BAB II
TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Berikut adalah beberapa penelitian yang telah dilakukan oleh peneliti sebelumnya yang relevan dengan penelitian yang akan dilaksanakan.

1. Penelitian dengan judul "Optimisasi Web Service REST API Menggunakan Load Balancer dan Cache dengan Algoritma Round Robin (Studi Kasus: Madani Infosphere)"

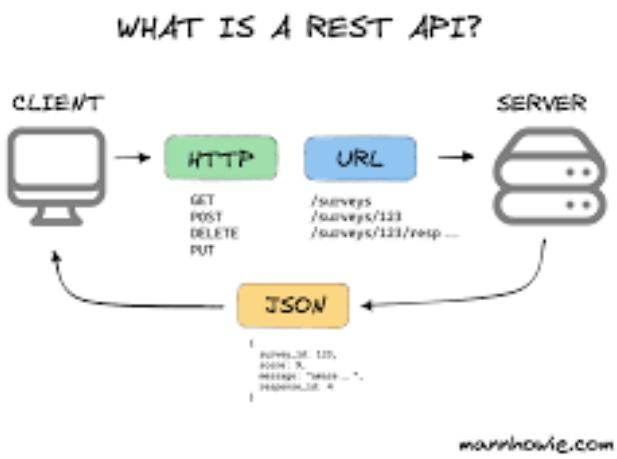
   Penelitian dengan judul "Optimisasi Web Service REST API Menggunakan Load Balancer dan Cache dengan Algoritma Round Robin (Studi Kasus: Madani Infosphere)" yang dilakukan oleh F. Z. Junaedy, da U. Surapati berfokus pada optimasi REST API dengan menggunakan metode *load balancing* dan *caching*. Tujuan pada penelitian ini adalah untuk meningkatkan kinerja web service REST API dengan memanfaatkan dua metode, yaitu *load balancing* dan *caching*. *Load balancing* diterapkan untuk mendistribusikan beban kerja secara merata ke beberapa server menggunakan algoritma Round Robin, sementara *caching* dilakukan untuk menyimpan data yang sering diakses agar dapat mengurangi waktu respon pada REST API. Untuk mengukur performa REST API pada penelitian ini menggunakan *load test* sebagai pengujiannya dengan menggunakan parameter *request per second* da waktu respon. Hasil dari penelitian ini menunjukkan bahwa *load balancing* dengan algoritma round robin dapat mendistribusikan beban kerja pada server secara merata. Sedangkan *caching* berhasil mengurangi beban kerja pada server dengan menyimpan data yang sering digunakan pada *cache*. Dengan demikian, server tidak perlu memproses permintaan yang sama berulang kali, sehingga waktu respon pada REST API dapat dipersingkat dan efisiensi kinerja sistem dapat meningkat.
2. Penelitian dengan judul "Performance Optimisation of Web Applications using In-memory Caching and Asynchronous Job Queues"

Pada penelitian yang dilakukan oleh Sidharth S Prakash dan Binsu C Kovoor bertujuan untuk mengoptimalkan performa web aplikasi dengan 2 cara. Cara pertama adalah dengan menggunakan *in-memory caching* yang dirancang untuk mempercepat proses pengambilan data dari database melalui penggunaan data yang telah disimpan di *in-memory qcache*. Cara kedua berfokus pada pengurangan keterlambatan waktu respon yang dirasakan oleh pengguna dengan memindahkan tugas-tugas berat yang memakan waktu ke latar belakang, sehingga tidak mengganggu *main user interface thread*. Penelitian ini juga menganalisis kinerja *in-memory caching* dan *asynchronous job queues* dalam berbagai skenario *concurrent*. Hasil penelitian ini menunjukkan bahwa *in-memory cache* dapat meningkatkan performa aplikasi ketika terdapat *query* yang berat pada aplikasi yang membutuhkan banyak proses dan tugas yang memakan waktu. Semakin banyak pengguna yang megakses permintaan semakin besar pula pengurangan waktu respon. Namun performa *caching* pada *in-memory cache* sangat bergantung dengan kapasitas memori pada server. Sedangkan metode *queue* dapat meningkatkan performa ketika ada tugas-tugas yang memerlukan waktu Panjang karena tugas-tugas tersebut dapat dijalankan di latar belakang aplikasi. Semakin lama waktu yang dibutuhkan untuk menyelesaikan tugas, semakin besar pula pengurangan waktu yang dirasakan pengguna saat menggunakan metode *queue*.

2.2. REST API

REST API (*Representational State Transfer Application Programming Interface*) adalah antarmuka yang digunakan untuk komunikasi antara sistem *frontend* dan *backend service*. REST API digunakan untuk komunikasi antar klien dan server dalam suatu aplikasi dengan memanfaatkan protokol HTTP [16]. Dengan menggunakan REST API sistem dapat saling berinteraksi dengan mengirim dan menerima data dalam format JSON atau XML. Pada penelitian ini REST API pada *backend service* aplikasi Jadi Juara digunakan sebagai studi kasus untuk dilakukan optimasi. REST API aplikasi Jadi Juara berperang penting sebagai komunikasi antara *backend service* dengan aplikasi *mobile* Jadi Juara. Dengan

menggunakan REST API pengolahan data pada database tidak akan bergantung pada platform tertentu.

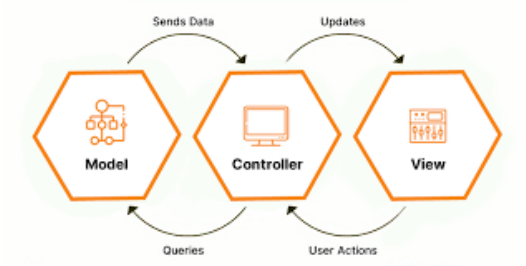


Gambar 2. 1 Arsitektur REST API

Sumber: Dokumentasi dari website manhowie.com, diakses dari https://mannhowie.com/rest-api pada 31 Desember 2024.

REST API menggunakan metode HTTP dengan method GET, POST, PUT, DELETE untuk memanggil fungsi yang ada pada *backend service*. Setiap method diwakili oleh URL yang unik atau dapat disebut dengan *endpoint*. Dengan menggunakan REST API aplikasi dapat diintegrasikan dengan mudah oleh beberapa *platform*.

2.3. Laravel

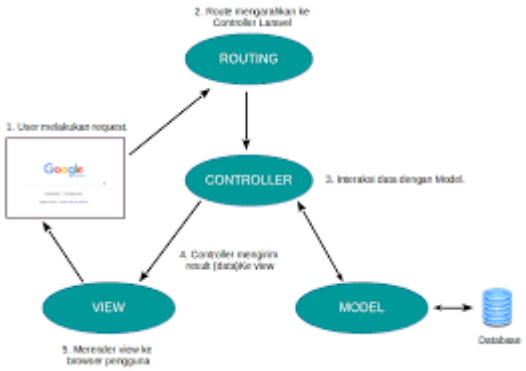


Gambar 2. 2 Arsitektur MVC

Sumber: Dokumentasi dari website fkrihnif.medium.com diakses dari https://fkrihnif.medium.com/understanding-the-mvc-architecture-in-laravel-a-comprehensive-guide-8f620cc139b6 pada 31 Desember 2024

Aplikasi Jadi Juara saat ini menggunakan Laravel sebagai *backend service* untuk pengelolaan data dan logika bisnis pada aplikasi Jadi Juara. Laravel merupakan framework yang ditulis menggunakan bahasa pemrograman PHP. Framework ini menggunakan arsitektur MVC (Model-View-Controller). Gambar 2.2 merupakan visualisasi dari arsitektur MVC, dimana model berperan untuk menangani semua interaksi dengan database. Pada REST API aplikasi Juara model merupakan representasi dari tabel yang ada pada database dan relasinya. Sedangkan view berperan untuk menampilkan data kepada pengguna. Terakhir yaitu controller yang berperan sebagai penghubung antara view dan model. Controller juga berperan sebagai logika bisnis pada aplikasi.

Prinsip routing juga digunakan Laravel untuk mengelola dan mendifinisikan URL yang akan mengarah ke fungsi pada controller. Pada REST API Jadi Juara penamaan URL pada routing merepresentasikan setiap *endpoint*. URL ini mendukung berbagai metode HTTP seperti GET, POST, PUT, dan DELETE. Gambar 2.3 merupakan cara kerja dari arsitektur pada Laravel dengan menggunakan routing.



Gambar 2. 3 Arsitektur MVC dengan Routing

Sumber: Dokumentasi dari website sulhi.id diakses dari https://sulhi.id/konsep-mvc-model-view-controller-pada-laravel/ pada 31 Desember 2024

2.4. Load Testing

Load testing merupakan salah satu metode pengujian yang digunakan untuk mengevaluasi seberapa baik sistem dapat menangani beban atau jumlah pengguna yang meningkat. Pada penelitian ini *load testing* digunakan untuk mengukur

kemampuan REST API pada aplikasi Jadi Juara. Tujuan utama dari *load testing* adalah untuk mengukur kapasitas sistem dalam merespons permintaan yang tinggi, baik dari segi waktu respon dan ketahanan sistem saat berada di bawah beban berat. Dengan menggunakan *load testing*, peniliti dapat mengidentifikasi titik-titik lemah dalam REST API aplikasi Jadi Juara dan memastikan bahwa aplikasi Jadi Juara dapat beroperasi dengan ketika jumlah penggunanya meningkat secara signifikan. Menurut [14] *load testing* memiliki dua metrik utama yaitu:

1) *Request per Second*

   metrik ini mengukur jumlah permintaan yang dapat diproses oleh server setiap detik, yang merepresentasikan tingkat throughput. Semakin tinggi nilainya, semakin banyak permintaan yang dapat ditangani server dalam waktu tertentu, mencerminkan kinerja yang lebih baik saat menghadapi beban kerja.

$$Request\ per\ Second = \frac{Complete\ Request}{Time\ taken\ for\ tests}$$

2) *Time per Request*

   Metrik ini menghitung rata-rata waktu yang diperlukan untuk memproses satu permintaan, sehingga memberikan gambaran mengenai tingkat latensi.

$$Time\ per\ Request = \frac{Time\ taken\ for\ tests}{Complete\ Requests}$$

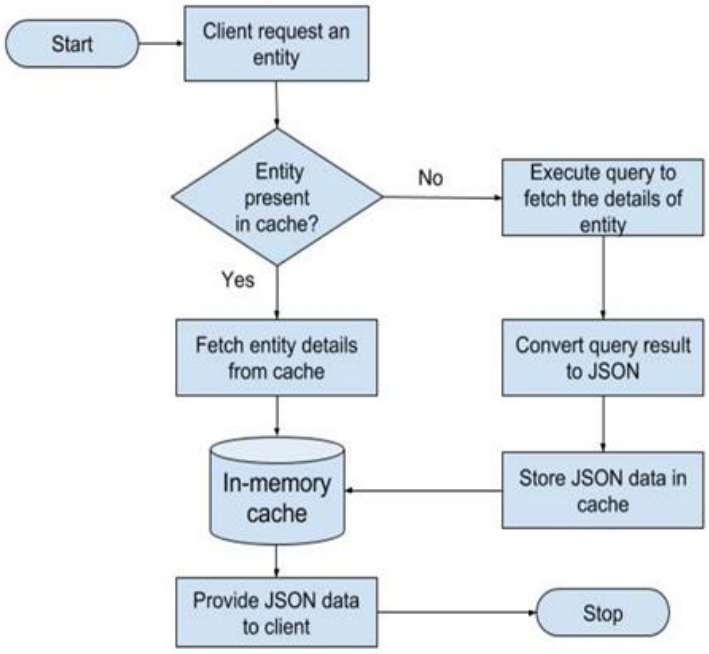
Request per Second saling berkaitan dengan *Time per Request*, karena metrik ini memberikan gambaran tentang seberapa cepat server dapat merespons setiap permintaan yang diterima.

Pemantauan penggunaan CPU dan Memori pada *load testing* sangat diperlukan, karena aplikasi dengan kinerja yang baik adalah aplikasi yang mampu menangani sejumlah besar permintaan dari klien tanpa menggunakan terlalu banyak sumber daya pada perangkat server, sehingga dapat mengurangi biaya operasional yang harus ditanggung oleh pengelola aplikasi [17]. Sehingga parameter *load testing* yang digunakan pada penelitian ini meliputi *request per second, time per request,* penggunaan CPU dan penggunaan memori.

2.5. K6

K6 digunakan sebagai *tool* untuk menguji REST API pada *load testing*. K6 merupakan sebuah *tool open source* yang dirancang untuk melakukan *load testing* dan *performance testing*. K6 menggunakan bahasa pemrograman javascript untuk menulis scenario pengujian [18]. Pada penelitian ini javascript ditulis dengan berabagai kondisi beban menurut *test* case yang telah dibuat. Dengan menggunakan K6 REST API aplikasi Jadi Juara dapat diukur performanya apabila terjadi lonjakan pengguna secara bersamaan. Dokumentasi K6 dapat dilihat melalui (https://k6.io/)

2.6. In-Memory Caching



Gambar 2. 4 Algoritma *In-Memory Caching*

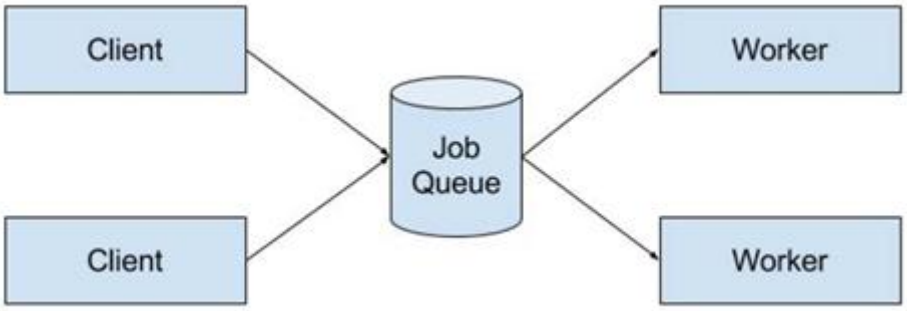
Sumber: Jurnal dari [5] *diakses pada 31 Desember 2024*

Salah satu metode untuk optimasi REST API pada peneletian ini yaitu *In-memory cache. In-memory caching* adalah metode penyimpanan data sementara yang dilakukan di memori utama (RAM) untuk mempercepat pengambilan data dan meningkatkan performa aplikasi. Gambar 2.4 merupakan flowchart dari algoritma pada metode *in-memory cache*. Metode ini bekerja dengan cara menyimpan data

yang sering diakses atau hasil dari operasi berat seperti pada *query* SQL, sehingga server tidak perlu memproses permintaan ke database secara berulang. Alur penggunaan *in-memory cache* pertama yaitu memeriksa apakah data yang diminta sudah ada di dalam memoru. Jika data ditemukan pada memori server secara langsung memberikan respon ke klien tanpa melakukan query database. Namun, jika data tidak ditemukan pada memori, server akan mengambil data dari database kemudian menyimpannya ke dalam memori, dan kemudian mengirimkannya ke klien [5]. Keunggulan utama *in-memory cache* yaitu dapat mengurangi latensi dan meningkatkan performa server yang lebih baik karena data dapat diakses langsung dari RAM yang lebih cepat dibandingkan akses ke disk atau database. Meskipun demikian, metode ini memiliki keterbatasan, seperti kapasitas memori yang terbatas, data yang tidak persisten, dan perlunya mekanisme untuk mengelola data agar data yang diberikan ke pengguna tidak usang. Pada penelitian ini implementasi *in-memory cache* menggunakan sistem penyimpanan key-value dengan Redis. Dengan menggunakan *in-memory cache* diharapkan performa REST API pada aplikasi Jadi Juara dapat meningkat.

2.7. Queue

Untuk menangani proses yang berat pada REST API aplikasi Jadi Juara dibutuhkan penggunaan *Queue* karena dapat meningkatkan performa REST API. *Queue* digunakan oleh background task untuk memproses tugas secara asinkron. Metode ini memungkinkan pemrosesan tugas berat dilakukan di latar belakang tanpa mengganggu interaksi pengguna dengan antarmuka utama aplikasi. Dengan menggunakan *queue*, tugas-tugas yang memakan waktu lama seperti pengiriman email dapat dipindahkan ke latar belakang, sementara pengguna tetap dapat melanjutkan aktivitasnya tanpa gangguan.



Gambar 2. 5 Interaksi Asinkron *Client* dan *Worker*

Sumber: Jurnal dari [5] *diakses pada 31 Desember 2024*

Gambar 2.5 merupakan gambar interaksi secara asinkron dari klien dan *worker*. Ketika tugas berat dimulai, sistem akan menempatkan tugas tersebut ke dalam *job queue*, sebuah antrian yang menyimpan daftar tugas yang menunggu untuk diproses. *Worker* atau *task runner* kemudian secara rutin mengambil tugas dari antrian tersebut dan menjalankannya di latar belakang, tanpa membebani thread utama aplikasi. Setelah proses selesai, hasilnya dapat disimpan atau diteruskan sesuai kebutuhan sistem. Penggunaan *queue* menawarkan berbagai manfaat, seperti menjaga antarmuka pengguna tetap responsif, mengurangi waktu tunggu yang dirasakan oleh pengguna, serta meningkatkan skalabilitas sistem dengan mendistribusikan tugas ke beberapa worker secara parallel [5]. Dalam penelitian ini, *queue* diterapkan untuk menangani tugas-tugas berat, seperti pengolahan data dalam jumlah besar, sehingga secara signifikan meningkatkan kinerja sistem aplikasi Jadi Juara, terutama saat terdapat banyak permintaan yang harus diproses secara bersamaan.

2.8. Redis

Dalam implementasi *queue* dan *in-memory caching* Redis dipilih sebagai teknologi untuk mengimplementasikan performa REST API. Alasan menggunakan Redis pada penelitian ini yaitu Redis terpilih sebagai database yang paling digemari selama lima tahun berturut-turt [19]. Redis, atau Remote Dictionary Server, adalah database in-memory key-value store yang bersifat open-source dan dirancang untuk memberikan performa tinggi dengan kecepatan akses yang sangat cepat. Data pada

Redis disimpan langsung di memori utama (RAM), sehingga baik untuk berbagai skenario seperti *caching*, penyimpanan sesi, *message broker*, dan aplikasi *real-time*. Redis mendukung berbagai struktur data, termasuk string, list, set, sorted set, hash, dan stream, sehingga memberikan fleksibilitas dalam menyimpan data sesuai kebutuhan aplikasi. Selain itu, Redis juga mendukung replikasi *master-slave*, *clustering*, dan partisi data, menjadikannya sangat skalabel untuk aplikasi berskala besar. Dengan operasi yang bersifat atomik, Redis memastikan konsistensi data, sementara fitur persistensi opsional memungkinkan penyimpanan data ke disk jika diperlukan. Meskipun memiliki keunggulan dalam kecepatan dan fleksibilitas, Redis memiliki keterbatasan kapasitas karena penyimpanan berbasis RAM, sehingga lebih cocok untuk data yang sering diakses atau digunakan. Dengan kemampuannya yang andal dan komunitas open-source yang aktif, Redis menjadi pilihan populer untuk pengembangan aplikasi modern yang membutuhkan kinerja tinggi dan latensi rendah. Dokumentasi dari redis dapat dilihat pada link (https://redis.io/).

2.9. FrankenPHP

Pada penelitian ini digunakan FrankenPHP sebagai web server untuk optimasi REST API aplikasi Jadi Juara. Alasan penggunaan FrankenPHP yaitu karena menurut [9] FrankenPHP dengan konfigurasi *worker mode* memiliki performa yang paling baik diantara web server PHP yang lainnya. FrankenPHP adalah server aplikasi PHP modern yang dibangun di atas web server Caddy [20]. FrankenPHP tidak hanya berfungsi sebagai server aplikasi, tetapi juga menyediakan pustaka Go yang memungkinkan pengembang untuk menanamkan interpreter PHP ke dalam aplikasi Go mereka. Web server ini dilengkapi dukungan HTTP/2 dan HTTP/3, serta *worker mode* untuk menciptakan performa tinggi. FrankenPHP kompatibel dengan berbagai framework PHP seperti Symfony dan Laravel. Dengan menggunakan sintaks Caddyfile yang sederhana, pengembang dapat dengan mudah mengkonfigurasi server sesuai kebutuhan mereka. Keunggulan lainnya termasuk dukungan HTTPS otomatis dan kemampuan kontainerisasi dengan Docker, menjadikan FrankenPHP sebagai platform yang ideal untuk deployment aplikasi

PHP dalam lingkungan modern. Dokumentasi FrankenPHP dapat dilihat pada (https://frankenphp.dev/).

2.10. Docker

Pada penelitian ini, Docker digunakan sebagai lingkungan kerja untuk mendukung proses implementasi dan pengujian berbagai metode optimasi pada REST API aplikasi Jadi Juara. Docker merupakan aplikasi berbasis teknologi *open source* yang mendukung pengembang maupun pengguna lainnya untuk membangun, menjalankan, menguji, dan mendistribusikan aplikasi di dalam sebuah container [21]. Pada container ini pengembang dapat mengemas aplikasi beserta semua dependensi dan konfigurasi yang diperlukan ke dalam satu paket untuk dijalankan atau dirubah.

Menurut penelitian [22] docker merupakan salah satu solusi standar untuk mengatasi salah satu bagian paling mahal dalam siklus pengembangan perangkat lunak, yaitu proses deployment. Saat proses deployment atau running program, Docker menjalankan container berdasarkan base image menggunakan sistem file berlapis (file system as a layer). Dalam sistem ini, Docker hanya menyalin lapisan perubahan saja sehingga dapat menjalankan container yang berbeda meskipun menggunakan base image yang sama [21].

Keunggulan utama Docker terletak pada isolasi yang diberikannya, sehingga setiap kontainer berjalan secara independen tanpa saling memengaruhi, bahkan jika mereka menggunakan teknologi atau konfigurasi yang berbeda. Hal ini sangat bermanfaat dalam pengembangan dan pengujian pada penelitian ini karena peneliti dapat menciptakan lingkungan kerja yang stabil dan seragam.

2.11. Prometheus

Prometheus merupakan salah satu perangkat lunak *open source* yang digunakan dalam dunia monitoring dan peringatan sistem [23]. Alat ini dikembangkan dengan tujuan utama untuk mengumpulkan, merekam, dan menganalisis data metrik dari berbagai jenis sistem dan layanan, baik itu server, aplikasi, hingga container. Pada penelitian ini Prometheus digunakan untuk

memantau data penggunaan CPU dan penggunaan Memori *virtual privater server* pada saat *load testing* dilakukan.

2.12. Grafana

Grafana merupakan perangkat lunak open source yang digunakan untuk menampilkan data metrik dalam bentuk grafik atau teks secara visual dan informatif [24]. Grafana mengubah data dalam bentuk time-series menjadi visualisasi yang informatif [23]. Platform ini mendukung berbagai jenis sumber data, sehingga menjadi alat yang fleksibel untuk membuat dashboard yang interaktif dan dapat disesuaikan. Dengan menggunakan Grafana metrik, log, dan data lainnya secara real-time dapat divisualisasikan, sehingga memudahkan pemahaman yang lebih mendalam terhadap sistem yang dikelola. Pada penelitian ini Grafana digunakan untuk menampilkan visualisasi dalam penggunaan CPU dan Memory saat dilakukan *load testing*.

2.13. Aplikasi Jadi Juara

Pada penelitian ini REST API aplikasi Jadi Juara digunakan sebagai studi kasus optimasi REST API. Jadi Juara adalah aplikasi pembelajaran yang dirancang khusus untuk siswa Bimbingan Belajar Himalaya. Aplikasi ini memiliki beberapa fitur seperti materi pembelajaran, ujian online, berita terkait Pendidikan dan event atau lomba yang yang dapat diikuti oleh siswa. Studi kasus ini sangat relevan pada penelitian ini karena aplikasi Jadi Juara pernah mengalami *downtime* saat mengalami lonjakan pengguna. Akibat dari lonjakan pengguna dapat menyebabkan beban yang meningkat pada server sehingga dapat memengaruhi performa aplikasi, seperti meningkatnya response time, menurunnya throughput, atau bahkan ketidakmampuan server untuk menangani permintaan tambahan. Permasalahan ini menunjukkan bahwa aplikasi Jadi Juara memerlukan optimasi REST API agar dapat memberikan pelayanan yang baik saat siswa menggunakan aplikasi.

Halaman ini sengaja dikosongkan