. *Analysis*

   Tahap ini mencakup identifikasi masalah sistem dan kebutuhan pengguna, termasuk analisis kebutuhan pengelolaan data layanan kesehatan yang fungsional dan non-fungsional.

2. *Design*

Hasil analisis diterjemahkan ke dalam rancangan teknis seperti use case, sequence diagram, dan model data (*CDM* dan *PDM*.)

3. *Development*

   Pengembangan *backend RESTful API* menggunakan *Express.js* dan *Sequelize*. Ini mencakup pengaturan, *routing*, struktur folder, dan koneksi ke *database PostgreSQL* (Bogner, Kotstein, & Pfaff, 2023).

4. *Testing*

   Pengujian dilakukan untuk memastikan setiap *endpoint* berfungsi sesuai kebutuhan. Pengujian menggunakan *Swagger UI* dengan pendekatan black-box *testing*. (Ehsan, Fatima, & Ahmed, 2022).

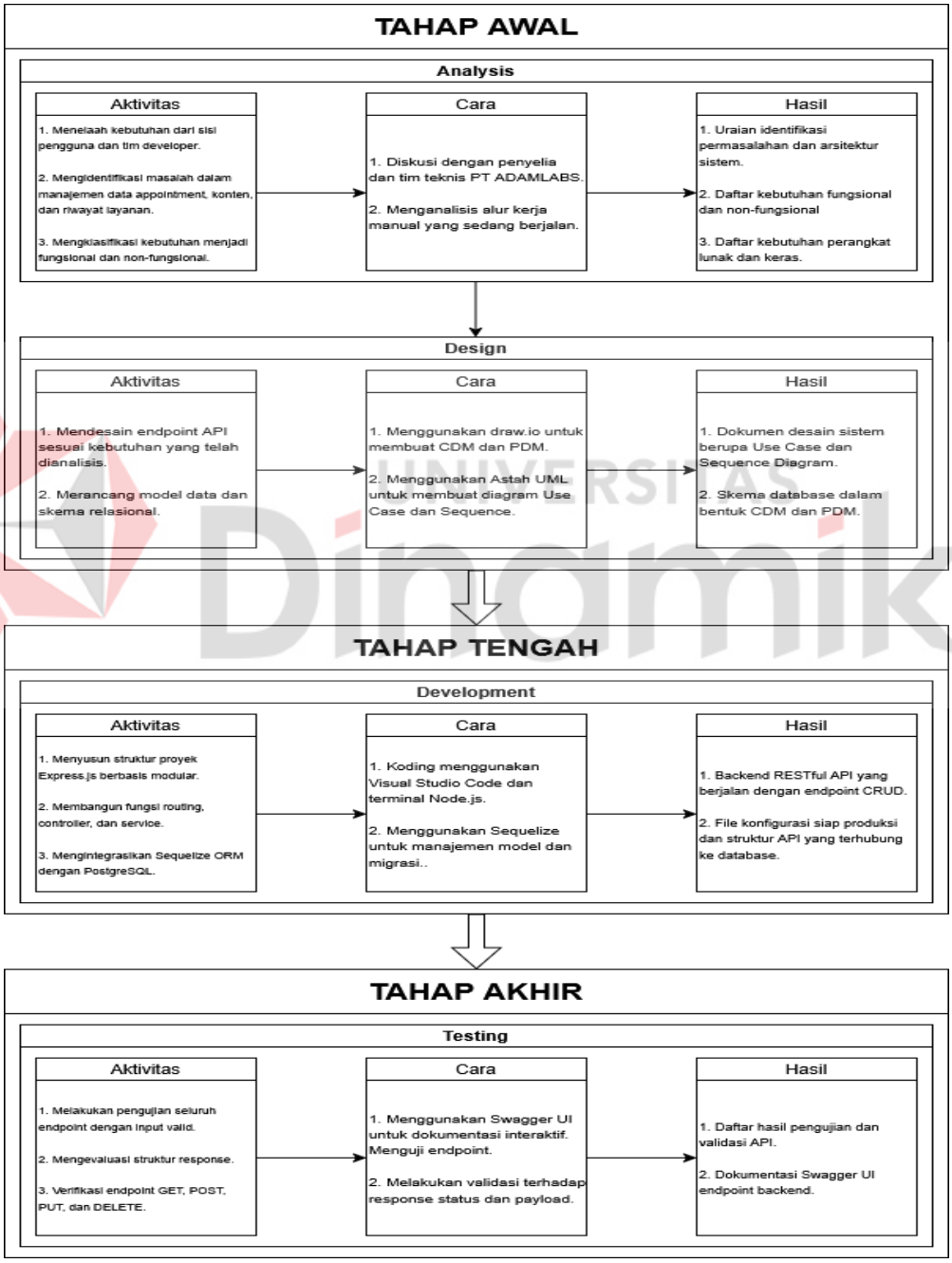
Model Waterfall cocok untuk proyek ini, yang memiliki kebutuhan sistem yang jelas sejak awal, dan dapat membantu mengembangkan sistem *backend* yang terstruktur dan terkontrol dengan baik.

BAB IV

DESKRIPSI PEKERJAAN

4.1 Metodologi Pengembangan Sistem

Untuk Metodologi lebih jelasnya dapat dilihat pada gambar 4.1.



Gambar 4.1 Metodologi Penelitian

Dalam penelitian ini, pendekatan *System Development* Life Cycle (*SDLC*)

model *Waterfall* digunakan. *Model* ini berjalan secara linier dan berurutan dari awal proyek hingga akhir, dan dipilih karena kebutuhan sistem *backend* sudah jelas sejak awal, dan proses pengembangannya memerlukan dokumentasi dan langkah-langkah sistematis.

Pada penelitian ini, *SDLC Waterfall* dibagi ke dalam tiga tahap utama berdasarkan alur waktu dan peran fungsionalnya, sehingga implementasinya lebih mudah dipahami, yaitu:

A. Tahap Awal (*Analysis* dan *Design*)

bagian dari dua tahap awal *SDLC Waterfall*, *Analysis* dan *Design*. Tahap ini berkonsentrasi pada menentukan kebutuhan dan membangun sistem sebagai dasar pengembangan *backend RESTful API*.

1. *Analysis*: Fokusnya adalah mengumpulkan kebutuhan pengguna dan masalah teknis. Kebutuhan dikategorikan menjadi fungsional dan non-fungsional.
2. *Design*: Mengubah kebutuhan yang sudah dianalisis menjadi rancangan teknis, yang mencakup diagram alur sistem, struktur *database*, dan perancangan *endpoint*.

Tujuan Tahap Awal: Memberikan fondasi konseptual dan rancangan sistem untuk pengembangan berikutnya.

B. Tahap Tengah (*Development*)

Tahap ini merepresentasikan tahapan *Development* dalam *SDLC*. Fokus pekerjaan adalah bagaimana membuat sistem *backend RESTful API* dengan desain teknis. *Sequelize ORM* dan *Express.js* digunakan untuk menjalankan kode, struktur folder, dan koneksi ke *database*.

Tujuan Tahap Tengah: Menghasilkan sistem *backend* fungsional yang siap diuji dan diintegrasikan.

C. Tahap Akhir (*Testing*)

Tahap ini mencerminkan tahapan *Testing* dalam *SDLC*:. Untuk memastikan sistem berjalan sesuai kebutuhan, *endpoint API* diuji. Dengan menggunakan *Swagger UI* sebagai alat bantu, aspek validasi, pengendalian kesalahan, dan struktur respons diuji.

Tujuan Tahap Akhir: Memastikan sistem beroperasi dengan benar dan siap digunakan oleh frontend.

4.2 *System Requirement Analysis*

Analisis kebutuhan adalah tahap penting dalam memahami masalah dan menetapkan spesifikasi sistem sebelum pengembangan sistem dimulai. Analisis ini meliputi identifikasi masalah, kebutuhan fungsional dan non-fungsional, serta kebutuhan keras dan perangkat lunak.

4.2.1 Identifikasi Permasalahan

Sebelum pengembangan *backend RESTful API*, sistem layanan kesehatan PT ADAMLABS menghadapi masalah berikut:

1. Tanpa sistem yang terpusat, pengelolaan konten, riwayat layanan, dan data *appointment* masih dilakukan secara terpisah.
2. Karena integrasi frontend dan *backend* yang buruk, sinkronisasi dan pembaruan data tertunda.
3. *API* yang tersedia saat ini tidak dapat mengelola konten berita dan iklan secara langsung di aplikasi pengguna.

Permasalahan-permasalahan ini menjadi dasar kebutuhan akan sistem *backend* yang lebih terstruktur dan terintegrasi secara menyeluruh.

4.2.2 Kebutuhan Fungsional

Untuk memenuhi proses bisnis yang telah ditentukan, sistem harus memiliki fitur yang disebut sebagai kebutuhan fungsional. Sistem *backend RESTful API* yang digunakan dalam proyek ini dimaksudkan untuk menangani pengelolaan data layanan kesehatan secara digital. Ini terutama mencakup fitur seperti *appointment*, *history*, *homepage*, dan konten. Berikut persyaratan fungsional untuk sistem ini:

A. Modul *Appointment*

1. Sistem harus dapat menambahkan data janji temu (*appointment*) baru.
2. Sistem harus dapat menampilkan daftar semua janji temu.
3. Sistem harus dapat menampilkan detail janji temu berdasarkan *UUID*.
4. Sistem harus dapat memperbarui data janji temu tertentu.
5. Sistem harus dapat menghapus data janji temu tertentu (*soft delete*).

B. Modul *History*

1. Sistem harus dapat menampilkan seluruh riwayat janji temu yang tersimpan.
2. Sistem harus dapat menghasilkan dan mengunduh laporan riwayat dalam bentuk *PDF*.

C. Modul *Homepage*

1. Sistem harus dapat mengambil data konten terbaru untuk ditampilkan pada halaman utama aplikasi mobile.

D. Modul Konten (Berita dan Iklan)

1. Sistem harus dapat menambahkan data konten baru (berita atau iklan).
2. Sistem harus dapat menampilkan seluruh data konten aktif.
3. Sistem harus dapat menampilkan konten berdasarkan *UUID* tertentu.

4. Sistem harus dapat memperbarui konten yang sudah ada.
5. Sistem harus dapat menghapus konten tertentu secara soft delete

Untuk meningkatkan efisiensi kerja dan manajemen data layanan kesehatan, setiap fungsi di atas disediakan dalam bentuk *endpoint RESTful API* yang dapat diakses oleh tim internal ADAMLABS dan frontend aplikasi mobile.

4.2.3 Kebutuhan Non-Fungsional

Meskipun kebutuhan non-fungsional sangat penting untuk menjamin kinerja, keamanan, dan keandalan sistem secara keseluruhan, mereka tidak terkait langsung dengan fungsi utama sistem. Kebutuhan non-fungsional untuk proyek *backend RESTful API* PT ADAMLABS termasuk:

A. Kinerja Sistem

1. Sistem harus dapat menanggapi permintaan *API* dengan waktu tanggap (*response* time) kurang dari 2 detik untuk setiap *endpoint*.
2. Sistem harus dapat menangani permintaan concurrency dari banyak pengguna tanpa mengurangi kinerja.

B. Keamanan Data

1. Selama masa pengembangan, sistem hanya dapat diakses melalui lingkungan tertentu, yaitu localhost atau jaringan internal.
2. *Database PostgreSQL* harus menyimpan semua data penting, seperti informasi janji temu dan histori layanan, dengan aman
3. Backup *database* dilakukan secara berkala (diatur oleh *server* atau pihak terkait ADAMLABS).

C. Skalabilitas dan Kemudahan Pemeliharaan

1. Struktur kode harus modular dan terpisah berdasarkan fungsi (*routes*,

controllers, *services*, *repositories*, dll).

2. Sistem harus mudah dikembangkan untuk fitur tambahan tanpa mengganggu fungsionalitas utama yang telah berjalan.

D. Kompatibilitas dan Aksesibilitas

1. *API* harus menggunakan format data *JSON* dan standar *REST* agar dapat diakses dari berbagai platform, seperti aplikasi mobile.
2. *Endpoint API* harus dapat diuji menggunakan alat pengujian umum seperti *Postman* atau *Swagger*.

4.2.4 Kebutuhan Perangkat Lunak dan Keras

Kebutuhan perangkat lunak dan keras mencakup spesifikasi teknis yang dibutuhkan agar sistem dapat dibangun, dijalankan, dan diuji secara optimal. Kebutuhan ini terdiri dari dua aspek, yaitu perangkat lunak (*software*) dan perangkat keras (*hardware*) yang digunakan selama pengembangan proyek *backend RESTful API*.

A. Kebutuhan Perangkat Lunak

1. Sistem Operasi: *Windows 10* atau *Linux* (*Ubuntu*)
2. Framework *Backend*: *Node.js* dengan *Express.js*
3. *Database*: *PostgreSQL*
4. *ORM*: *Sequelize*
5. *Swagger*: Untuk pengujian *endpoint API*
6. *Visual Studio Code*: Sebagai code editor utama
7. *Git* & *GitHub*: Untuk version control
8. *Terminal*/*Bash CLI*: Untuk menjalankan perintah dan migrasi *database*

B. Kebutuhan Perangkat Keras

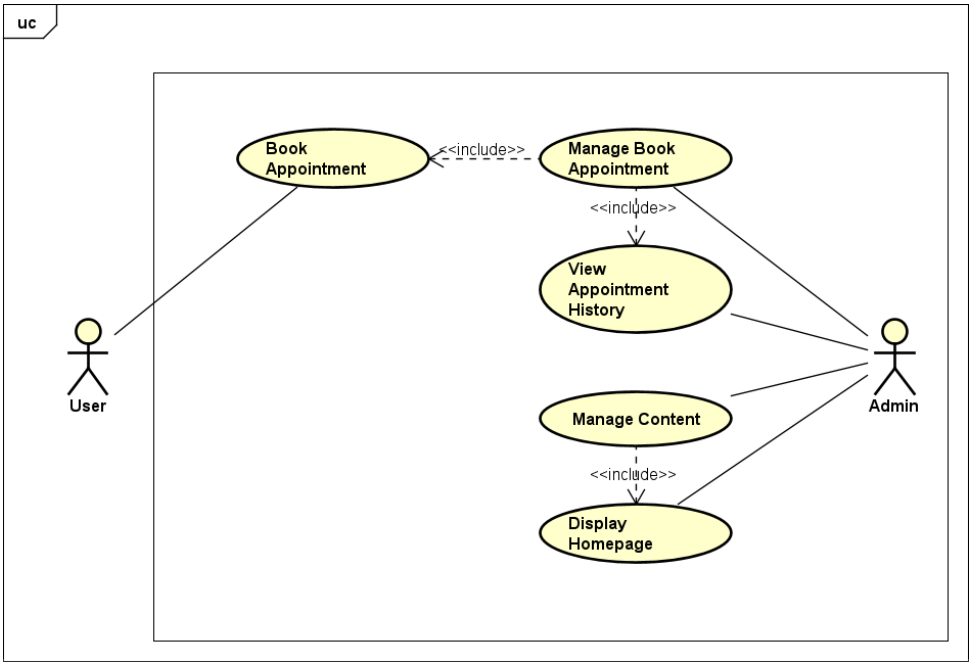
1. *Processor*: Minimal *Intel Core i5* atau setara
2. *RAM*: Minimal 8 GB
3. Penyimpanan: Minimal 256 *GB SSD*
4. Koneksi Internet: Stabil untuk keperluan integrasi, *testing*, dan dokumentasi online

4.3 Perancangan Sistem

Perancangan sistem bertujuan untuk menggambarkan alur kerja aplikasi sebelum diimplementasikan. Tahapan ini mencakup pembuatan diagram seperti *Use Case*, *Sequence*, *ERD*, dan arsitektur sistem agar pengembangan berjalan sesuai kebutuhan dan terstruktur.

4.3.1 *Use Case Diagram* Manajemen Layanan Kesehatan

Use case diagram Manajemen Layanan Kesehatan menggambarkan interaksi antara dua aktor, yaitu *User* dan *Admin*, dengan sistem layanan kesehatan. Dengan melakukan *Book Appointment*, pengguna dapat mengikuti proses Manajemen *Book Appointment*. *Administrator* dapat mengelola pemesanan, melihat riwayat janji temu, dan mengelola konten beranda. Karena beberapa use case merupakan bagian dari proses utama lainnya, hubungan antar proses ditunjukkan dengan istilah "*include*". Lebih jelasnya dapat dilihat gambar 4.2.



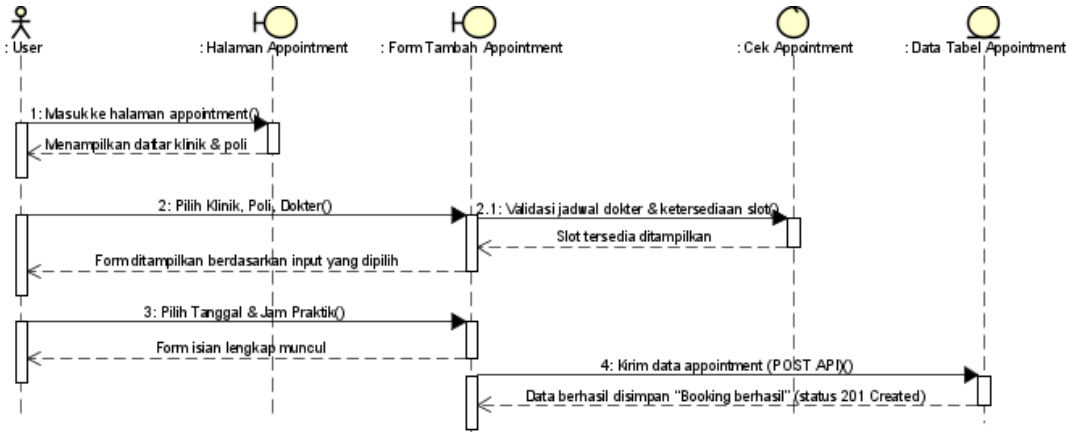
Gambar 4.2 *Use Case Diagram* Manajemen Layanan Kesehatan

4.3.2 *Sequence Diagram*

Diagram ini menjelaskan alur pemesanan janji temu oleh *user*. Pengguna mengisi form, memilih klinik dan dokter, lalu data dikirim ke *server* melalui *API* untuk disimpan ke basis data.

A. *Sequence Diagram API Appointment*

Berikut *Sequence API Appointment* jika *user* melakukan pemesanan janji temu melalui aplixkasi. Interaksi dimulai dari pengguna yang mengisi form pemesanan, kemudian data dikirim ke *server* melalui *API* untuk diproses dan disimpan dalam basis data. Lebih jelasnya dapat dilihat gambar 4.3.

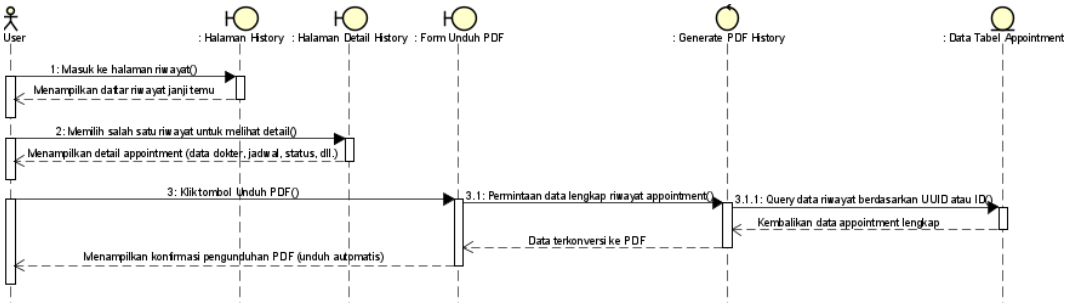


Gambar 4.3 *Sequence Diagram API Appointment*

Dalam diagram sequence *API Appointment*, digambarkan bahwa pengguna memulai dengan membuka Form Booking *Appointment*, kemudian data dikirim melalui *controller* ke *backend*. Sistem akan memverifikasi dan menyimpan data ke tabel appointments. Setelah itu, sistem mengembalikan respons berupa detail booking yang berhasil dilakukan, yang kemudian ditampilkan kembali ke pengguna.

B. *Sequence Diagram API History*

Berikut *Sequence API History* jika *user* ingin melihat riwayat janji temu yang pernah dilakukan serta mengunduhnya dalam format *PDF*. Proses ini mencakup pemanggilan riwayat dan interaksi unduhan file. Lebih jelasnya dapat dilihat gambar 4.4.

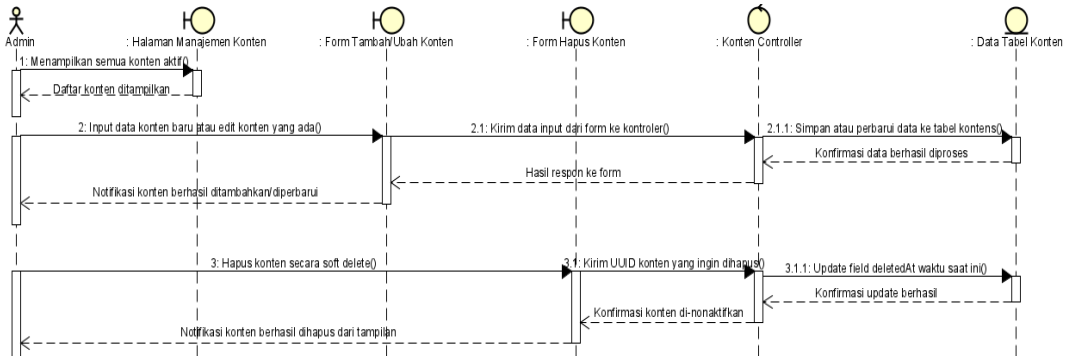


Gambar 4.4 *Sequence Diagram API History*

Dalam diagram sequence *API History*, digambarkan bahwa pengguna mengakses Halaman *History*, lalu memilih salah satu data untuk melihat detailnya di Halaman Detail *History*. Jika diinginkan, pengguna dapat menekan tombol pada Form Unduh PDF, dan sistem akan menghasilkan serta mengembalikan file PDF dari riwayat tersebut berdasarkan data di tabel appointments.

C. *Sequence Diagram API* Konten (*Admin*)

Berikut *Sequence API* Konten (*Admin*) jika *admin* ingin menambahkan atau mengelola konten seperti berita atau iklan. Proses dimulai dari halaman *admin* untuk input konten baru. Lebih jelasnya dapat dilihat gambar 4.5.



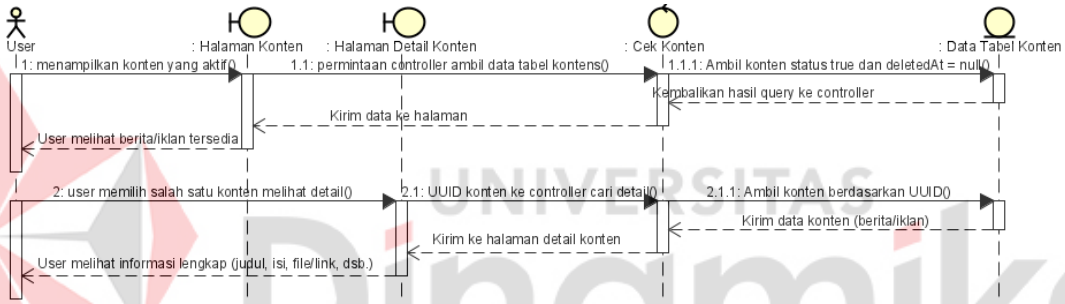
Gambar 4.5 *Sequence Diagram API* Konten (*Admin*)

Dalam diagram sequence *API* Konten (*Admin*), digambarkan bahwa *admin* mengakses Form Tambah Konten, lalu mengirim data melalui *controller* ke sistem

backend. Data konten seperti judul, jenis konten, isi, *URL*, hingga status dikirim dan divalidasi melalui kontrol logika sebelum disimpan ke tabel kontens. Setelah berhasil, sistem memberikan respons bahwa konten berhasil ditambahkan atau diperbarui.

D. *Sequence Diagram API* Konten (*User*)

Berikut *Sequence API* Konten (*User*) ketika pengguna mengakses daftar konten seperti berita atau iklan dari aplikasi mobile dalam bentuk *homepage*. Pengguna dapat memilih konten untuk melihat informasi detailnya. Lebih jelasnya dapat dilihat gambar 4.6.



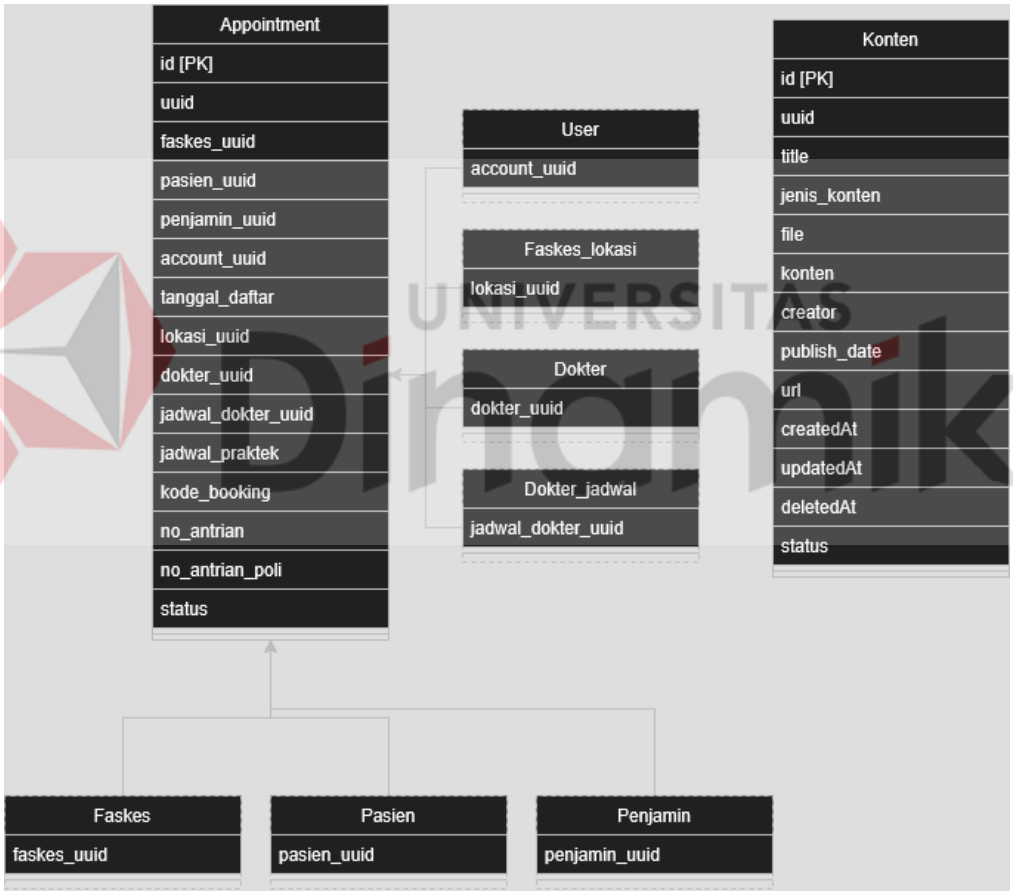
Gambar 4.6 *Sequence Diagram API* Konten (*User*)

Dalam diagram sequence *API* Konten (*User*), digambarkan bahwa pengguna pertama-tama mengakses Halaman Konten, lalu sistem mengambil data konten aktif dari tabel kontens. Setelah daftar ditampilkan, pengguna dapat memilih salah satu konten yang tersedia untuk diarahkan ke Halaman Detail Konten, dan melihat informasi lengkap sesuai *UUID* konten yang diklik.

4.3.3 *Conceptual Data Model* (*CDM*)

Dua entitas utama dalam *CDM* ini adalah janji temu, yang menyimpan data pemesanan janji temu layanan kesehatan, dan konten, yang mengelola informasi, seperti berita atau iklan yang ditunjukkan kepada pengguna. Selain itu, ada entitas

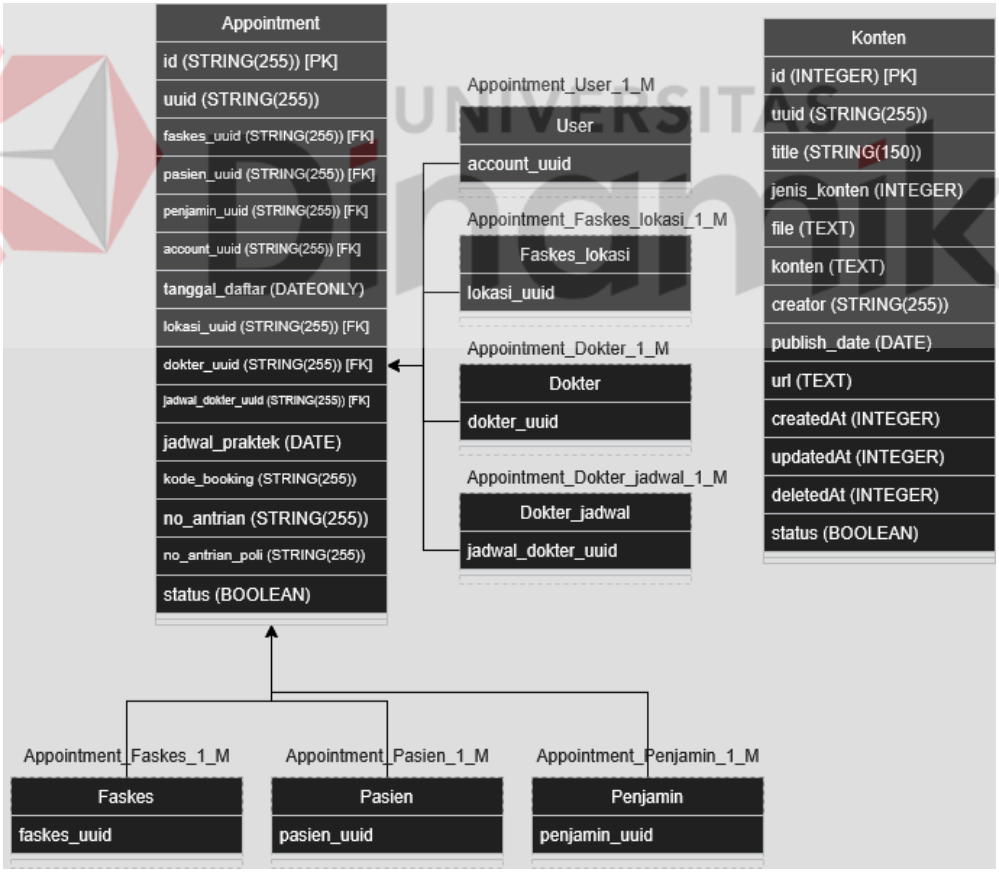
luar seperti *User*, Faskes_lokasi, Dokter, dan Dokter_jadwal yang ditunjukkan dengan garis kotak putus-putus. Entitas-entitas ini berfungsi sebagai representasi sistem atau modul lain yang tidak termasuk dalam ruang lingkup pengembangan utama. Karena model ini bertujuan untuk memberikan pemahaman umum tentang struktur data tanpa menyertakan detail teknis implementasi hubungan, tidak ada gambaran tentang hubungan antar entitas.Berikut adalah gambaran diagram *CDM* pada gambar 4.7.



Gambar 4.7 *Conceptual Data Model*

4.3.4 *Physical Data Model* (*PDM*)

PDM ini merupakan pengembangan dari *CDM* dengan menambahkan detail teknis implementasi seperti tipe data pada setiap atribut dan menggunakan kunci luar untuk menunjukkan hubungan antara entitas. Ini terlihat dalam relasi one-to-many antara entitas janji temu dengan entitas seperti Faskes, Pasien, dan Penjamin, yang menunjukkan bahwa satu data pada tabel-tabel tersebut dapat mengandung banyak data janji temu. Relasi juga diperluas dengan tabel eksternal yang mengandung *User*, Faskes_lokasi, Dokter, dan D Oleh karena itu, model ini menunjukkan secara fisik bagaimana data tersimpan dan terhubung dalam sistem. Berikut adalah gambaran diagram *PDM* pada gambar 4.8.



Gambar 4.8 *Physical Data Model*

Tabel Appointment menyimpan informasi utama terkait pemesanan janji temu layanan kesehatan, termasuk identitas fasilitas, pasien, jadwal dokter, hingga status proses janji. Lebih jelasnya dapat dilihat pada Tabel 4.1.

Tabel 4.1 *Appointment*

Field	**Tipe Data**	**Keterangan**
id	INTEGER (PK)	Primary Key
uuid	STRING	Unique ID (*UUID* v7)
faskes_*uuid*	INTEGER	ID fasilitas kesehatan
pasien_*uuid*	STRING	ID pasien
penjamin_*uuid*	STRING	ID penjamin (opsional)
account_*uuid*	STRING	ID akun pemesan
tanggal_daftar	DATEONLY	Tanggal pendaftaran
lokasi_*uuid*	STRING	Lokasi janji (opsional)
dokter_*uuid*	STRING	ID dokter (opsional)
jadwal_dokter_*uuid*	STRING	Jadwal dokter (opsional)
jadwal_praktek	DATE	Waktu praktek dokter
kode_booking	STRING	Kode booking janji temu
no_antrian	STRING	Nomor antrian
no_antrian_poli	STRING	Nomor antrian di poliklinik

Field	Tipe Data	Keterangan
status	INTEGER	Status janji temu (0 = cancel, 1 = upcoming, 2 = checkin, 3 = discharge)
createdAt	TIMESTAMP	Timestamp dibuat
updatedAt	TIMESTAMP	Timestamp terakhir diperbarui
deletedAt	TIMESTAMP	Soft delete

Selain data janji temu, sistem juga mengelola data informasi digital seperti berita dan iklan. Informasi tersebut disimpan dalam Tabel Konten, yang mencakup judul, isi, jenis konten, tanggal publikasi, dan status. Lebih jelasnya dapat dilihat pada Tabel 4.2.

Tabel 4.2 Konten

Field	Tipe Data	Keterangan
id	INTEGER (PK)	Primary Key
uuid	STRING	Unique ID (*UUID* v7)
title	STRING	Judul konten
jenis_konten	INTEGER	0 = News, 1 = Ads
meta_description	STRING	Deskripsi singkat (SEO)
file	TEXT	File pendukung (opsional)
konten	TEXT	Isi konten (wajib jika jenis = News)

Field	Tipe Data	Keterangan
creator	STRING	Nama pembuat
publish_date	DATE	Tanggal publikasi
url	TEXT	URL (wajib jika jenis = Ads)
createdAt	INTEGER	Timestamp dibuat (epoch)
updatedAt	INTEGER	Timestamp terakhir diperbarui (epoch)
deletedAt	INTEGER	Soft delete (epoch, opsional)
status	BOOLEAN	Status aktif (true) / tidak aktif (false)

4.3.5 Arsitektur Sistem

Aplikasi layanan kesehatan ini memiliki sistem arsitektur berbasis tiga lapisan, yang terdiri dari

A. *Presentation Layer* (Lapisan Presentasi)

1. Merupakan antarmuka yang berinteraksi langsung dengan pengguna (*user* interface.)
2. Berupa aplikasi web atau mobile yang mengirimkan permintaan (*request*) melalui HTTP ke *backend*.
3. Menggunakan tools seperti *Postman* atau *Swagger* untuk uji *endpoint*.

B. *Application Layer* (Lapisan Aplikasi/Logika Bisnis)

1. Bertanggung jawab dalam pemrosesan logika aplikasi dan alur data.
2. Dibangun menggunakan *Node.js* dengan *Express.js* sebagai framework *backend*.

C. *Data Layer* (Lapisan Data)

1. Menyimpan seluruh informasi penting seperti data janji temu, data konten, dan informasi pengguna.
2. Menggunakan *PostgreSQL* sebagai sistem manajemen basis data relasional.
3. Fitur paranoid dan timestamps diaktifkan untuk mendukung soft delete dan pencatatan waktu perubahan data.

4.4 Implementasi *Backend RESTful API*

Untuk mendukung layanan digital PT ADAMLABS, terutama dalam hal pengelolaan data janji temu, histori layanan, dan konten informasi, sistem *backend RESTful API* diimplementasikan. Arsitektur modularnya berbasis *Express.js* dan *Sequelize* memudahkan pengembangan dan pemeliharaan *API*.

4.4.1 Struktur Proyek *Express.js*

Untuk layanan kesehatan PT ADAMLABS, struktur *backend RESTful API* menggunakan framework *Express.js* dan *ORM Sequelize*. Arsitektur proyek ini berorientasi pada layanan dan modular, yang memudahkan pengembangan dan pemeliharaan kode. Lebih jelasnya dapat dilihat pada tabel 4.3.

Tabel 4.3 Struktur Direktori Proyek

Direktori / File	**Fungsi / Deskripsi**
src/configurations/	Menyimpan file konfigurasi seperti koneksi *database* dan *Swagger*
├─ *sequelize-instance.js*	Konfigurasi *Sequelize* untuk koneksi ke *PostgreSQL*

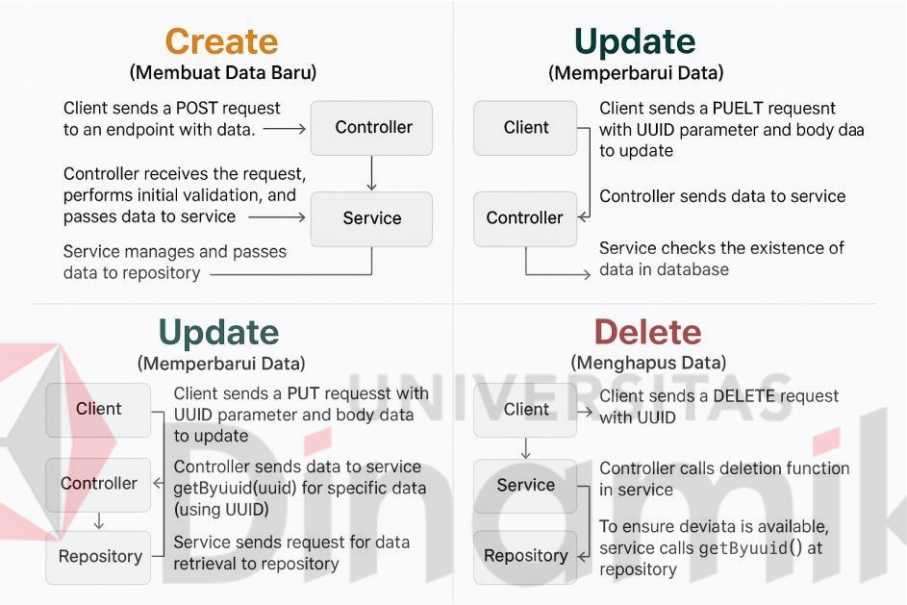
Direktori / File	Fungsi / Deskripsi
└─ *swagger.js*	Konfigurasi *Swagger* untuk dokumentasi *REST API*
src/controllers/	Menangani logika respons *API* dari request pengguna
├─ *appointment-controller.js*	*Controller* modul *appointment*
├─ *history-controller.js*	*Controller* modul riwayat
├─ *homepage-controller.js*	*Controller* modul homepage
└─ *konten-controller.js*	*Controller* modul konten
src/errors/	Menyediakan class custom untuk penanganan error
├─ *bad-request-exception.js*	*Error* untuk input tidak valid
├─ *internal-server-exception.js*	*Error* untuk kesalahan server
└─ *not-found-exception.js*	*Error* jika data tidak ditemukan
src/middlewares/	*Middleware global* atau validasi input
├─ *error-middleware.js*	Menangani *error* secara *global*
└─ *validate-schema.js*	Validasi request menggunakan schema
src/models/	Definisi model *Sequelize* untuk struktur tabel *database*
├─ *appointment*-model.*js*	Model untuk tabel *appointment*
└─ konten-model.*js*	Model untuk tabel konten

Direktori / File	**Fungsi / Deskripsi**
src/repositories/	Akses langsung ke *database* menggunakan *Sequelize ORM*
├─ *appointment-repository.js*	*Repository* untuk operasi *CRUD appointment*
└─ *konten-repository.js*	*Repository* untuk operasi *CRUD* konten
src/routes/	File *routing* yang memetakan *URL* ke *controller*
├─ *appointment-route.js*	Rute untuk *endpoint appointment*
├─ *history-route.js*	Rute untuk *endpoint* riwayat
├─ *homepage-route.js*	Rute untuk *endpoint homepage*
└─ *konten-route.js*	Rute untuk *endpoint* konten
src/services/	Menyimpan logika bisnis aplikasi
├─ *appointment-service.js*	*Service appointment*
├─ *history-service.js*	*Service* riwayat
├─ *homepage-service.js*	*Service homepage*
└─ *konten-service.js*	*Service* konten
src/validations/	Validasi skema input (menggunakan *Validator*)
└─ *konten-validation.js*	Validasi konten berdasarkan tipe konten
src/server.js	*Entry point* aplikasi *backend Express.js*

Direktori / File	Fungsi / Deskripsi
.env	File konfigurasi *environment variable* seperti *DB URL*, *PORT*, dll

4.4.2 Logika *CRUD*

Berikut gambaran Logika *CRUD* pada gambar 4.9.



Gambar 4.9 Logika *CRUD*

Setiap modul pada *backend RESTful API* menerapkan prinsip *CRUD* (*Create*, *Read*, *Update*, *Delete*) secara terstruktur dan modular. Proses *CRUD* dikendalikan melalui tiga lapisan utama: controller, service, dan repository.

A. *Create* (Membuat Data Baru)

1. Client mengirimkan permintaan *POST* ke endpoint yang terkait dengan data yang dibutuhkan.
2. *Controller* menerima permintaan, melakukan validasi awal, dan meneruskan data ke service.

3. Service mengelola dan meneruskan data ke repository.
4. Dengan menggunakan *ORM Sequelize*, repository menyimpan data ke dalam *database*.

B. *Read* (Mengambil Data)

1. *Controller* menggunakan metode *getAll*() pada *service* untuk semua data.
2. *Controller*menggunakan *getByUuid*(*uuid*) untuk data tertentu (menggunakan *UUID*).
3. Service mengirimkan permintaan untuk pengambilan data ke *repository*.
4. *Repository* menjalankan *query* dan mengembalikan hasil ke *controller*.

C. *Update* (Memperbarui Data)

1. Klien mengirimkan permintaan *PUT* yang mencantumkan parameter *UUID* dan body data yang ingin diperbarui.
2. *Controller* mengirimkan data ke *service*.
3. *Service* mengecek keberadaan data di *database*.
4. Jika data ditemukan, *repository* melakukan *update* menggunakan metode *Sequelize*.
5. Dalam tanggapan klien, perubahan dikembalikan.

D. *Delete* (Menghapus Data)

1. Klien mengirimkan permintaan *DELETE* berdasarkan *UUID*.
2. Kontroler memanggil fungsi penghapusan di *service*.
3. Untuk memastikan data tersedia, *service* memanggil *getByUuid*() di *repository*.
4. Jika ditemukan, *repository* akan menjalankan metode *destroy*() dari *Sequelize* untuk penghapusan *soft* karena menggunakan mode paranoid.

5. Data tidak dihapus secara permanen, tetapi ditandai melalui kolom *deletedAt*.

4.5 Pengujian *API*

Untuk memastikan bahwa setiap endpoint yang dibangun berfungsi sesuai dengan spesifikasi, pengujian dan validasi *API* dilakukan dengan menggunakan *Swagger UI*, yang secara otomatis menghasilkan dokumentasi interaktif dari definisi endpoint *API* dalam format OpenAPI. Dengan *Swagger*, pengembang dapat mencoba berbagai metode (*GET*, *POST*, *PUT*, dan *DELETE*) secara langsung dari browser dan melihat struktur permintaan dan tanggapan secara langsung. Setiap endpoint diuji satu per satu, dan hasilnya sesuai dengan ekspektasi. Untuk proyek *backend API* ADAMLABS, *Swagger* dikonfigurasi melalui file *swagger.js* sesuai di gambar 4.10.


```
// src/configurations/swagger.js
import swaggerJsdoc from "swagger-jsdoc";

const options = {
  definition: {
    openapi: "3.0.0",
    info: {
      title: "Backend API ADAMLABS",
      version: "1.0.0",
    },
    servers: [
      {
        url: "http://localhost:3000",
      },
    ],
  },
  apis: ["./src/routes/*.js"],
};

const swaggerSpec = swaggerJsdoc(options);

export default swaggerSpec;
```

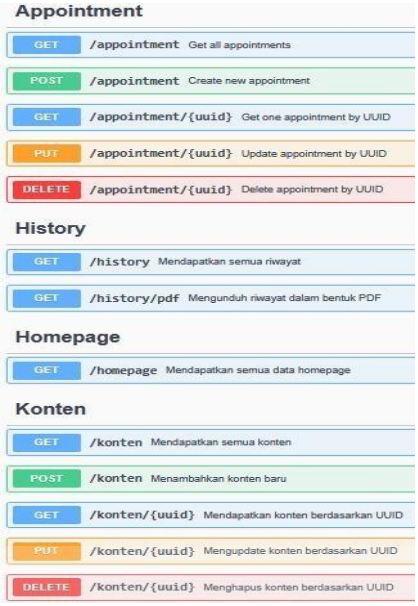
Gambar 4.10 Konfigurasi *swagger.js*

Swagger kemudian diintegrasikan ke dalam server.js menggunakan barisan kode seperi di gambar 4.11.

```
// Swagger
import swaggerUi from "swagger-ui-express";
import swaggerSpec from "./configurations/swagger.js"; // pakai export default
```

Gambar 4.11 Integrasi *swagger* ke server.js

Jika *swagger.js* berhasil berjalan, tampilan *Swagger UI* dari URL maka muncul seperti di gambar 4.12



Gambar 4.12 Tampilan *Swagger UI*

4.6 Dokumentasi *API*

Dua pendekatan utama digunakan untuk membangun dokumentasi *API* pada sistem *backend API* ADAMLABS: *Swagger UI* dan Postman Collection. Berikut adalah ringkasan endpoint utama sistem yang tersedia, serta metode HTTP dan fungsinya. Lebih jelasnya dapat dilihat pada tabel 4.4.

Tabel 4.4 *Endpoint* pada *API*

Modul	**Endpoint**	**Method**	**Deskripsi Fungsi**
Appointment	/*appointment*	*GET*	Mengambil seluruh data *appointment*
Appointment	/*appointment*	*POST*	Menambahkan data *appointment* baru
Appointment	/*appointment*/{*uuid*}	*GET*	Mengambil data *appointment* berdasarkan *UUID*
Appointment	/*appointment*/{*uuid*}	*PUT*	Memperbarui data *appointment* berdasarkan *UUID*
Appointment	/*appointment*/{*uuid*}	*DELETE*	Menghapus *appointment* berdasarkan *UUID*
Homepage	/homepage	*GET*	Mengambil data untuk ditampilkan di halaman depan aplikasi
Konten	/konten	*GET*	Mengambil seluruh data konten (berita atau iklan)
Konten	/konten	*POST*	Menambahkan data konten baru
Konten	/konten/{*uuid*}	*GET*	Mengambil data konten berdasarkan *UUID*

Modul	Endpoint	Method	Deskripsi Fungsi
Konten	/konten/{*uuid*}	*PUT*	Memperbarui data konten berdasarkan *UUID*
Konten	/konten/{*uuid*}	*DELETE*	Menghapus data konten berdasarkan *UUID*
History	/history	*GET*	Menampilkan seluruh riwayat *appointment*
History	/history/*pdf*	*GET*	Mengunduh data riwayat *appointment* dalam bentuk file *PDF*

BAB V

PENUTUP

5.1 Kesimpulan

Hasil dari perancangan dan implementasi *backend RESTful API* PT ADAMLABS untuk manajemen *appointment* dan data layanan kesehatan adalah sebagai berikut:

1. Desain dan implementasi *backend RESTful API* menggunakan teknologi *Node.js* berhasil, menggunakan framework *Express.js* dan *ORM Sequelize*, serta basis data *PostgreSQL*. Struktur proyek disusun secara modular untuk menjadikannya mudah untuk dikembangkan dan dipelihara.
2. Modul utama dari sistem *backend* yang telah dibangun termasuk jadwal, catatan, situs web, dan konten (berita dan iklan). Untuk proses *CRUD* dalam aplikasi mobile PT ADAMLABS, masing-masing modul menyediakan *endpoint API*.
3. *API* yang dibuat telah diuji dengan baik menggunakan *Swagger UI* dan *Postman*. Semua ujung berfungsi sesuai spesifikasinya. Ini termasuk fitur untuk mengambil, mengimpor, mengubah, menghapus data dengan soft delete, dan mengunduh file *PDF* riwayat jadwal.
4. Dokumentasi *API* disusun dalam dua format: *JSON* dan *Swagger* Collection. Ini memudahkan tim frontend dalam proses integrasi dan pengujian *API* secara menyeluruh.
5. Diharapkan sistem *RESTful API* ini akan meningkatkan efisiensi pengelolaan data layanan kesehatan PT ADAMLABS, mengurangi kesalahan pencatatan manual, dan memberi pengguna (pasien dan tenaga medis) akses ke data dalam

waktu nyata.

5.2 Saran

Dalam proses pengembangan dan implementasi sistem *backend RESTful API* ini, terdapat beberapa hal yang perlu diperhatikan dan dapat menjadi masukan untuk pengembangan di masa depan:

1. Sistem *backend* saat ini tidak memiliki autentikasi atau otorisasi. Untuk mendukung keamanan akses data, disarankan untuk menambahkan *middleware* autentikasi (misalnya *JWT*) untuk memastikan bahwa *endpoint* hanya dapat diakses oleh pengguna dengan hak tertentu.
2. Tampilan dokumentasi *Swagger* sudah cukup jelas. Namun, dapat ditingkatkan dengan menambahkan contoh *request*/*response* pada setiap *endpoint* dan memberikan penjelasan parameter yang lebih rinci untuk membantu tim pengembang frontend mengintegrasikan.
3. Struktur *backend* yang dibangun telah mengikuti pola modular yang baik, dengan pemisahan antara file *routes*, *controller*, dan konfigurasi *database*. Meskipun demikian, pengelolaan model perlu ditingkatkan terutama dalam penanganan field bawaan seperti createdAt dan updatedAt, agar lebih konsisten dan tidak menimbulkan konflik saat proses sinkronisasi atau migrasi data dengan *Sequelize*.
4. Untuk memenuhi permintaan dalam jumlah yang lebih besar dan lebih efisien, sistem dapat memasukkan fitur caching menggunakan *Redis* atau teknik pagination pada *endpoint* yang mengambil banyak data sekaligus untuk meningkatkan kinerja.
5. Pengembangan sistem *backend* dapat dilakukan dengan menambah modul

manajemen pengguna, notifikasi janji temu, dan integrasi dengan sistem eksternal seperti rekam medis elektronik.

DAFTAR PUSTAKA

Bogner, J., Kotstein, S., & Pfaff, T. (2023). *Do RESTful API design rules have an impact on the understandability of web APIs? Empirical Software Engineering, 28*(4), 10367. https://doi.org/10.1007/s10664-023-10367-y

Ehsan, S., Fatima, M., & Ahmed, S. (2022). *RESTful API testing methodologies: Rationale, challenges, and approaches. Applied Sciences, 12*(9), 4369. https://www.mdpi.com/2076-3417/12/9/4369

Garg, L., & Gupta, D. (2023). *Data mining and analytics in healthcare management*. Springer. https://doi.org/10.1007/978-3-031-28113-6

Grand View Research. (2023). *Healthcare information systems industry data book: Hospital information systems, pharmacy automation systems, laboratory informatics and revenue cycle management market size, share, trends analysis, and segment forecasts, 2023–2030*. https://www.grandviewresearch.com/sector-report/healthcare-information-systems-industry-data-book

Santikarama, I. (2023). *Implementing secure REST API on the integration of electronic medical records between a local hospital and nearby private clinics. AIP Conference Proceedings, 2714*(1), 020046. https://doi.org/10.1063/5.0132951

Saravanos, A., & Curinga, M. X. (2023). *Simulating the Software Development Lifecycle: The Waterfall Model*. *Applied System Innovation*, 6*(6), 108. https://doi.org/10.3390/asi6060108