ANALISIS KINERJA RESPONSE TIME RESTFULL API QUERY DATABASE POSTGRESQL, MYSQL, DAN MARIADB PADA PERANCANGAN DATABASE APLIKASI QUESTMD

(Skripsi)

Oleh

MUHAMAD SATRIO

NPM 2015061055

JURUSAN TEKNIK ELEKTRO

FAKULTAK TEKNIK

UNIVERSITAS LAMPUNG

2024

ANALISIS KINERJA RESPONSE TIME RESTFULL API QUERY DATABASE POSTGRESQL, MYSQL, DAN MARIADB PADA PERANCANGAN DATABASE APLIKASI QUESTMD

Oleh

MUHAMAD SATRIO

NPM 2015061055

Skripsi

Sebagai Salah Satu Syarat untuk Mencapai Gelar

SARJANA TEKNIK

Pada

Jurusan Teknik Elektro

Fakultas Teknik Universitas Lampung

JURUSAN TEKNIK ELEKTRO

FAKULTAK TEKNIK

UNIVERSITAS LAMPUNG

2024

ABSTRAK

ANALISIS KINERJA RESPONSE TIME RESTFULL API QUERY DATABASE POSTGRESQL, MYSQL, DAN MARIADB PADA PERANCANGAN DATABASE APLIKASI QUESTMD

Oleh

MUHAMAD SATRIO

Saat ini dalam pengembangan aplikasi memerlukan penyimpanan data yang sistematis dan cepat. Kecepatan akses informasi dalam aplikasi merupakan faktor kunci dalam pengalaman pengguna, yang dipengaruhi oleh pemilihan *database*, contohnya pada aplikasi QuestMD yang memerlukan kecepatan *response time* yang singkat untuk meningkatkan pengalaman aplikasi ketika digunakan. Aplikasi QuestMD adalah aplikasi yang dirancang untuk mengurangi tingkat stres mahasiswa kedokteran ketika belajar dengan menambahkan pengalaman belajar sambil bermain. Untuk mengatasi hal tersebut maka dilakukan perancangan *database* untuk aplikasi QuestMD. Hasil perancangan database digunakan juga sebagai *database* pengujian API menggunakan tiga *database* yaitu PostgreSQL, MariaDB, dan MySQL untuk mengetahui *response time* tercepat dari masing-masing *database* tersebut. Pada perancangan *database* penelitian menggunakan metode *Database Life Cycle* (DBLC). Sedangkan, untuk pengujian *response time* API pada *database* menggunakan perangkat lunak Apache Jmeter untuk membatu proses pengujian. Hasil penelitian ini adalah berupa rancangan *database* aplikasi QuestMD sebanyak 26 tabel untuk mendukung setiap fitur pembelajaran. Setelah diuji didapatkan bahwa *database* sudah ternormalisasi, hak akses *database* dibatasi sesuai jenis *user*, sandi *user* yang tersimpan telah terenkripsi, dan log aktivitas berjalan dengan baik. Sedangkan pengujian *response time* didapatkan bahwa *database* PostgreSQL adalah *database* tercepat pada skenario *get* (10 dan 100 data), *insert, update,* dan *delete.* Kemudian pada skenario API *get* (1000, 10.000, dan 100.000 data) *database* MySQL sebagai *database* tercepat untuk kondisi tersebut dan MariaDB yang berada di tengah dari kedua *database.* Berdasarkan hasil tersebut diketahui *database* yang cocok untuk aplikasi QuestMD adalah PostgreSQL dikarenakan aplikasi tidak menggunakan pengambilan data diatas 1000 data dan pada setiap skenario selain *get* diatas 1000 data, PostgreSQL merupakan yang paling cepat diantara ketiganya.

Kata kunci: DBLC, Apache Jmeter, QuestMD, *Response time*

ABSTRACT

PERFORMANCE ANALYSIS OF RESPONSE TIME FOR RESTFUL API QUERY DATABASES POSTGRESQL, MYSQL, AND MARIADB IN QUESTMD APPLICATION DATABASE DESIGN

By

MUHAMAD SATRIO

Currently, application development requires systematic and fast data storage. The speed of information access in an application is a key factor in the user experience, which is influenced by the choice of database. For example, in the QuestMD application which requires a short response time to improve the application experience when used. The QuestMD application is an application designed to reduce the stress level of medical students when studying by adding a learning experience while playing. To overcome this, a database was designed for the QuestMD application. The results of the database design are also used as an API testing database using three databases, namely PostgreSQL, MariaDB, and MySQL to determine the fastest response time from each of these databases. In designing the research database using the Database Life Cycle (DBLC) method. Meanwhile, for testing the API response time on the database, Apache Jmeter software is used to help with the testing process. The results of this research are a QuestMD application database design with 26 tables to support each learning feature. After testing, it was found that the database has been normalized, database access rights are limited according to user type, stored user passwords have been encrypted, and activity logs run well. . Meanwhile, response time testing showed that the PostgreSQL database was the fastest database in the get (10 and 100 data), insert, update and delete scenarios. Then, in the API get scenario (1000, 10,000, and 100,000 data) the MySQL database is the fastest database for these conditions and MariaDB in the middle of the two databases. Based on these results, it is known that the database that is suitable for the QuestMD application is PostgreSQL which does not use data retrieval above 1000 data and in every scenario other than get above 1000 data PostgreSQL is the fastest of the three.

Keyword : DBLC, Apache Jmeter, QuestMD, *Response Time*

Judul Skripsi	: **ANALISIS KINERJA RESPONSE TIME RESTFULL API QUERY DATABASE POSTGRESQL, MYSQL, DAN MARIADB PADA PERANCANGAN DATABASE APLIKASI QUESTMD**
Nama Mahasiswa	: *Muhamad Satrio*
Nomor Pokok Mahasiswa	: **2015061055**
Program Studi	: **Teknik Informatika**
Jurusan	: **Teknik Elektro**
Fakultas	: **Teknik**

MENYETUJUI

1. Komisi Pembimbing

Pembimbing Utama	Pembimbing Pendamping
Ir. Ing. Hery Dian Septama, S.T., IPM	**Ir. Trisya Septiana, ST.,MT., IPM**
NIP. 198509152008121001	NIP. 99009212019032025

2. Mengetahui

Ketua Jurusan Teknik Elektro	Program Studi Teknik Informatika
Herlinawati, S.T.,M.T.	**Yessi Mulyani, S.T., M.T**
NIP. 197103141999032001	NIP. 197312262000122001

MENGESAHKAN

1. Tim Penguji

   Ketua : **Ir. Ing. Hery Dian Septama, S.T., IPM**

   Sekretaris : **Ir. Trisya Septiana, ST.,MT., IPM**

   Penguji : **Wahyu Eko S., S.T., M.Sc.**

2. Dekan Fakultas Teknik :

Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc.
NIP. 19750928 200112 1 002

Tanggal Lulus Ujian Skripsi : **23 April 2024**

SURAT PERNYATAAN

Saya yang bertanda tangan di bawah ini, menyatakan bahwa skripsi saya dengan judul "Analisis Kinerja *Response Time Restfull* API *Query Database* PostgreSQL, MySQL, Dan MariaDB Pada Perancangan *Database* Aplikasi QuestMD" dibuat oleh saya sendiri. Semua hasil yang tertuang dalam skripsi ini telah mengikuti kaidah penulisan karya ilmiah Universitas Lampung. Apabila di kemudian hari terbukti bahwa skripsi ini merupakan salinan atau dibuat oleh orang lain, maka saya bersedia menerima sanksi sesuai dengan ketentuan hukum atau akademik yang berlaku.

Bandar Lampung, 24 April 2024

Pembuat Pernyataan,

Muhamad Satrio

NPM. 2015061055

RIWAYAT HIDUP

Penulis bernama Muhamad Satrio yang dilahirkan di Sukamarga, tanggal 08 Mei 2002. Penulis merupakan anak ketiga dari pasangan Bapak Ciktam, seorang ayah terbaik yang pernah ada, dan Ibu Wasila seorang ibu yang selalu menyayangi dan mendukung setiap jalan yang penulis pilih. Penulis menyelesaikan pendidikannya di SD Negeri 1 Sukamarga pada tahun 2012, SMP Negeri 1 Bukit Kemuning pada tahun 2017, dan SMA Negeri 1 Bukit Kemuning pada tahun 2020. Pada tahun 2020 penulis terdaftar sebagai mahasiswa Program Studi Teknik Informatika, Jurusan Teknik Elektro, Fakultas Teknik Universitas Lampung melalui jalur SBMPTN (Seleksi Bersama Masuk Perguruan Tinggi Negeri). Selama menjadi mahasiswa, penulis melakukan beberapa kegiatan, antara lain:

1. Mengikuti program Kredensial Mikro Mahasiswa Indonesia (KMMI) dari Kementerian Pendidikan dan Kebudayaan dengan mengambil kursus Teknologi Multimedia pada Agustus 2021.
2. Mengikuti kegiatan Magang Kampus Merdeka dari Kementerian Pendidikan dan Kebudayaan pada perusahaan PT Bisa Artificial Indonesia pada tahun 2022.
3. Mengikuti kegiatan Magang Kampus Merdeka dari Kementerian Pendidikan dan Kebudayaan pada perusahaan PT Paragon Technology and Innovation pada tahun 2023.
4. Melaksanakan Kuliah Kerja Nyata pada bulan Juni sampai dengan Agustus 2023 di Desa Sri Kuncoro, Kecamatan Semaka, Kabupaten Tanggamus, Provinsi Lampung.
5. Menjadi anggota pengurus Bina Rohani Islam Mahasiswa Universitas Lampung, Departemen Media dan Branding pada tahun 2022.

6. Menjadi anggota pengurus Himpunan Mahasiswa Teknik Elektro Universitas Lampung, Divisi Media Informasi pada tahun 2022.
7. Mengikuti Penelitian Universitas Lampung “Inovasi Pembelajaran Pendidikan Profesi Berbasis Sistem Informasi Terintegrasi Pada Rumah Sakit Pendidikan: Studi Kasus RSUAM” pada tahun 2023.

MOTTO

“Kesalahan dan ketidaktahuan jangan dihindari, namun dihadapi. Hal inilah yang akan membuatmu sukses menapaki duniawi”

Penulis

“Karena sesungguhnya sesudah kesulitan itu ada kemudahan”

(Q.S. Al-Insyirah : 5)

SANWACANA

Puji syukur kehadirat Allah SWT yang telah memberikan rahmat dan hidayat-Nya sehingga penulis dapat menyelesaikan penelitian yang berjudul "Analisis Kinerja Response Time Restfull Api Query Database PostgreSQL, MySQK, Dan MariaD Pada Perancangan Database Aplikasi QuestMD". Selama pelaksanaan penelitian ini penulis menerima banyak dukungan, bimbingan serta bantuan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Kedua orang tua tercinta dan seluruh keluarga yang selalu memberikan doa, motivasi dan kasih sayang tiada terkira yang selalu mengingatkan penulis untuk menyelesaikan penelitian ini;
2. Bapak Dr. Eng. Helmy Fitriawan, S.T., M.Sc., selaku Dekan Fakultas Teknik Universitas Lampung;
3. Ibu Herlinawati, S.T., M.T. selaku Ketua Jurusan Teknik Elektro Universitas Lampung;
4. Ibu Yessi Mulyani, S.T., M.T. selaku Ketua Program Studi Teknik Informatika Universitas Lampung dan telah membantu proses kelancaran pengerjaan penelitian.;
5. Bapak Ir. Ing. Hery Dian Septama, S.T., IPM selaku Pembimbing Utama yang selalu memberikan dukungan berupa waktu maupun ilmu kepada penulis dalam menyelesaikan penelitian ini
6. Ibu Ir. Trisya Septiana, S.T., M.T., IPM., selaku Pembimbing Pendamping yang telah membimbing dan memberikan dukungan dalam menyelesaikan penelitian ini;
7. Bapak Wahyu Eko S., S.T., M.Sc. selaku Penguji yang telah memberikan banyak saran dan masukan terhadap penelitian ini;

8. Bapak Ir. Gigih Forda Nama, S.T., M.T.I. selaku Pembimbing Akademik yang selalu memberikan dukungan serta bimbingan agar menjadi lebih baik;
9. Mbak Rika selaku Admin Program Studi Teknik Informatika yang telah banyak membantu penulis dalam segala urusan administrasi selama perkuliahan dan penelitian;
10. Seluruh dosen dan staf Jurusan Teknik Informatika Unila yang memberi masukan dan mempermudah proses penelitian ini.
11. Charles Gunawan, Muhammad Wafa AlAusath, Sabri, Hesti, dan segenap teman teman Tim Penelitian pembuatan aplikasi QuestMD yang telah membantu penulis selama pembuatan aplikasi.
12. Dwiki Armanto Setiawan, Rendy, Affan Maulana, Aldo Wijaya dan seluruh teman-teman PSTI Angkatan 2020 telah mewarnai masa perkuliahan penulis dan menulis banyak cerita bersama.
13. Keluarga besar Teknik Elektro Angkatan 2020 yang telah menjadi teman seperjuangan sejak mahasiswa baru. Terima kasih telah mewarnai masa perkuliahan penulis dan menulis banyak cerita bersama;

Penulis berharap agar laporan ini dapat menjadi referensi bagi pengembangan keilmuan di bidang teknik informatika. Oleh karena itu, semoga penelitian ini bermanfaat bagi yang membacanya.

Bandar Lampung, 25 April 2024
Penulis,

Muhamad Satrio

DAFTAR ISI

Halaman

DAFTAR ISI ..... **i**

DAFTAR GAMBAR ..... **iv**

DAFTAR TABEL ..... **vi**

I. PENDAHULUAN **1**

1.1 Latar Belakang 1

1.2 Perumusan Masalah 3

1.3 Tujuan Penelitian 4

1.4 Manfaat Penelitian 4

1.5 Batasan Masalah 4

1.6 Sistematika Penulisan 5

II. TINJAUAN PUSTAKA **6**

2.1 RESTful API 6

2.2 Node JS 7

2.3 Express JS 7

2.4 *Object Relational Mapping* (ORM) 8

2.4.1 Sequelize 8

2.5 *Relational Database Management System* (RDBMS) 9

2.5.1 MySQL 9

2.5.2 MariaDB 9

2.5.3 PostgreSQL 9

2.6 Apache Jmeter 10

2.7 Normalisasi *Database* 10

2.8 *Database Life Scycle* (DBLC) 11

2.9 *Entity Relation Diagram* (ERD) 14

2.10 Penelitian Terkait 15

2.10.1 Perbandingan Kinerja Cassandra dan MongoDB Sebagai *Backend* IoT *Data Storage* 15

2.10.2 *Database Systems Performance Evaluation For IOT Applications* ... 16

2.10.3 Studi Perbandingan Performansi Antara MySQL Dan PostgreSQL... 16

2.10.4 Analisis Perbandingan Kinerja *Query Database Management System* (DBMS) Antara MySQL 5.7.16 dan MariaDB 10.1 16

2.10.5 Experiments of Search Query Performance for SQL-Based Open Source Databases 17

2.10.6 Perbandingan Performa SQL dan NoSQL Dengan PHP pada 5 Juta Data 17

2.10.7 Analisis Perbandingan Performa Database DuckDB Dan SQLite Pada Pengolahan Big Data 18

2.10.8 Pengembangan Database E-commerce De Janggelan Menggunakan Metode Database Life Cycle 18

2.10.9 Desain Model Database Layanan Panti Werdha dengan Menerapkan Metode *Database Life Cycle* 19

2.10.10 Boarding House Search Information System Database Design 19

2.10.11 Perancangan Aplikasi Pembelajaran Dengan Konsep Gamifikasi.... 20

2.10.12 Gamifikasi Untuk Pembelajaran 20

III. Metodologi penelitian 21

3.1 Waktu dan Tempat Penelitian 21

3.2 Alat Penelitian 22
3.3 Tahapan Penelitian..... 23
3.3.1 *Database Initial Study* 24
3.3.2 *Database Design*..... 27
3.3.3 *Implementation and Loading*..... 36
3.3.4 *Testing and Evaluation* 37
3.3.5 *Operation*..... 37
3.3.6 *Maintenance and Evolusion*..... 38
3.3.7 Analisis Kinerja..... 38
IV. Hasil dan pembahasan..... 40
4.1 Implementation and Loading..... 40
4.2 Testing and Evaluation 50
4.3 Operation..... 53
4.4 Maintenance and Evolusion 59
4.5 Analisis Kinerja *Database*..... 60
V. Kesimpulan dan saran..... 80
5.1 Kesimpulan..... 80
5.2 Saran..... 81
DAFTAR PUSTAKA..... 82

DAFTAR GAMBAR

Gambar 2. 1 Tahapan Perancangan DBLC 12
Gambar 3. 1 Tahapan Penelitian 23
Gambar 3. 2 ERD Aplikasi QuestMD 34
Gambar 3. 3 Tabel *user, leaderboard_value, leaderboard, my_achievement*, dan *achievement* 35
Gambar 3. 4 Tabel *my_avatar, avatar, stase, bab, challen*ge 35
Gambar 3. 5 Tabel *discussion, comment, historie, jawaban_penanganan,* dan *penanganan* 36
Gambar 3. 6 Tabel jawaban_penyakit, penyakit, kunci_penyakit, dan investigasi36
Gambar 3. 7 Pembagian Peran Pembuatan Aplikasi 37
Gambar 4. 1 Daftar Tabel *Database* Aplikasi QuestMD 49
Gambar 4. 2 Folder *seeders backend* 50
Gambar 4. 3 Pengecekan normalisasi tabel *avatar* 50
Gambar 4. 4 *permission denied* user tidak diijinkan 52
Gambar 4. 5 *Password* Enkripsi Pada *Database* 53
Gambar 4. 6 Log Aktivitas *Database* 53
Gambar 4. 7 ERD *Database* QuestMD Final 58
Gambar 4. 8 Proses Backup *Database* 59
Gambar 4. 9 *Source Code API Register User* 61
Gambar 4. 10 *Source Code API Get All User* 62
Gambar 4. 11 *Source Code API Update User* 64
Gambar 4. 12 *Source Code API Delete User* 65
Gambar 4. 13 Kode *Migration User* 66
Gambar 4. 14 Tabel *user database* MariaDB dan MySQL 67
Gambar 4. 15 Tabel *user database* PostgreSQL 67

Gambar 4. 16 *Source Code* Data Dummy Python 68
Gambar 4. 17 Grafik Response Time API Get Kolom Id 69
Gambar 4. 18 Grafik Response Time API Get Kolom Exp, Total_coins, dan otp 70
Gambar 4. 19 Grafik Response Time API Get All User 10 data..... 72
Gambar 4. 20 Grafik Response Time API Get All User 100 data..... 73
Gambar 4. 21 Grafik Response Time API Get All User 1000 data..... 73
Gambar 4. 22 Grafik Response Time API Get All User 10.000 data..... 74
Gambar 4. 23 Grafik Response Time API Get All User 100.000 data..... 75
Gambar 4. 24 Grafik Response Time API Register User 75
Gambar 4. 25 Grafik Response Time API Update User..... 76
Gambar 4. 26 Grafik Response Time API Delete User..... 77

DAFTAR TABEL

Tabel 3. 1 Alur Penelitian 21
Tabel 3. 2 Alat Penelitian 22
Tabel 3. 3 Aktivitas Pengguna 26
Tabel 3. 4 Identifikasi Tipe Entitas 28
Tabel 3. 5 Identifikasi Tipe Relational 29
Tabel 3. 6 Identifikasi Atribut dan Kandidat *Key* 31
Tabel 3. 7 Pembagian Tugas Pembuatan Aplikasi 38
Tabel 4. 1 Hasil Pengecekan Normalisasi 51
Tabel 4. 2 *Connection Database* 66

I. PENDAHULUAN

1.1 Latar Belakang

Pada saat ini menyimpan data adalah fungsi yang sangat krusial bagi para pengembang, untuk menyimpan data informasi secara sistematis di dalam komputer maka dibutuhkannya sebuah alat yaitu yang bernama *database*. [1]. Pada praktiknya *database* dalam pembuatan sebuah aplikasi biasanya digunakan untuk menyimpan setiap data yang di dalamnya terdapat seperti gambar, suara, video, dan masih banyak lainnya sesuai kebutuhan pengembang. Dalam menggunakan *database* perlu sebuah *software* agar dapat mengakses dan mengubah isi *database*, jenis *software* ini disebut dengan DBMS (*Database Management System*).

Pada saat ini jenis DBMS (*Database Management System*) yang paling banyak digunakan untuk membuat aplikasi oleh para pengembang yaitu SQL. SQL (*Structured Query Language*) adalah suatu bahasa yang terstruktur yang di dalamnya terdapat aturan-aturan yang telah distandarkan oleh sebuah asosiasi yang bernama ANSI (*America National Stadards Institute*) [2]. Kelebihan SQL sebagai DBMS yaitu *database* terstruktur, dapat melakukan *join* dan *grouping/agregation*, dan dapat menjalankan *query* yang kompleks dengan cepat [3]. Berdasarkan kelebihan inilah SQL sangat banyak digunakan untuk pembuatan aplikasi terutama aplikasi yang memiliki skala yang besar atau kompleks, salah satu contohnya yaitu aplikasi pembelajaran. Aplikasi pembelajaran memiliki skalabilitas yang cukup kompleks sehingga sangat cocok untuk menggunakan *database* SQL.

Pentingnya kecepatan saat mengakses informasi merupakan salah satu aspek yang penting dalam pengalaman menggunakan aplikasi, jika *response time* aplikasi buruk maka pengguna tidak akan nyaman dan susah mengakses informasi yang

diinginkan. Kecepatan aplikasi ini salah satunya dipengaruhi oleh pemilihan *database* yang digunakan, contoh dari *open source database* yang paling banyak digunakan yaitu PostgreSQL, MySQL, dan MariaDB yang ketiganya merupakan *database* dengan jenis SQL. Selain dipengaruhi oleh database yang dipakai kecepatan dari aplikasi dipengaruhi oleh bahasa ataupun teknologi yang dipakai di API yang digunakan oleh aplikasi. Penelitian ini menggunakan *runtime environment* Node JS dengan bahasa Javascript. Node JS dipilih dengan alasan adalah bahasa yang cukup cepat untuk digunakan dalam bidang *backend* dibandingkan dengan Laravel dari bahasa PHP dan Django dari bahasa Python [4].

Pemilihan *database* dengan jenis SQL pada saat menggunakan Node JS untuk membuat sebuah API membuat bingung pengembang. Penelitian ini membandingkan tiga buah *database* yaitu PostgreSQL, MySQL, dan MariaDB apakah lebih baik menggunakan PostgreSQL, MySQL, atau MariaDB dalam hal kecepatan *response time*, seperti saat CRUD (*Create, Read, Update, Delete*), dan kondisi lainnya seperti gabungan dari fungsi CRUD itu sendiri. Kecepatan aplikasi yang baik dari aplikasi yang diakses pengguna maka meningkatkan kepuasan pelanggan saat menggunakan aplikasi [5].

Kecepatan *response time* API dapat diketahui dengan cara dilakukan pengujian di masing-masing API. Namun, sebelum pengujian diperlukan rancangan *database* aplikasi QuestMD yang merupakan studi kasus dalam penelitian. Aplikasi QuestMD adalah aplikasi untuk membantu mahasiswa kodokteran memahami pembelajaran dokter sesuai dengan kurikulum yang mereka pelajari. Sesuai dengan temanya yaitu *gamifikasi* nantinya mahasiswa tidak hanya belajar seperti biasanya. Namun, mahasiswa diberikan pembelajaran sekaligus bermain yang di dalam aplikasi tersebut terdapat permainan yang biasa dimainkan di *smartphone. Gamifikasi* sendiri sudah banyak diterapkan oleh banyak media pembelajaran yang sudah terbukti efektif untuk meningkatkan keinginan dan tidak bosan ketika belajar.

Aplikasi QuestMD dikembangkan sebagai respons terhadap tingginya tingkat stres yang dialami oleh mahasiswa dalam proses pembelajaran, serta faktor-faktor eksternal lainnya yang memengaruhi kesejahteraan mereka. Melalui pemanfaatan QuestMD, diharapkan bahwa tingkat stres mahasiswa dapat diredakan, sementara

motivasi mereka untuk terus belajar akan ditingkatkan. Fitur-fitur yang disediakan oleh QuestMD, seperti sistem peringkat antar mahasiswa, dirancang untuk merangsang semangat kompetisi sehat di antara mereka, memacu mereka untuk mencapai prestasi akademis yang lebih tinggi serta fitur lainnya yang menunjang perkembangan belajar mahasiswa.

Berdasarkan masalah yang telah diuraikan penelitian ini dilakukan untuk melihat perbandingan *response time* dari *database* berjenis SQL yaitu PostgreSQL MySQL, dan MariaDB pada Restfull API menggunakan Node JS. *Response time* diukur berdasarkan dari masing-masing *database* dan dihitung dari beberapa kondisi sesuai dengan kebutuhan aplikasi yang dibuat, yaitu aplikasi pembelajaran dokter dengan tema *gamifikasi*. Penelitian ini diharapkan dapat membantu pengembang aplikasi terutama pada bagian *backend* dalam membuat API untuk mengetahui *database* yang tepat untuk aplikasi mereka dari segi *response time* dalam lingkup API yang dibuat menggunakan Node JS.

1.2 Perumusan Masalah

Berdasarkan latar belakang masalah yang telah diuraikan, rumusan masalah dari penilitian ini adalah:

1. Bagaimana cara membangun *database* aplikasi QuestMD sesuai metode *database life cycle* (DBLC).
2. Bagaimana *response time* PostgreSQL, MySQL, dan MariaDB saat digunakan sebagai *database* untuk mendukung Restfull API di Node JS.
3. Bagaimana perbandingan *response time* API yang menggunakan *database* PostgreSQL, MySQL, dan MariaDB di API aplikasi pembelajaran.

1.3 Tujuan Penelitian

Tujuan penelitian ini sebagai berikut:

1. Membangun *database* aplikasi QuestMD menggunakan metode *database life cycle* (DBLC)
2. Menganalisis dan membandingkan *response time* dari *database* PostgreSQL, MySQL, dan MariaDB saat digunakan sebagai *database* pada Restfull API di Node JS.
3. Menganalisis *database* terbaik untuk *runtime environment* NodeJS dari perbandingan *response time* PostgreSQL, MySQL, dan MariaDB saat digunakan sebagai API di aplikasi pembelajaran.

1.4 Manfaat Penelitian

Manfaat penelitian ini sebagai berikut:

1. Membantu pengembang mengetahui *database* tercepat diantara *database* SQL *open source.*
2. Membantu pengembang untuk mengetahui *database open source* yang paling cocok pada *runtime environtment* Node JS *framework* Express diantara MySQL, PostgreSQL, dan MariaDB khususnya pada aplikasi QuestMD.

1.5 Batasan Masalah

Adapun batasan masalah dalam penelitian ini meliputi hal-hal sebagai berikut:

1. Bahasa yang digunakan pada API untuk menguji *response time database* adalah Javascript dengan *runtime environment* Node JS.
2. Pengujian *response time database* dikhususkan pada API aplikasi pembelajaran menggunakan *software* Jmeter.
3. Perancangan aplikasi pembelajaran untuk pengujian *response time* berbentuk *restfull* API.

1.6 Sistematika Penulisan

Sistematika penulisan yang digunakan pada skripsi ini terdiri dari 5 (lima) bab sebagai berikut:

BAB I : PENDAHULUAN

Pada bab ini secara umum memuat latar belakang penelitian, rumusan masalah, tujuan penelitian, batasan penelitian dan sistematika penulisan penelitian.

BAB II : TINJAUAN PUSTAKA

Pada bab ini memuat prinsip dan dasar-dasar teori yang menunjang dari penelitian skripsi.

BAB III : METODOLOGI PENELITIAN

Pada bab ini memuat metodologi penelitian yang digunakan dalam menyusun penelitian analisis perbandingan kinerja *response time query* PostgreSQL, MySQL, dan MariaDB.

BAB IV : PEMBAHASAN

Pada bab ini berisi hasil dan pembahasan yang diperoleh dari analisis perbandingan kinerja *response time* query PostgreSQL, MySQL, dan MariaDB Pada Restfull API Aplikasi E-Learning Menggunakan Node Js.

BAB V : KESIMPULAN DAN SARAN

Pada bab ini berisi kesimpulan dan saran hasil dari penelitian.

DAFTAR PUSTAKA

LAMPIRAN

II. TINJAUAN PUSTAKA

2.1 RESTful API

Restful API memiliki kepanjangan yaitu *Representational State Transfer Application Programming Interface*, *restful* API dibuat dengan pendekatan dari arsitektur perangkat lunak yang digunakan dalam merancang dan mengembangkan layanan web secara interaktif. Dasar dari *restfull* API adalah protokol HTTP (*Hypertext Transfer Protocol)* yang memungkinkan komunikasi tiap pengguna yang masuk di server dalam bentuk permintaan dan respons. [6].

Application Programming Interface (API) yang merupakan layanan antarmuka dan dikembangkan untuk dapat mengakses sistem yang dibuat secara terprogram. API memungkinkan pengembang untuk mengambil data dari aplikasi yang dibuat oleh pengembang lain tanpa perlu membuat sendiri fungsi tersebut [7]. Sedangkan nama REST diciptakan oleh Roy Fielding REST API dapat berkomunikasi baik mengirim atau menerima data dengan cepat dan sangat sederhana [8].

Layanan pada sebuah *web service* dapat dikatakan *restful* API jika layanan tersebut memenuhi seluruh syarat yang ada pada arsitektur REST [9]. Pada layanan REST pengembang dapat mengakses *endpoint* atau titik akhir melalui *HTTP* atau aplikasi layanan lainnya. Layanan *endpoint* inilah yang nantinya digunakan untuk operasi CRUD (*create, read, update,* dan *delete)* serta dipakai oleh *frontend* dalam pengembangan aplikasi yang dibuat. Pada penerapannya *restful* API yang dibuat oleh *backend* ini berbentuk JSON (*JavaScript Object Notation*). JSON inilah yang diakses oleh *frontend* yang nantinya data tersebut ditampilkan ataupun digunakan di tampilan yang dibuat.

2.2 Node JS

Node JS adalah platform perangkat lunak yang berjalan di sisi server dan aplikasi jaringan. Node JS ditulis menggunakan bahasa Javascript yang dapat berjalan pada sistem Windows, Mac OS, dan Linux [10]. Pada praktiknya Node JS perlu perangkat lunak editor untuk mengetik *script*, contohnya yaitu Visual Studio Code. Namun perangkat lunak tersebut tidak terbatas pada satu jenis editor saja tergantung dari pengembang untuk memilih perangkat lunak yang cocok. Node JS sendiri memiliki beberapa *framework* yang digunakan oleh pengembang, seperti Express, Koa, Meteor, dan masih banyak lagi. Namun *framework* yang paling sering digunakan oleh pengembang saat ini yaitu Express.

Pada sejarahnya Node JS pertama kali diciptakan dan diperkenalkan oleh Ryan Dahl dan disponsori oleh Joyent. Untuk menjalankan Node JS saat itu 2009 pengguna harus menggunakan Linux [10]. Node JS Memiliki kelebihan yaitu pada pendekatan *non-blocking* yang memungkinkan sebuah operasi dapat dijalankan secara simultan yaitu tanpa harus menunggu penyelesaian dari proses sebelumnya [11].

2.3 Express JS

Framework yang merupakan kerangka kerja dari aplikasi dan berisi kumpulan kode-kode yang tersusun rapi pada sebuah folder yang terstruktur dengan tujuan dibuat untuk memudahkan pengembang dalam membuat sebuah aplikasi dalam bidang *backend* [12]. Sehingga dapat diartikan Express merupakan sebuah kerangka kerja (*framework*) di Node.js yang berguna dalam pembuatan server web serta aplikasi jaringan. Dengan Express, pengembang dapat lebih mudah mengelola permintaan HTTP, melakukan *routing*, dan menggunakan *middleware* [13]. Express Js juga merupakan *framework* dengan bahasa Javascript yang paling banyak digunakan oleh para pemula saat ini, hal ini dikarenakan Express Js paling mudah dipelajari jika ingin membuat API dengan bahasa Javascript yang biasanya pengembang sebelumnya sudah mempelajari *frontend* yang sama-sama menggunakan bahasa Javascript. Sehingga ketika masuk dan belajar *backend*

pengembang tidak perlu lagi belajar bahasa baru dan cukup lanjutkan dengan menggunakan Express Js.

2.4 *Object Relational Mapping* (ORM)

Object Relational Mapping adalah teknik yang dapat memetakan antara objek dan *database* menggunakan sebuah metadata ke *database* relational sebagai objek dan data [14]. Teknik ini dapat mempermudah pengembang dalam menjembatani antar SQL sehingga tidak harus menulis *query* SQL secara utuh, yang awalnya menulis *query* sampai beberapa baris, dengan ORM maka bisa mempermudah hingga hitungan baris saja. Beberapa jenis dari ORM pada Javascript adalah Sequelize, Prisma, TypeORM, MikroORM, dan masih banyak lagi.

2.4.1 Sequelize

Sequelize adalah Object Relation Mapping (ORM) yang pertama kali berbasis *promise-based* yang diluncurkan pada tahun 2011. Sequelize sendiri memiliki dokumentasi yang lengkap serta luas yang dapat digunakan dengan menggunakan Node JS. Sequelize juga mendukung beberapa *database* seperti PostgreSQL, MySQL, MariaDB, SQLite, dan MSSQL.

Sedangkan pengertian Sequelize adalah kerangka kerja Object-Relational Mapping (ORM) untuk Node.js yang mengadopsi model pemrograman berbasis promise. Alat ini dirancang untuk bekerja dengan berbagai jenis *database* seperti PostgreSQL, MySQL, MariaDB, SQLite, dan MSSQL. Dengan Sequelize, Pengembang dapat mengelola transaksi dengan efisien, mengatur hubungan antar tabel, menerapkan *lazy loading* dan *eager loading*, serta mengakses fitur seperti membaca replikasi dan banyak lagi [15].

2.5 *Relational Database Management System* (RDBMS)

Relational Database Management System (RDBMS) adalah terminologi atau konsep yang digunakan untuk menyediakan fasilitas menyimpan dan mengambil data kepada pengguna, yang saling terkait dengan relasi lain, yang disebut model relasional . memungkinkan pengguna *database* untuk membuat, mengatur, dan menggunakan data dalam bentuk relasional [16]. Beberapa contoh RDBMS meliputi MySQL, PostgreSQL, MariaDB, dan sejenisnya.

2.5.1 MySQL

MySQL adalah sistem manajemen *database* relasional (RDBMS) yang cepat dan mudah digunakan, cocok untuk berbagai kebutuhan. Pengembangan MySQL dilakukan oleh MySQL AB, perusahaan asal Swedia [17]. Arsitektur MySQL menggunakan model *klien-server*, di mana pusat kendali berada di server. Server ini adalah program yang memungkinkan manipulasi *database*. Klien, pada prosesnya, tidak berinteraksi langsung dengan *database*, melainkan berkomunikasi dengan server melalui *query* dalam bahasa SQL (*Structured Query Language*) [18].

2.5.2 MariaDB

Pada awalnya MySQL yang diakuisi oleh Perusahaan Oracle Corporation sehingga membuat salah satu pembuat MySQL yaitu Michael Widenius memutuskan untuk membuat sebuah *database* baru yang memisahkan diri dari MySQL dan dikenal sekarang sebagai MariaDB. Kemudian MariaDB berkembang dan memiliki beberapa keunggulan dibandingan MySQL. Seperti pemakaian yang menjadi lebih mudah, stabil, dan dapat mengakses banyak data [19]. Hingga saat ini MariaDB menjadi salah satu *database* yang banyak dipilih oleh para pengembang.

2.5.3 PostgreSQL

Menurut lisensi BSD, PostgreSQL merupakan salah satu sistem *database* yang dapat didistribusikan secara bebas [20]. PostgreSQL adalah *database* sumber terbuka dengan kinerja yang sangat baik dibandingkan dengan sistem *database*

lainnya. PostgreSQL memiliki fitur yang sangat lengkap, menjadikannya pilihan yang sangat cocok untuk aplikasi *database* dengan skala menengah hingga besar [21]. Pada awalnya, PostgreSQL dikembangkan oleh mahasiswa dan staf *programmer* di University of California, Berkeley, di bawah supervisi Professor Michael Stonebraker. Proyek ini pertama kali disebut Postgres, tetapi karena semakin banyaknya fitur SQL yang ditambahkan, pada tahun 1995 namanya diubah menjadi Postgres95. Pada tahun 1996, nama tersebut diubah lagi menjadi PostgreSQL, dan sejak itu nama ini tetap digunakan hingga saat ini.

2.6 Apache Jmeter

Apache Jmeter merupakan sebuah aplikasi sumber terbuka yang berbasis Java, berguna bagi QA Engineer dalam melakukan pengujian kinerja. JMeter dapat digunakan untuk melakukan uji beban dan stres pada berbagai aplikasi seperti Web Application, FTP Application, serta server Database. Dengan Apache JMeter, pengujian kinerja bisa dilakukan baik pada sumber daya yang bersifat statis maupun dinamis, seperti layanan web (SOAP / REST), beragam bahasa pemrograman web (seperti PHP, Java, ASP.NET), berkas, objek Java, *database* beserta kueri-kuerinya, server FTP, dan sebagainya. Fungsinya meliputi simulasi beban berat pada server, kelompok server, jaringan, atau objek tertentu untuk menguji daya tahan atau menganalisis kinerja secara keseluruhan dalam kondisi beban yang berbeda[22].

2.7 Normalisasi *Database*

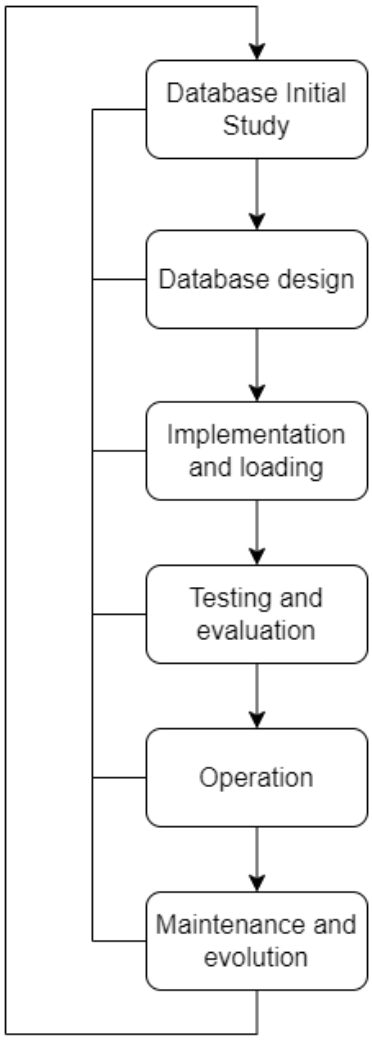
Normalisasi adalah suatu parameter yang digunakan untuk mencegah terjadinya duplikasi pada tabel dalam *database*. Ini juga merupakan proses yang dilakukan untuk memecah tabel yang masih mengandung anomali atau ketidakwajaran, sehingga menghasilkan tabel yang lebih sederhana dan struktur yang lebih baik. Dengan normalisasi, tujuan utamanya adalah menciptakan tabel tanpa adanya redundansi data, sehingga memungkinkan pengguna untuk melakukan operasi insert, delete, dan update pada baris (record) tanpa menimbulkan inkonsistensi data [23].

Normalisasi *database* dibagi menjadi tiga yaitu 1NF dengan aturan sebuah atribut tidak boleh bernilai banyak atau mengandung atribut lain dan setiap atribut pada tabel unik atau tidak sama, kemudian 2NF dengan aturan sebuah tabel bergantung pada satu buah kunci yaitu *primary key* dan jika terdapat data yang memiliki kesamaan maka perlu dibuat *foreign key* dan tetap bergantung pada *primary key* utama, kemudian yang terakhir 3NF yaitu dengan aturan hanya atribut kunci yang dapat bergantung pada *primary key* dan atirubut *foreign key* dibuat tabel baru.

2.8 *Database Life Scycle* (DBLC)

Database Life Cycle (DBLC) merujuk pada siklus hidup sebuah *database* yang digunakan dalam perancangan sistem *database*. Model DBLC melibatkan proses dasar dalam merencanakan kerangka kerja dari struktur logis *database*, menentukan cara data disimpan dan diakses di berbagai komputer, serta menetapkan kerangka kerja khusus bagi sistem manajemen *database* lokal. Setelah selesai merancang, siklus hidup berlanjut ke tahap implementasi dan pemeliharaan *database* [24].

DBLC terdiri dari enam tahap, mulai dari *database initial study, database design, implementation and loading, testind and evaluation, operation, dan maintenance and evolution*. Berikut adalah gambaran lebih jelas mengenai tahapan dari DBLC [24]:



Gambar 2. 1 Tahapan Perancangan DBLC

a. Database Initial Study

Tahap awal dalam pengembangan atau perbaikan sistem *database*, yang dikenal sebagai *Database Initial Study*, memiliki beberapa langkah krusial yang perlu dipahami secara mendalam. Tahapan ini merupakan fondasi utama dalam penyusunan skripsi terkait dengan *database*. Dalam tahap ini menjelaskan mulai dari perencanaan *database*, pendefinisian sistem, dan analisis pengumpulan kebutahan *database*.

b. Database Design

Fase kedua dalam *database life cycle* (DBLC) fokus pada desain model *database* yang mendukung operasi dan tujuan perusahaan. Fase ini dapat dianggap sebagai fase paling krusial dalam DBLC karena memastikan bahwa produk akhir memenuhi

persyaratan pengguna dan sistem. Pada tahap ini dibuat beberapa model *database* seperti:

a. *Conceptual database Design*

   Conceptual database Design merupakan tahap dalam pembangunan model yang didasarkan pada informasi yang telah dikumpulkan, tanpa mempertimbangkan aspek perencanaan fisik, dan bersifat independen dari segala pertimbangan fisik.

b. *Logical Database Design*

   Logical Database Design merupakan tahap dalam pembuatan model informasi untuk perusahaan, yang didasarkan pada suatu model data tertentu, namun tidak bergantung pada *Database Management System* (DBMS) khusus atau pertimbangan fisik lainnya. Pada tahap ini juga dilakukann pengecekan normalisasi dari *database* mulai dari 1NF, 2NF, dan 3NF.

c. *Physical Database Design*

   Phisical Database Design merupakan langkah transformasi dari perancangan logis *database* yang sebelumnya dibuat menjadi bentuk fisik.

c. Implementation and Loading

Implementasi adalah proses pembuatan *database* secara langsung berdasarkan hasil desain yang telah dibuat sebelumnya. Pada proses ini menggunakan DBMS (*Database Management System*) sebagai alat untuk membuat *database* tersebut. Selain membuat database langsung pada DBMS pada tahap ini juga dilakukan pemasukan data kedalam database jika ada data yang harus dimasukkan.

d. Testing and Evaluation

Tahap *testing and evaluation* adalah tahap melaukan pengetasan pada database apakah berjalan dengan baik atau tidak, dalam pengetesan ada berbagai cara untuk mengetahui apakah database berjalan lancar atau tidak seperti melakukan CRUD (Create, Read, Update, Delete) kedalam database sesuai dengan spesifikasi database dan pengetesan juga dapat diuji mengenai kecepatan dari pemroresan CRUD dalam banyak data yang berbeda.

e. Operation

Operation adalah tahap rancangan *database* sebelumnya digunakan oleh pengembang untuk diimplementasikan didalam proyek yang pengembang buat. Dalam proses implementasi kedalam aplikasi terkadang terdapat perubahan database yang perlu dilakukan, serta tambahan jika ternyata ada ketidakcocokan ketika dibuat kedalam proyek sehingga perlu diperbaiki rancangan *database* tersebut.

f. Maintenance and Evolution

Tahap *maintenance and evolution* merupakan tahap pemeliharaan pada database. Setelah database diimplementasikan pada aplikasi yang dibuat database tidak ditinggalkan begitu saja, sehingga perlu dilakukan pemeliharan agar database terus berjalan normal. Seperti misalnya jika terdapat *update* terbaru pada DBMS yang digunakan maka database yang kita miliki mungkin diperbaiki juga.

2.9 *Entity Relation Diagram* (ERD)

Entity Relation Diagram (ERD) adalah alat pemodelan data semantik yang digunakan untuk mencapai tujuan mendeskripsikan atau menggambarkan data secara abstrak serta menggambarkan koneksi antara data dalam suatu *database* berdasarkan objek-objek data mendasar yang memiliki relasi di antara mereka [25].

Dalam *Entity-Relationship Diagram* (ERD), terdapat tiga elemen dasar yang mendefinisikan struktur data, yaitu entitas, atribut, dan relasi.

1. Entitas adalah representasi objek dalam *database*, yang bisa berupa manusia, tempat, benda, atau informasi lainnya. Setiap entitas memiliki atribut dan *primary key* yang mengidentifikasikannya. Entitas direpresentasikan oleh simbol persegi panjang.
2. Atribut adalah informasi yang terkait dengan entitas. Entitas harus memiliki *primary key* sebagai pengenal unik, dan atribut merupakan deskripsi dari entitas tersebut. Atribut bisa disimpan dalam tabel entitas atau terpisah dari tabel. Simbol atribut direpresentasikan dengan bentuk elips.

3. Relasi dalam ERD menggambarkan hubungan antara dua atau lebih entitas. Simbol relasi berbentuk belah ketupat. Terdapat beberapa jenis relasi dalam ERD, yaitu *One to One* (satu entitas berhubungan dengan satu entitas lainnya), *One to Many* (satu entitas berhubungan dengan beberapa entitas lainnya), dan *Many to Many* (banyak entitas berhubungan dengan banyak entitas lainnya) [26]

2.10 Penelitian Terkait

Penelitian terkait adalah penelitian terdahulu yang pernah dilakukan sebelumnya. Penelitian terkait ini dapat membantu penelitian-penelitian saat ini yang ingin dibuat sebagai referensi dan mendukung penelitian tersebut. Berikut adalah penelitian yang berhubungan dan menjadi pendukung dalam penelitian ini.

2.10.1 Perbandingan Kinerja Cassandra dan MongoDB Sebagai *Backend* IoT *Data Storage*

Penelitian ini dilakukan oleh Adama Kukuh Kurniawan, Eko Sakti Pramukantoro, dan Primantara Hari Trisnawan. Penelitian ini bertujuan membandingkan *database* Cassandra dan MongoDB pada lingkungan *backend* di alat IoT sehingga diketahui *database* mana yang paling tepat dan cocok pada IoT tersebut. Terdapat dua buah pengujian dalam penelitian ini yaitu pengujian fungsionalitas dan pengujian kinerja. Berdasarkan hasil pengujian fungsionalitas Casandra sebagai sistem manajemen *database* NoSQL, mampu menyimpan beragam data yang berasal dari *node sensor* yang terdistribusi dalam suatu topologi sistem yang digunakan. Sedangkan, pada pengujian kinerja ketika ada pertukaran data dengan ukuran yang besar, NoSQL Cassandra menjadi pilihan yang efektif untuk menyimpan data tersebut. Namun, jika ukuran data yang dipertukarkan lebih kecil, NoSQL MongoDB menjadi solusi yang lebih tepat sebagai tempat penyimpanannya [27].

2.10.2 *Database Systems Performance Evaluation For IOT Applications*

Penelitian ini dilakukan oleh Critodoulos Asiminidis, George Kokkonis, dan Sotirios Kontogiannis. Penelitian ini bertujuan membandingkan *database* PostgreSQL, MySQL, dan MongoDB pada lingkungan *backend* di alat IoT sehingga diketahui *database* mana yang paling diantara kedua *database*. Dalam penelitian ini salah satu pengujian yang utama yaitu pengujian *select* dan *insert.* Pada pengujian *select* pada pengujian ambil data dibawah 100.000 data PostgreSQL lebih cepat diatara MySQL dan MongoDB, namun ketika data yang diambil diatas 100.000 maka MySQL lebih cepat dibandingkan dengan PostgreSQL, namun jika dibandingkan dengan MongoDB lebih cepat MongoDB. Kemudian, pada pengujian *insert* pada sejumlah besar pengujian PostgrSQL sedikit lebih baik dibandingkan dengan MySQL dan MongoDB [28].

2.10.3 Studi Perbandingan Performansi Antara MySQL Dan PostgreSQL

Penelitian ini dilakukan oleh Ardian Dwi Praba, Maryanah Safitri STMIK Nusa Mandiri Jakarta. Penelitian ini bertujuan untuk mengetahui *database* yang tercepat antara MySQL dan PostgreSQL. Penelitain ini menggunakan perangkat lunak Adminer untuk melakukan tes di kedua *database* tersebut. Tes yang dilakukan dalam penelitian ini terdapat tiga skenario berbeda yaitu skenario *select* seluruh data, *Select* dengan *Inner Join,* dan *select* dengan *count* menggunakan 50.000, 100.000, dan 1.000.000 data. Berdasarkan hasil penelitian didapatkan bahwa perbandingan performa *database* antara MySQL dan PostgreSQL dari segi *response time* bahwa *database* PostgreSQL lebih cepat dibandingkan *database* MySQL [29].

2.10.4 Analisis Perbandingan Kinerja *Query Database Management System* (DBMS) Antara MySQL 5.7.16 dan MariaDB 10.1

Penelitian ini dilakukan oleh Indra Warman dan Rizki Ramdaniansyah dari Institut Teknologi Padang. Penelitian ini memiliki tujuan untuk menguji perbandingan antara *database* MySQL dan PostgreSQL berdasarkan *response time* pertukaran data. Pengujian pada penelitian ini menggunakan beberapa jumlah data yang

berbeda yaitu 50, 100, 1000, 5000, 10.000, dan 100.000 data. Selain jumlah data yang berbeda pengujian ini dilakukan di beberapa kondisi seperti *inser,update, select, average, count, max, min, sum, and, or, stored procedure, trigger event insert,* dan *trigger event update.* Berdasarkan hasil penelitian ini didapatkan kesimpulan bahwa MySQL hanya unggul pada pengujian *stored procedure* pada jumlah 100.000 data. Sedangkan, *database* MongoDB jauh lebih unggul untuk banyak kondisi seperti *insert, update, select, average, count, max, min, sum, and, or, trigger event insert, dan trigger event update* dengan jumlah pengujian 100.000 data [30].

2.10.5 Experiments of Search Query Performance for SQL-Based Open Source Databases

Penelitian ini dilakukan oleh Meekyung Min berasal dari departemen komputer sains Universtas Seokyeong. Pada penelitian mengkaji mengenai perbandingan performa dari *database* komersial dengan *database open source. Database* komersial diwakili oleh Microsoft SQL Server sedangkan *database open source* diwakili oleh MySQL. Terdapat beberapa kondisi yaitu *single query, multi-condition query, range query,* dan *join query.* Berdasarkan hasil penelitian didapatkan kinerja *database* open source paling dipengaruhi oleh ada atau tidaknya indeks. Saat indeks digunakan, performa database *open source* sering kali melebihi performa *database* komersial, bergantung pada jenis *query* penelusuran. Selanjutnya, kinerja dipengaruhi oleh jumlah tupel. Semakin besar jumlah tupel, semakin sering kinerja *database open source* menurun. Terakhir, dalam hal jenis *query*, dalam kueri gabungan, performa *database* open source tidak mencapai performa *database* komersial [31].

2.10.6 Perbandingan Performa SQL dan NoSQL Dengan PHP pada 5 Juta Data

Penelitian ini dikerjakan oleh Setiawan Budiman bersama keempat orang lainnya dari Universitas Amikom Yogyakarta. Tujuan penelitian ini yaitu untuk mencari database yang paling ideal diantara SQL dan NoSQL dengan membandingkan

performa database MySQL dan MongoDB sebagai perwakilan database tersebut. Proses pengujian dalam penelitian dicoba dalam bebera kondisi seperti *select, insert, delete,* dan *update* serta dalam jumlah data yang berfariasi mulai dari 1000, 10.000, 100.000, 1.000.000, 2.000.000, dan 5.000.000 data. Berdasarkan hasil pengujian didapatkan bahwa pada proses *select, insert,* dan *update* MongoDB memiliki waktu respon tercepat. Sedangkan, untuk kondisi *delete* MySQL memiliki waktu respon yang lebih baik [32].

2.10.7 Analisis Perbandingan Performa Database DuckDB Dan SQLite Pada Pengolahan Big Data

Penelitian ini dilakukan oleh dua orang yaitu Farid A. Nugraha dan Yerymia A. Susetyuo dari Universitas Kristen Satya Wacana. Sesuai dengan judul tujuan dari penelitian ini untuk membuktikan *database* mana yang memiliki performa yang lebih baik saat mengolah data mulai dari 100 sampai 10.000 data. Terdapat beberapa kondisi yang diuji yaitu *query insert, update, delete, select, sum, count, max, average.* Berdasarkan hasil penelitian antara kudua *database* didapatkan SQLite unggul pada kondisi *insert, update* (seluruh data)*, delete,* dan *select* data. Sedangkan DuckDB unggul pada *update* dua buah data dan semua fungsi agregat pengujian[33].

2.10.8 Pengembangan Database E-commerce De Janggelan Menggunakan Metode Database Life Cycle

Penelitian ini dilakukan oleh Sugiarto dan Evi Triandini dari UPN Veterjan Jawa Timur. Penelitian ini bertujuan untuk melakukan pengembangan *database* yang efisien dan mudah dijalankan sehingga kedepannya mempermudah *maintenance* dikemudian hari pada usaha Janggelan. Metode yang digunakan dalam pengembangan *database* ini yaitu *Database Life Cycle* (DBLC) yang nantinya menghasilkan model *database* secara bertahap. Berdasarkan hasil dari penelitian didapatkan kesimpulan yaitu berhasilnya pembuatan model *database* berupa *conceptual model data, logical model data,* dan *physical model data.* Model

database pada hasil pengembangan ini sudah termasuk pembuatan *database* pada DBMS MySQL [34].

2.10.9 Desain Model Database Layanan Panti Werdha dengan Menerapkan Metode *Database Life Cycle*

Pada penelitian ini dilakukan oleh Stefanus Setyo Wibagso dan Enjel Lia yang berasar dari Universitas Katolik Musi Charitas. Penelitan ini bertujuan Menyusun model *database* bagi layanan Panti Werdha sehingga dapat digunakan untuk menyimpan dan mengelola data yang ada. Penelitian ini menggunakan metode perancangan Database Life Cycle (DBLC) yang dapat membuat *database* secara efisien dan mudah dikelola. Terdapat tiga fase dalam perancangan ini yaitu fase 1 (satu) merupakan perencanaan *database*, pendefinisian sistem, dan Analisa pengumpulan data. Kemudian fase 2 (dua) yaitu desain konseptual *database*, desain logika *database*, dan desain fisik *database*. Selanjutnya, yaitu fase 3 (tiga) yaitu tahap akhir berupa implementasi langsung pada *database*. Hasil penelitian yaitu didapatkannya model rancangan *database* rasional sehingga dapat menyimpan dan mengelola data yang ada pada Panti Werdha [35].

2.10.10 Boarding House Search Information System Database Design

Pada Penelitian ini dilakukan oleh Muhamad Rehan Anwar dan Suryari Purnama dari University of Africulture Faisalabad dan Esa Unggl University. Penelitian ini bertujuan untuk membuat sebuah desain *database* pada sebuah rumah kost. Penelitian dikerjakan menggunakan metode pembuatan *database* yaitu Database Life Cycle (DBLC) dan menghasilkan model *database* berupa *conceptual, logical, dan physical design*. Hasil dari penelitian yaitu berupa desain konseptual sebanyak 7 (tujuh) entitas, desain logis menghasilkan gambaran hubungan antar entitas, dan desain fisik menghasilkan tabel yang telah direalisasikan di *database* [36].

2.10.11 Perancangan Aplikasi Pembelajaran Dengan Konsep Gamifikasi

Penelitian pada jurnal ini dilakukan oleh Erba Lutfina dan lainnya dari Universitas Nasional Karang Turi. Penelitian ini bertujuan untuk melakukan tinjauan Kembali terhadap konsep *gamifikasi* yang telah diuji oleh para peneliti dalam beberapa tahun terakhir. Pada penelitian ini dihasilkan bahwa terdapat 5 (lima) buah elemen atau fitur yang paling banyak dipakai dalam sebuah aplikasi tema *gamifikasi* yaitu "*level"* dengan 8 penelitian, "*point"* dengan 7 penelitian, *"leaderboard"* dengan 6 penelitian, *"badge"* dengan 6 penelitian, dan *"challenge"* dengan 4 penelitian [37].

2.10.12 Gamifikasi Untuk Pembelajaran

Penelitian pada jurnal ini dilakukan oleh Diana Ariani dari Universitas Negeri Jakarta. Penelitian ini bertujuan untuk untuk memberikan penjelasan mengenai definisi gamifikasi, komponen mendasar dalam gamifikasi, berbagai jenis gamifikasi, dan bagaimana penerapan gamifikasi dalam LMS E-learning. Pada penelitian ini didapatkan 5 (Lima) buah fitur atau elemen yang digunakan untuk sebuah aplikasi gamfikasi yaitu *point*, lencana, *level*, papan peringkat, dan *avatar* [38].

III. METODOLOGI PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian ini dilaksanakan pada semester ganjil tahun ajaran 2023 dengan rentang waktu bulan November 2023 hingga bulan Maret 2024. Penelitian bertempat di Jurusan Teknik Elektro, Program Studi Teknik Informatika, Universitas Lampung. Berikut adalah tabel alur penelitian yang dilakukan:

Tabel 3. 1 Alur Penelitian

Kegiatan	**Waktu Penelitian**				
	November	**Desember**	**Januari**	**Februari**	**Maret**
Database Initial Study					
Database Design					
Implementation and Loading					
Testing and Evaluation					
Maintenance and Evolution					
Analisis Kinerja *Database*					
Penulisan Laporan					

3.2 Alat Penelitian

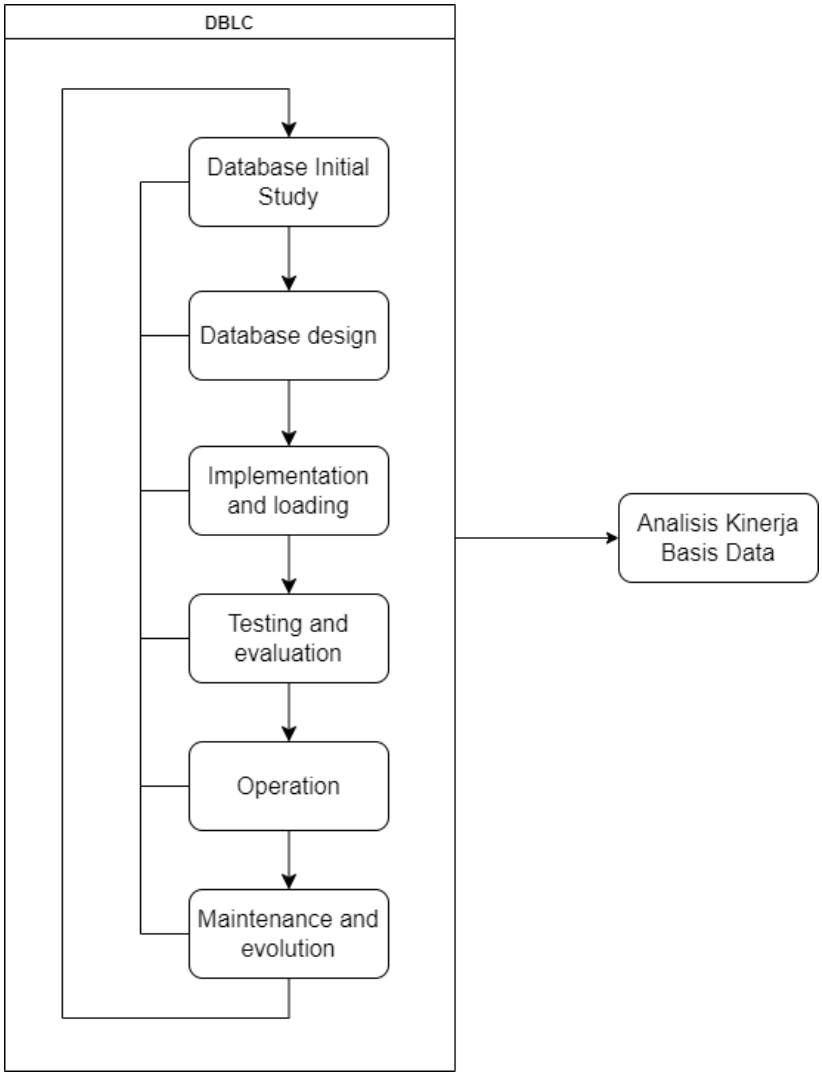
Adapun alat penelitian yang digunakan dalam pelaksanaan penelitian ini yaitu sebagai berikut:

Tabel 3. 2 Alat Penelitian

No	***Hardware* dan *Software***	**Spesifikasi**	**Deskripsi**
1	Laptop Acer Swift SF314-511	Processor intel i5-1135G7, Ram 16 GB, Sistem Operasi Windows 11	Perangkat keras yang digunakan untuk melakukan proses penelitian.
2	Visual Studio Code	Versi 1.84.2	Perangkat lunak yang digunakan untuk pembuatan API.
3	Apache Jmeter	Versi 5.6.2	Perangkat lunak yang digunakan untuk testing *respon time* server.
4	Postman	Versi 10.19.17	Perangkat lunak yang digunakan untuk menjalankan API.
5	DBeaver	Versi 22.0.5.202205220912	Perangkat lunak yang digunakan untuk mengakses bermacam *database* seperti MySQL, PostgreSQL, MariaDB, dan lainnya

3.3 Tahapan Penelitian

Penelitian ini dilakukan dalam beberapa tahapan mulai dari tahapan metode *Database Life Cycle* (DBLC) yang dilanjutkan analisisi kinerja *database* dan penulisan laporan. Berikut adalah gambaran tahapan penelitian yang dilakukan :



Gambar 3. 1 Tahapan Penelitian

Berdasarkan gambar 3.1 tahapan penelitin dimulai dari pembuatan *database* menggunakan metode *Database Life Cycle* (DBLC) yang didalamnya terdapat enam tahapan yaitu *database initial study, database design, implementation and loading, testing and evaluation, operation,* dan *maintenance*. Tahapan DBLC tersebut sudah mencakup pembuatan aplikasi termasuk pembuatan API. Setelah

tahapan DBLC telah selesai dibuat barulah menuju tahapan analisis kinerja *database* dan membandingkan tiga *database* yang ada. Kemudian yang terakhir penulisan laporan yang merupakan hasil dari penelitian yang telah dilakukan.

3.3.1 *Database Initial Study*

Pada *database initial study* terbagi menjadi tiga bagian yaitu perencanaan *database*, pendefinisian sistem, dan analisis dan pengumpulan kebutuhan.

a. Perencanaan *Database*

Pada tahap perencanaan *database* dijelaskan mengenai *Mission Statement* dan *Mission Objective* dari *database* yang dibuat. *Mission statement* dari penelitian ini adalah merancang dan membangun *database* dengan tema gamifikasi dalam aplikasi pembelajaran kedokteran QuestMD. Sementara itu, *Mission objective* dari penelitian ini menggunakan riset dari jurnal yang menjelaskan elemen dan fitur yang diperlukan dalam sebuah aplikasi pembelajaran untuk meningkatkan minat belajar. Penting untuk dicatat kembali bahwa tujuan dari aplikasi QuestMD adalah menciptakan pengalaman pembelajaran yang tepat dan menyenangkan bagi para mahasiswa kedokteran, dengan menerapkan konsep gamifikasi untuk mengurangi stres. Setelah dilakukan pencarian mengenai gamifikasi, ditemukan dua jurnal yang paling relevan untuk mendukung *Mission objective* penelitian ini dan memuat fitur dan elemen yang dibutuhkan dalam sebuah aplikasi gamifikasi, yaitu jurnal "Perancangan Aplikasi Pembelajaran Dengan Konsep Gamifikasi" dan "Gamifikasi untuk Pembelajaran". Berdasarkan kedua jurnal tersebut, fitur gamifikasi yang digunakan adalah sebagai berikut:

1. *Level*

   Level yaitu elemen yang berisi seberapa tinggi tingkatan suatu pengguna dihitung berdasarkan banyaknya *experience point* yang didapatkan ketika mengerjakan kasus yang ada pada aplikasi.

2. *Coins*

   Coins yaitu elemen yang berguna untuk membeli barang seperti *avatar* didalam aplikasi, *coins* didapatkan dari mengerjakan kasus yang ada dalam aplikasi.

3. *Leaderboard*

   Leaderboard adalah elemen yang berguna untuk mengetahui peringkat setiap pengguna, *leaderboard* juga dapat dilihat oleh setiap pengguna sehingga pengguna dapat berlomba-lomba menjadi juara pertama setiap bulannya.

4. *Badge*

   Badge yaitu elemen yang berisi prestasi atau pencapaian yang telah diraih pengguna, prestasi ini tidak dapat dibeli dengan *coins* sehingga hanya *user* yang benar-benar sesuai dengan kriteria mendapatkan prestasi-prestasi tertentu.

5. *Challenge*

   Challenge yaitu elemen atau fitur yang berisi kasus-kasus kedokteran yang Dimana pengguna memecahkan kasus tersebut sesuai penanganan dan penyakit yang *user* pilih.

6. *Avatar*

   Avatar yaitu elemen yang berisi foto profil pengguna yang didapatkan dari hadiah ataupun pembelian pada toko aplikasi.

b. Pendefinisian Sistem

Sesudah dilakukannya proses perencanaan *database* yang menghasilkan elemen-elemen *gamifikasi*. Selanjutnya berdasarkan elemen yang telah dibuat diidentifikasi jenis pengguna apa saja yang akan mengakses aplikasi tersebut dan peran yang yang dapat dilakukan setiap pengguna tersebut.

Pada penelitian ini terdapat dua pengguna yang dapat mengakses aplikasi yaitu *user* dan admin yang kedua pengguna sama-sama dapat mengakses elemen pada setiap aplikasi namun *user* memiliki keterbatasan terhadap beberapa elemen penting di dalam aplikasi. Berikut adalah aktivitas yang dapat dilakukan oleh pengguna yaitu:

Tabel 3. 3 Aktivitas Pengguna

No	Aktor	Aktivitas
1	*user*	• Mengelola data profile pribadi • Mengelola kepemilikan *avatar*, seperti membeli *avatar* menggunakan *coins.* • Mengerjakan *challenge* • Melihat peringkat *leaderboard* saat ini. • Mendapatkan *badge* berdasarkan *challenge* atau *leaderboard.* • Mendapatkan *coins* dan menaikkan *level* saat mengerjakan *challenge.* • Memberikan komentar pada *challenge.*
2	Admin	• Mengelola data *badge* yang ada dalam aplikasi. • Mengelola data *challenge* beserta penyakit, penanganan, *stase*, bab dan *investigasi* dari *challenge* tersebut. • Mengelola data *avatar* yang ada dalam aplikasi.

c. Analisis dan Pengumpulan Kebutuhan

Berdasarkan jurnal "Tingkat Stres dan Pencapaian Kompetensi Mahasiswa Program Profesi Dokter: Penelitian Kuantitatif dan Kualitatif" didapatkan bahwa tingkat stres mahasiswa kedokteran baik yang masih kuliah dan mahasiswa koas memiliki tingkat stres yang tinggi, sehingga dibutuhkan media pembelajaran baru yang dapat mengurangi stres dan tidak mengurangi atau mengganggu proses belajar mengajar[39]. Pada aplikasi QuestMD terdapat *challenge* yang merupakan kuis didalam aplikasi tersebut, kuis ini berbentuk sebuah kasus penyakit dan *user* nantinya mengisi penyakit dan penanganan apa yang perlu dilakukan terhadap kasus tersebut serta terdapat fitur lainnya berdasarkan perencanaan *database* sebelumnya. Berdasarkan alasan tersebut didapatkan kebutuhan dari aplikasi pembelajaran QuestMD untuk mahasiswa koas dengan tema *gamifikasi* dengan mempertimbangkan tahap sebelumya yaitu pendefinisian sistem.. Berikut adalah kebutuhan secara lengkap *database*:

- *Database* mampu menampung dua pengguna yaitu user dan admin.
- *Database* mampu menampung data *level*, *Coins*, *leaderboard*, *badge*, *challenge*, dan *avatar*.
- *Database* mampu menampung data *avatar* dan *badge* yang dimiliki oleh *user.*
- *Database* mampu memiliki *challenge* serta turunan *challenge* berupa penyakit, penanganan, investigasi dan pengelompokkan *challenge* berdasarkan jenis penyakit kasus *challenge* tersebut.
- *Database* mampu menampung data jawaban penyakit dan jawaban penanganan.
- *Database* mampu menampung data kunci jawaban penyakit dan penanganan dari *user*.
- *Database* mampu menampung data diskusi dan komentar dari setiap *challenge.*
- *Database* mampu menampung data Riwayat pengerjaan *challenge.*

Kebutuhan *database* diatas memenuhi kebutuhan aplikasi QuestMD yang merupakan aplikasi pembelajaran bagi mahasiswa kedokteran. Aplikasi ini diharapkan mampu memberikan mahasiswa motivasi belajar baru dan pengalaman belajar interaktif serta menyenangkan.

3.3.2 *Database Design*

Database design adalah tahap yang berfokus pada pembuatan model desain dari *database* yang ingin dibuat. Model *database* yang dibuat pada tahap ini dibagi menjadi tiga yaitu:

a. Desain Konseptual *Database*

Pada proses desain konseptual *database* terdapat dua kegiatan yang dilakukan yaitu idenfikasi tipe entitas dan identifikasi tipe relasional *database* yang dibuat. Berikut adalah tabel identifikasi tipe entitas dan tabel identifikasi tipe relasional.

Tabel 3. 4 Identifikasi Tipe Entitas

No	Nama Entitas	Deskripsi	Kegiatan
1	*user*	Entitas yang berisi data *user*.	Semua *user* yang menggunakan aplikasi.
2	*Leaderboard*_value	Entitas yang berisi data peringkat *user* setiap bulan.	Setiap kali *user* mengerjakan *challenge* menambahkan *score leaderboard*.
3	*Leaderboard*	Entitas yang berisi tanggal *leaderboard*.	Setiap awal bulan membuat data tanggal periode baru.
4	*My_achievement*	Entitas yang berisi *achievement* yang dimiliki *user*.	Setiap kali *user* menyelesaikan misi *achievement*.
5	*Achievement*	Entitas yang berisi data *achievement*.	Semua data *achievement* atau lencana yang dimasukkan admin.
6	*My_avatar*	Entitas yang berisi data *avatar* yang dimiliki *user*.	Setiap kali *user* membeli *avatar*.
7	*Avatar*	Entitias yang berisi data *avatar*.	Semua data *avatar* yang dimasukkan admin.
8	*Stase*	Entitas yang berisi data *stase* untuk setiap bab.	Semua data *stase* yang dimasukkan admin.
9	bab	Entitas yang berisi data bab dari stase.	Semua data bab dari stase yang dimasukkan admin.
10	*challenge*	Entitas yang berisi data *challenge* kasus yang ada.	Semua data *challenge* yang dimasukkan admin.
11	*Discussion*	Entitas yang berisi data grup diskusi.	Setiap kali *user* membuat grup diskusi baru.
12	*Comment*	Entitas yang beris data komentar setiap *user* pada masing-masing grup diskusi.	Setiap kali *user* berkomentar pada grup diskusi.

13	*History*	Entitas yang berisi data Riwayat *challenge* yang dikerjakan *user*.	Setiap kali *user* mengerjakan *challenge*.
14	Jawaban_penanganan	Entitas yang berisi jawaban penanganan dari *challenge* yang dikerjakan *user*.	Setiap kali *user* mengerjakan *challenge*.
15	Penanganan	Entitas yang berisi data penanganan *challenge*.	Semua data penanganan *challenge* yang dimasukkan admin.
16	Jawaban_penyakit	Entitas yang berisi jawaban penyakit dari *challenge* yang dikerjakan *user*.	Setiap kali *user* mengerjakan *challenge*.
17	Penyakit	Entitas yang berisi data penanganan *challenge*	Semua data penyakit *challenge* yang dimasukkan admin.
18	Kunci_penanganan	Entitas yang berisi data kunci jawaban penanganan dari *challenge*.	Semua data kunci jawaban penanganan setiap *challenge* yang dimasukkan admin.
19	Kunci_penyakit	Entitas yang berisi data kunci jawaban penyakit dari *challenge*.	Semua data kunci jawaban penyakit setiap *challenge* yang dimasukkan admin.
20	Investigasi	Entitas yang berisi investigasi atau riset yang telah dilakukan dokter terhadap kasus *challenge*.	Semua data investigasi *challenge* yang dimasukkan admin.

Kemudian berikut adalah tabel Identifikasi Tipe Relational yang memuat relasi antar tabel dalam aplikadi QuestMD:

Tabel 3. 5 Identifikasi Tipe Relational

Nama Entitas	Tipe Relasi	Nama Entitas	Deskripsi
user	1..*	*Leaderboard_value*	*User* dapat memiliki *leaderboard* berbeda setiap bulan.

	1..*	*My_achievement*	*User* dapat memiliki banyak *my_achievement*
	1..*	*Comment*	*User* dapat komentar lebih dari satu kali.
	1..*	*My_avatar*	*User* dapat memiliki banyak *my_avatar*
	1..*	Discussion	User dapat membuat lebih dari satu grup diskusi.
	1..*	history	User dapat memiliki banyak Riwayat pengerjaan *challenge*.
leaderboard	1..*	*Leaderboard_value*	*Leaderbboard* dapat memiliki banyak data *leaderboar_value*.
achievement	*..1	*My_achievement*	*Achievement* dapat dimiliki oleh banyak *my_achievement*.
discussion	1..*	*Comment*	*Discussion* dapat memiliki banyak komentar.
avatar	*..1	*My_avatar*	*Avatar* dapat dimiliki oleh banyak *my_avatar*.
challenge	*..1	*History*	*Challenge* dapat dimiliki oleh banyak *history*.
	1..*	*Discussion*	*Challenge* dapat memili banyak diskusi.
	1..*	Penanganan	*Challenge* dapat memiliki banyak penanganan sekaligus.
	1..*	Penyakit	*Challenge* dapat memiliki banyak penyakit.
	1..*	Investigasi	*Challenge* dapat memiliki banyak investigasi.
History	*..1	Jawaban_penanganan	*History* dapat dimiliki oleh banyak jawaban_penanganan..

	*..1	Jawaban_penyakit	*History* dapat dimiliki oleh banyak jawaban_penyakit..
penanganan	1..*	Jawaban_penanganan	Penanganan dapat memiliki banyak jawaban_penanganan.
	1..1	Kunci_penanganan	Penanganan hanya memiliki satu kunci_penanganan.
penyakit	1..*	Jawaban_penyakit	Penyakit hanya dapat memiliki banyak jawaban_penyakit.
	1..1	Kunci_penyakit	Penyakit hanya memiliki satu kunci_penyakit.

b. Desain Logikal *Database*

Pada proses logikal *database* kegiatan yang dilakukan yaitu melakukan identifikasi terhadap setiap atribut yang ada pada setiap tabel, mulai dari *primar key, foreign key,* dan atribut umum yang digunakan. Berikut adalah tabel atribut dari entitas yang ada:

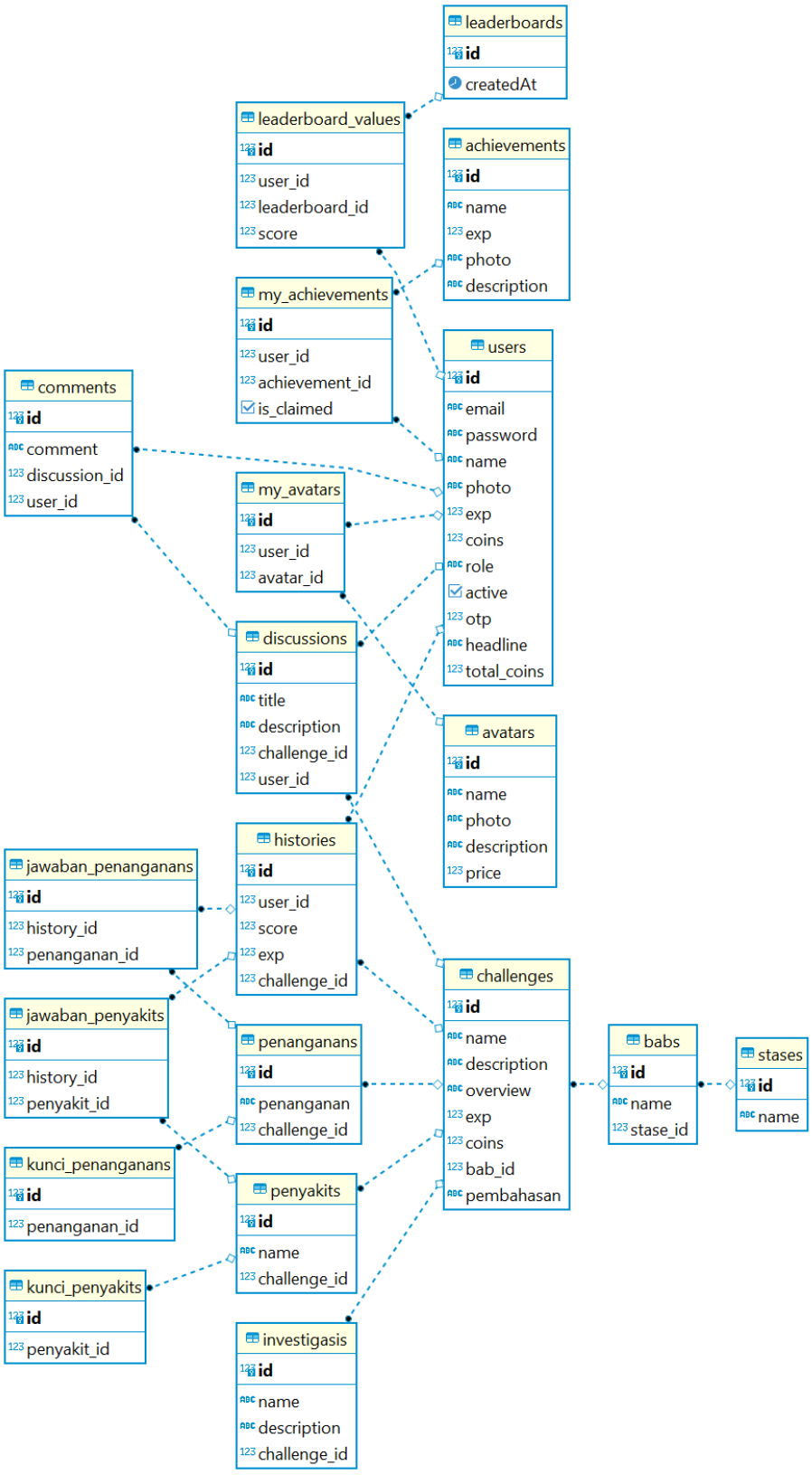
Tabel 3. 6 Identifikasi Atribut dan Kandidat *Key*

No	Nama Entitas	Atribut
1	*user*	*Id (PK)* *Email* *Password* *Name* *Photo* *Exp* *Coins* *Role* *Otp* *Headline* *Total_coins*
2	*Leaderboard_value*	*Id (PK)* *User_id (FK)* *Leaderboard_id (FK)*

		Score
3	*Leaderboard*	*Id (PK)* *createdAt*
4	*My_achievement*	*Id (PK)* *User_id (FK)* *Achievement_id (FK)*
5	*Achievement*	*Id (PK)* *Name* *Exp* *Photo* *Description*
6	*My_avatar*	*Id (PK)* *User_id (FK)* *Avatar_id (FK)*
7	*Avatar*	*Id (PK)* *Name* *Photo* *Description* *Price*
8	*Stase*	*Id (PK)* *Name*
9	*bab*	*Id (PK)* *Name* *Stase_id (FK)*
10	*challenge*	*Id (PK)* *Name* *Description* *Overview* *Exp* *Coins* *Bab_id (FK)* Pembahasan
11	*Discussion*	*Id (PK)* *Title* *Description* *Challenge_id (FK)* *User_id (FK)*

12	*Comment*	*Id (PK)* *Comment* *Discussion_id (FK)* *User_id (FK)*
13	*History*	*Id (PK)* *User_id (FK)* *Score* *Exp* *Challenge_id (FK)*
14	Jawaban_penanganan	*Id (PK)* *History_id (FK)* *Penanganan_id (FK)*
15	Penanganan	*Id (PK)* Penanganan *Challenge_id (FK)*
16	Jawaban_penyakit	*Id (PK)* *History_id (FK)* *Penyakit_id (FK).*
17	Penyakit	*Id (PK)* *name* *Challenge_id (FK)*
18	Kunci_penanganan	*Id (PK)* *Penanganan_id (FK)*
19	Kunci_penyakit	*Id (PK)* *Penyakit_id (FK)*
20	Investigasi	*Id (PK)* *Name* *Description* *Challenge_id (FK)*

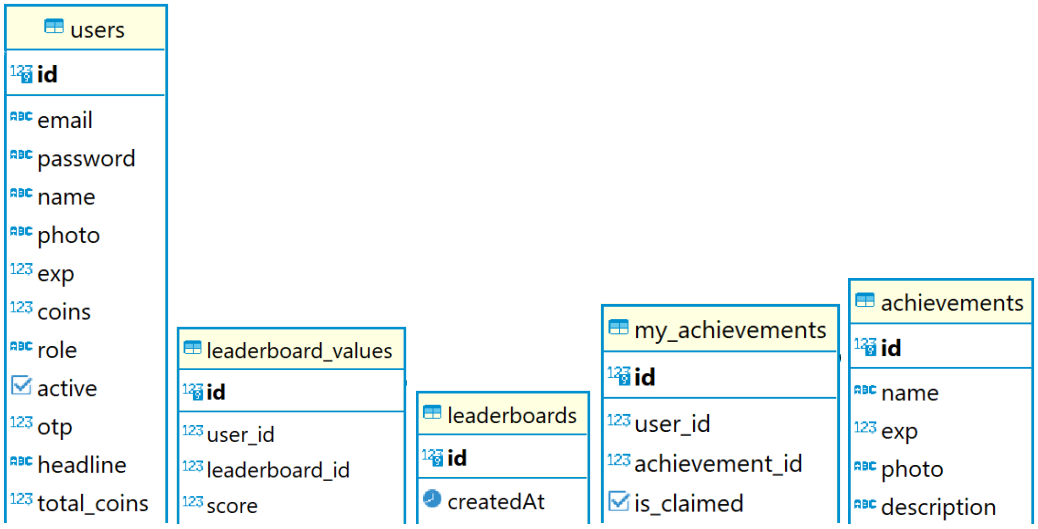
Pada Tabel 3.6 di atas, PK merupakan kunci utama (*primary key*) dan FK merupakan kunci asing (*foreign key*) yang digunakan sebagai penghubung antara relasi tabel yang ada. Hasil dari setiap tahapan sebelumnya *menghasilkan Entity-Relationship Diagram* (ERD). Berikut adalah ERD hasil dari perancangan *database* di atas:



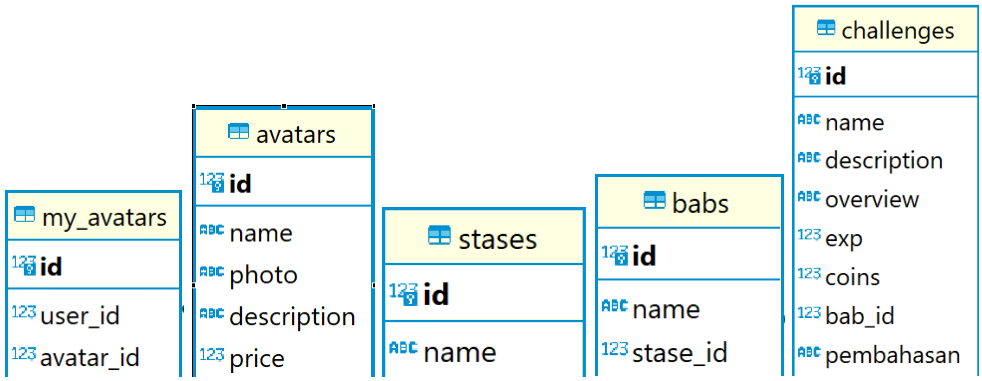
Gambar 3. 2 ERD Aplikasi QuestMD

c. Desain Fisik *Database*

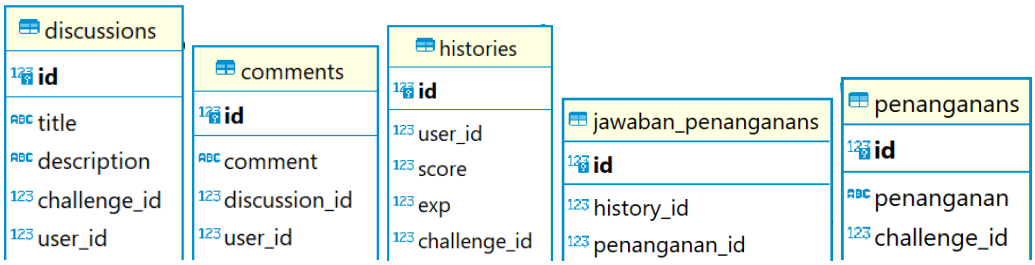
Setelah melakukan desain konseptual dan logika *database* proses selanjutnya yaitu melakukan desain fisik *database*, *database* yang digunakan sebagai desain fisik yaitu PostgreSQL, *database* dibuat dalam bentuk tabel-tabel dan belum berbentuk sempurna beserta relasi setiap tabel. Berikut adalah hasil desain fisik *database* dalam *database* PostgreSQL:



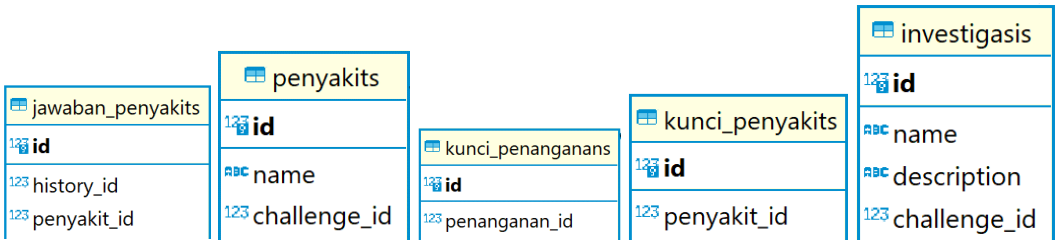
Gambar 3. 3 Tabel *user, leaderboard_value, leaderboard, my_achievement*, dan *achievement*



Gambar 3. 4 Tabel *my_avatar, avatar, stase, bab, challen*ge



Gambar 3. 5 Tabel *discussion, comment, historie, jawaban_penanganan,* dan *penanganan*



Gambar 3. 6 Tabel jawaban_penyakit, penyakit, kunci_penyakit, dan investigasi

3.3.3 *Implementation and Loading*

Tahapan Implementasi dan Pengisian (loading) melibatkan implementasi serta pengisian data yang dibutuhkan dari database yang telah dibuat. Implementasi pada tahapan ini mengacu pada hasil dari *Database life cycle* (DBLC). Pada tahapan ini, struktur fisik *database* yang telah dirancang direalisasikan ke dalam bentuk tabel dan hubungan antar tabel sesuai dengan rancangan yang telah disepakati. Proses implementasi ini dilakukan menggunakan Sistem Manajemen *database* (DBMS) dan pengisian data-data yang diperlukan ke dalam database.

Ada dua metode yang dapat digunakan untuk memasukkan data awal ke dalam database saat pembuatan aplikasi. Pertama, data dapat dimasukkan secara langsung menggunakan query insert. Alternatif kedua adalah melalui proses pengisian data awal (seeders) saat pembuatan API di bagian backend.

Dalam penelitian ini, data awal database dimasukkan melalui proses seeders. Pendekatan ini dipilih agar saat proses migrasi pada saat deployment atau saat perlu

menghapus seluruh data dalam database, tidak diperlukan banyak query insert. Hanya dengan menjalankan kode pada terminal, data dapat dimasukkan dengan efisien.

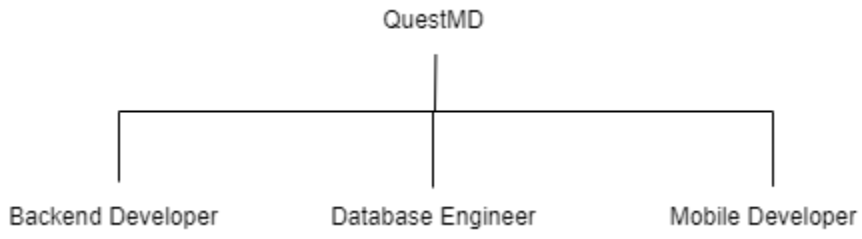
3.3.4 *Testing and Evaluation*

Tahap *Testing and Evaluation* yaitu database diuji apakah pembuatan database sudah sesuai atau tidak. Pengetesan dilakukan untuk mengetahui apakah *database* benar-benar sudah bisa digunakan adalah *insert, select, update,* dan *delete* terhadap tabel. Jika tabel sudah menampilkan data sesuai dengan isi dan nama tabel maka dapat dianggap bahwa implementasi dari metode *Database Life Cycle* (DBLC) telah berhasil dan dapat digunakan untuk pembuatan API di aplikasi sesuai hasil yang telah diperoleh.

3.3.5 *Operation*

Tahap selanjutnya yaitu Operation dengan diimplementasikannya *database* kedalam proyek asli seperti dalam pembuatan aplikasi pada bagian *backend* API dan melihat apakah *database* ada pembaruan pada saat pembuatan aplikasi. Dalam proses implementasi ke dalam *database* terkadang *database* mengalami perubahan-perubahan yang terjadi agar dapat menyesuaikan dengan fitur yang ada *database* diperbaiki sesuai dengan keadaan saat implementasi dilakukan tersebut.

Pada tahap ini pembuatan aplikasi dilakukan oleh orang yang berbeda, berikut adalah pembagian peran pada pembuatan aplikasi QuestMD yang dibuat:



Gambar 3. 7 Pembagian Peran Pembuatan Aplikasi

Pada gambar 3.7 terdapat tiga peran pada pembuatan aplikasi QuestMD yaitu Database Engineer yaitu Muhamad Satrio, Backend Developer yaitu Charles Gunawan, dan Mobile Developer yaitu Muhammad Wafa Al-Ausath. Sedangkan untuk tugas pada masing masing peran dapat dilihat pada tabel dibawah ini:

Tabel 3. 7 Pembagian Tugas Pembuatan Aplikasi

Peran	Nama	Tugas
Database Engineer	Muhamad Satrio	Membuat *database* sesuai dengan aturan DBLC (*database life cycle*) seperti desain kebutuhan hingga implementasi kedalam *database*, seperti PostgreSQL, MySQL, dan lainnya
Mobile Developer	Muhammad Wafa Al-Ausath	Membuat aplikasi android QuestMD secara keseluruhan dan menghubungkan ke API yang dibuat oleh *backend developer*
Backend Developer	Charles Gunawan	Membuat API seperti *login*, *logout*, dan CRUD (*Create*, *Read*, *Update*, *Delete*) lainnya sesuai fungsi dan permintaan dari *Mobile Developer*.

3.3.6 *Maintenance and Evolusion*

Tahap *maintenance and evolusion* merupakan tahap menjaga dan melaukan perbaikan pada *database* saat *database* telah digunakan, maksud dari tahap ini jika pada saat berjalannya waktu ternyata terdapat fitur baru yang ingin ditambahkan atau terdapat pembaruan sistem maka *database* bisa ikut diperbarui yang kemudian dapat berjalan dengan semestinya.

3.3.7 Analisis Kinerja

Pada tahap terakhir dari penelitian yaitu analisis kinerja masing-masing *database* yang digunakan mulai dari MySQL, PostgreSQL, dan MariaDB menggunakan Apache Jmeter yang merupakan *software* yang dapat digunakan untuk melakukan tes pada server. Testing ini diuji pada masing-masing *database* untuk mengetahui

response time setiap *database* pada setiap API yang diuji. Hasil dari testing ini dibandingkan dan akan diketahui pada setiap kondisi *database* yang tercepat dan paling cocok untuk digunakan di *runtime environtment* NodeJs *framework* express. Pengujian penelitian ini dilakukan dalam beberapa proses atau tahap yaitu:

- Pemasangan *software* Apache Jmeter, *database* (MySQL, PostgreSQL, MariaDB).
- Pemilihan dan perubahan API yang digunakan untuk pengujian.
- Melakukan perubahan *connection database*.
- Pembuatan data *dummy* sebanyak 100.000
- Melakukan *migration database*
- Pengujian *response time* pada tipe data yang berbeda di tabel *user*.
- Pengujian API dengan *query insert, create, read, update,* dan *delete* pada setiap *database*.

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang dilakukan, didapatkan beberapa kesimpulan seperti berikut:

1. Penelitian telah berhasil membuat *database* untuk aplikasi QuestMD dengan jumlah tabel sebanyak dua puluh enam dengan menggunakan metode pembuatan *database* DBLC (*Database life cycle*) dan dapat digunakan *Backend Developer*.
2. Berdasarkan hasil pengujian *response time* pada tipe data diketahui bahwa terdapat perbedaan kecepatan *response time* jika menggunakan tipe data yang berbeda, baik dari *database* PostgreSQL, MySQL, ataupun MariaDB.
3. Berdasarkan hasil pengujian *response time* API CRUD (*create, read, update*, dan *delete*) pada *database* PostgreSQL, MySQL, dan MariaDB diketahui untuk skenario *get* 10 dan 100 data *response time* paling cepat yaitu PostgreSQL dan kemudian MySQL dan MariaDB memiliki *response time* yang sama. Sedangkan, untuk skenario get 1000, 10.000, dan 100.000 data *response time* paling cepat yaitu MySQL dan disusul MariaDB baru kemudian PostgreSQL dengan *response time* paling lama.
4. Berdasarkan hasil pengujian *response time* pada *database* PostgreSQL, MySQL, dan MariaDB diketahui untuk skenario *insert, update, dan delete response time* paling cepat yaitu PostgreSQL dan disusul MariaDB kemudian terakhir MySQL.
5. Berdasarkan hasil keseluruhan pengujian *response time* API pada *database* PostgreSQL, MySQL, dan MariaDB secara umum PostgreSQL merupakan pilihan paling tepat untuk pengembangan API menggunakan *framework*

NodeJS Express pada aplikasi QuestMD dikarenakan pada aplikasi ini untuk operasi *get* hanya mengambil data dibawah 100 data, sehingga *database* yang tepat untuk digunakan yaitu PostgreSQL sesuai hasil pengujian.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan terdapat saran yang dapat diberikan untuk penelitian selanjutnya yaitu sebagai berikut:

1. Pengujian *response time database* dapat ditambahkan jenis *database* lain seperti NoSQL untuk mengetahui perbedaan kecepatan antara jenis *database* SQL dan NOSQL.
2. Pengujian *response time database* dapat diperluas dengan ditambahkan kondisi baru yang sering dipakai dalam pembuatan API seperti *query SUM, JOIN*, dan *query* lainnya.

DAFTAR PUSTAKA

[1] R. Sitanggang, "Sistem Informasi Laporan Penjualan Komputer Berbasis Lan," *Jurnal Mahajana Informasi*, vol. 4, pp. 62–77, Jun. 2019, doi: https://doi.org/10.51544/jurnalmi.v4i1.729.

[2] B. Nugroho, *Pandung Lengkap Menguasai Perintah SQL*. Jakarta Selatan: mediakita, 2008.

[3] W. N. Suliyanti, "Studi Literatur Basis Data SQL dan NoSQL," *KILAT*, vol. 8, no. 1, May 2019, doi: 10.33322/kilat.v8i1.460.

[4] A. Amarulloh, K. Kurniasih, and Muchlis, "Analisis Perbandingan Performa Web Service Rest Menggunakan Framework Laravel, Django, Dan Node Js Untuk Akses Data Dengan Aplikasi Website," *Jurnal Teknik Informatik*, vol. 9, Feb. 2023, doi: https://doi.org/10.51998/jti.v9i1.515.

[5] A. Munawir, "Pengaruh Kecepatan Transaksi Terhadap Kepuasan Pelanggan Pt Marga Mandala Sakti Pada Gerbang Tol Serang Timur," *Jurnal Ilmiah Sistem Informasi*, vol. 1, Jun. 2021, doi: https://doi.org/10.46306/sm.v1i1.5.

[6] F. Febriansyah and R. A. Awangga, *Membangun RestFul API dengan GO*, Pertama. Bandung: PT. Penerbit Buku Pedia, 2023.

[7] S. Surahman and E. B. Setiawan, "Aplikasi Mobile Driver Online Berbasis Android Untuk Perusahaan Rental Kendaraan," *Jurnal ULTIMA InfoSys*, vol. 8, no. 1, pp. 35–42, Aug. 2017, doi: 10.31937/si.v8i1.554.

[8] Hasanuddin, H. Asgar, and B. Hartono, "Rancang Bangun Rest API Aplikasi Weshare Sebagai Upaya Mempermudah Pelayanan Donasi Kemanusiaan," *Jurnal Informatika Teknologi dan Sains*, vol. 4, no. 1, pp. 8–14, Feb. 2022, doi: 10.51401/jinteks.v4i1.1474.

[9] D. V Kornienko, S. V Mishina, S. V Shcherbatykh, and M. O. Melnikov, “Principles of securing RESTful API web services developed with python frameworks,” *J Phys Conf Ser*, vol. 2094, no. 3, p. 032016, Nov. 2021, doi: 10.1088/1742-6596/2094/3/032016.

[10] Y. Supardi, *Semua Bisa Menjadi Programmer Javascript & Node.JS*. Jakarta: PT Elex Media Komputindo, 2020.

[11] A. Firdaus, S. Widodo, A. Sutrisman, and R. Mardiana, “Rancang Bangun Sistem Informasi Perpustakaan Menggunakan Web Service Pada Jurusan Teknik Komputer Polsri,” *Jurnal Informanika*, vol. 5, Aug. 2019, doi: https://doi.org/10.52233/informanika.v5i2.99.

[12] Sholihin, Nurjaya, and M. Ardhiansyah, *Membangun Web dengan Framework Laravel 8*. Tangerang Selatan: Pascal Books, 2021.

[13] D. J. V. Siahaan, M. Putri, R. Andarsyah, and R. M. Awangga, *Cara Praktis Membuat Chatbot Whatsapp*. Bandung: Penerbit Buku Pedia, 2023.

[14] Vijay P. Mehta, *Pro LINQ Object Relational Mapping with C# 2008*. New York: Apress, 2008.

[15] I. Kurniawan, Humaira, and F. Rozi, “REST API Menggunakan NodeJS pada Aplikasi Transaksi Jasa Elektronik Berbasis Android,” *JITSI : Jurnal Ilmiah Teknologi Sistem Informasi*, vol. 1, no. 4, pp. 127–132, Dec. 2020, doi: 10.30630/jitsi.1.4.18.

[16] M. K. Pradana, A. Andrianto, and Y. A. Auliya, “Pengembangan Sistem Informasi Desa Terpadu Menggunakan Metode Rapid Application Development (RAD) Studi Kasus Desa Arjasa,” *INFORMAL: Informatics Journal*, vol. 7, no. 2, p. 64, Aug. 2022, doi: 10.19184/isj.v7i2.25238.

[17] Jubile Enterprise, *MySQL Untuk Pemula*. Jakarta: PT Elex Media Komputindo, 2014.

[18] Sianipar, *Pemrograman Database Menggunakan MySQL*. Yogyakarta: CV Andi Offset, 2015.

[19] R. B. Sentosa, “Membangun Web Konten Manajemen Sistem Secara Dinamis Dengan Bahasa Pemrograman PHP Framework CodeIgniter Dengan Database MariaDB,” *INTECOMS: Journal of Information Technology and Computer Science*, vol. 1, no. 2, pp. 212–223, Aug. 2018, doi: 10.31539/intecoms.v1i2.295.

[20] M. R. Faisal, *ASP.NET Core MVC & PostgreSQL dengan Visual Studio Code*. M Reza Faisal, 2017.

[21] A. D. Praba, “Aplikasi Rekap Mengajar Berbasis Webiste Dengan Database PostgreSQL,” *Indonesian Journal on Networking and Security*, vol. 8, pp. 27–31, 2018, doi: http://dx.doi.org/10.55181/ijns.v8i1.1571.

[22] D. I. Permatasari, “Pengujian Aplikasi menggunakan metode Load Testing dengan Apache JMeter pada Sistem Informasi Pertanian,” *Jurnal Sistem dan Teknologi Informasi (JUSTIN)*, vol. 8, no. 1, p. 135, Jan. 2020, doi: 10.26418/justin.v8i1.34452.

[23] Z. Efendy, “Normalisasi Dalam Desain Database,” *Jurnal CoreIT*, vol. 4, Jun. 2018.

[24] C. Coronel, S. Morris, and P. Rob, *Database Systems: Design, Implementation, and Management*, Ninth Edition. Boston: Joe Sabatino, 2011.

[25] S. saha Bagui and R. Earp, *Database Design Using Entity-Relationship Diagrams*, Third Edition. Boca Raton: CRC Press, 2022.

[26] S. M. Pulungan, R. Febrianti, T. Lestari, N. Gurning, and N. Fitriana, “Analisis Teknik Entity-Relationship Diagram Dalam Perancangan Database,” *Jurnal Ekonomi Manajemen dan Bisnis (JEMB)*, vol. 1, no. 2, pp. 98–102, Feb. 2023, doi: 10.47233/jemb.v1i2.533.

[27] A. K. Kurniawan, E. S. Pramukantoro, and P. H. Trisnawan, “Perbandingan Kinerja Cassandra dan MongoDB Sebagai Backend IoT Data Storage,” *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, vol. 3, pp. 364–371, Aug. 2018.

[28] C. Asiminidis, G. Kokkonis, and S. Kontogiannis, “Database Systems Performance Evaluation for IoT Applications,” *International Journal of Database Management Systems*, vol. 10, no. 06, pp. 01–14, Dec. 2018, doi: 10.5121/ijdms.2018.10601.

[29] A. D. Praba and M. Safitri, “Studi Perbandingan Performansi Antara Mysql Dan Postgresql,” *Jurnal Khatulistiwa Informatika*, vol. 8, no. 2, Dec. 2020, doi: 10.31294/jki.v8i2.8851.

[30] I. Warman and R. Ramdaniansyah, “Analisis Perbandingan Kinerja Query Database Management System (DBMS) Antara MySQL 5.7.16 Dan MariaDB 10.1,” *JURNAL TEKNOIF*, vol. 6, no. 1, pp. 32–41, Apr. 2018, doi: 10.21063/JTIF.2018.V6.1.32-41.

[31] M. Min, “Experiments of Search Query Performance for SQL-Based Open Source Databases,” *International Journal of Internet, Broadcasting, and Communication*, vol. 10, pp. 31–38, 2018, doi: 10.7236/IJIBC.2018.10.2.6.

[32] S. Budiman, F. Fadhila, V. A. Saputro, E. Utami, and K. Khusnawi, “Perbandingan Performa SQL dan NoSQL Dengan PHP Pada 5 Juta Data,” *IJCIT (Indonesian Journal on Computer and Information Technology)*, vol. 6, no. 1, May 2021, doi: 10.31294/ijcit.v6i1.9692.

[33] F. A. Nugraha and Y. A. Susetyo, “Analisis Perbandingan Performa Database DuckDB Dan Sqlite Pada Pengolahan Big Data,” *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. 8, no. 3, pp. 1052–1060, Aug. 2023, doi: 10.29100/jipi.v8i3.4032.

[34] Sugiarto and E. Triandini, “Pengembangan Database E-commerce De Janggelan Menggunakan Metode Database Life Cycle,” *Jurnal Sistem Dan Informatika (JSI)*, vol. 16, pp. 122–132, Jun. 2022, doi: https://doi.org/10.30864/jsi.v16i2.478.

[35] S. S. Wibagso and E. Lia, “Desain Model Database Layanan Panti Werdha dengan Menerapkan Metode Database Life Cycle,” *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 6, no. 3, Dec. 2020, doi: 10.28932/jutisi.v6i3.3047.

[36] M. R. Anwar and S. Purnama, “Boarding House Search Information System Database Design,” *International Journal of Cyber and IT Service Management*, vol. 2, no. 1, pp. 70–81, Mar. 2022, doi: 10.34306/ijcitsm.v2i1.89.

[37] E. Lutfina, R. O. C. Setiawan, A. Nugroho, and M. Z. Abdillah, “Perancangan Aplikasi Pembelajaran Dengan Konsep Gamifikasi Systematic Literature Review,” *METHOMIKA Jurnal Manajemen Informatika dan Komputerisasi Akuntansi*, vol. 7, no. 1, pp. 78–87, Apr. 2023, doi: 10.46880/jmika.Vol7No1.pp78-87.

[38] D. Ariani, “Gamifikasi untuk Pembelajaran,” *Jurnal Pembelajaran Inovatif*, vol. 3, no. 2, pp. 144–149, Nov. 2020, doi: 10.21009/JPI.032.09.

[39] A. N. Hakim, A. Kusumawati, Y. B. H. Sakti, and I. Qoimatun, “Tingkat Stres dan Pencapaian Kompetensi Mahasiswa Program Profesi Dokter: Penelitian Kuantitatif dan Kualitatif,” *Jurnal Kedokteran dan Kesehatan*, vol. 19, no. 2, p. 173, Aug. 2023, doi: 10.24853/jkk.19.2.173-186.