BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan hasil yang diperoleh dari penelitian ini, tujuan utama dari penelitian adalah untuk menjawab rumusan masalah yang telah disampaikan pada Bab 1. Fokus penelitian mencakup optimasi REST API Laravel dengan metode *in-memory cache* dan *queue* serta penggunaan FrankenPHP sebagai pengganti web server untuk meningkatkan performa REST API. Berdasarkan proses pengujian yang telah dilakukan, berikut disajikan kesimpulan dari penelitian ini:

1. Optimasi REST API dilakukan dengan menggunakan metode *queue*, *in-memory cache* dan penggantian web server ke FrankenPHP pada REST API aplikasi Jadi Juara. Implementasi dilakukan dengan menerapkan metode *queue* dan *in-memory cache* menggunakan Redis sebagai penyimpanan data pada memori. Metode *queue* digunakan pada endpoint REST API yang menggunakan pihak ketiga dan yang dapat dijalankan secara asinkron seperti pada endpoint `/register-student` dan `/finish-exam`. Kemudian *in-memory cache* diimplementasikan pada REST API Laravel yang sering diakses oleh pengguna seperti `/top-3-banner`. Selanjutnya FrankenPHP digunakan sebagai pengganti web server yang lama yaitu Nginx dan PHP-FPM. Terakhir yaitu proses optimasi dilakukan dengan menggabungkan penggunaan FrankenPHP dan Redis sebagai implementasi metode *queue* dan *in-memory cache*.
2. Implementasi penggunaan *in-memory cache*, *queue*, FrankenPHP, dan kombinasi FrankenPHP dengan metode *in-memory cache* serta *queue* menunjukkan peningkatan performa pada REST API. Pada endpoint yang diimplementasikan metode *queue* saja menunjukkan bahwa *request per second* (RPS) meningkat dari RPS sekitar 2,6/s menjadi 26/s – 30/s (kenaikan sebesar 905%–1039%). Pada penggunaan FrankenPHP saja RPS terjadi penurunan sekitar 15%. Namun ketika FrankenPHP dan metode *queue* digabungkan, terjadi peningkatan RPS sebesar 30/s – 50/s (kenaikan

sebesar 1407%–1778%), atau sekitar 50%–68% lebih tinggi dibandingkan hanya menggunakan *queue* dengan web server Nginx+PHP-FPM. Dengan peningkatan RPS ini menunjukkan bahwa *request* data pada REST API dapat ditangani dengan banyak sehingga *time per request* dapat menurun. *Time per request* saat diimplementasi metode *queue* menurunkan waktu respon sebesar 85%–86,66%, sementara ketika menggunakan FrankenPHP saja terdapat peningkatan sebesar 23,6%–32%. Sedangkan kombinasi antara FrankenPHP dan metode *queue* dapat menurunkan waktu respon hingga 89%–89,37%. Penurunan *time per request* menunjukkan bahwa pengguna mendapatkan data lebih cepat melalui kombinasi FrankenPHP dan metode *queue*. Penggunaan CPU juga turun sebesar 27,23%–44,05% saat diimplementasi dengan metode *queue*, sedangkan FrankenPHP saja tidak terlalu berdampak pada penurunan CPU dengan perubahan berkisar antara -6,98% hingga +1,13%. Kombinasi metode *queue* dan FrankenPHP menghasilkan penurunan CPU terbesar, yaitu 33,03%–47,17%. Dengan penurunan CPU tersebut maka beban kerja server juga ikut menurun sehingga menghindari terjadinya server *down*. Namun demikian, seluruh metode optimasi menunjukkan peningkatan konsumsi memori. Penggunaan *queue* saja meningkatkan memori sekitar 3,74%–3,96%, FrankenPHP sebesar 8,98%–9,14%, dan kombinasi keduanya mencatat kenaikan tertinggi yakni 13,84%–14,29%. Kemudian pada endpoint yang dioptimasi dengan *in-memory cache* hasil pengujian menunjukkan bahwa terjadi peningkatan RPS sebesar 11% setelah menggunakan *in-memory cache*, kemudian naik lagi sekitar 72% saat menggunakan FrankenPHP, dan mencapai peningkatan total hingga 116% saat menggunakan kombinasi FrankenPHP dan metode *in-memory cache*. Hal ini menyebabkan sistem mampu menangani jumlah permintaan yang jauh lebih besar dalam setiap detiknya. Akibatnya, *time per request* mengalami penurunan rata-rata sebesar 10% setelah penerapan *in-memory cache*, lalu turun lagi sekitar 32% dengan penggunaan FrankenPHP, dan mencapai penurunan total sekitar 48% setelah dikombinasikan antara FrankenPHP dan metode *in-memory*

cache. Penurunan *time per request* juga mengakibatkan penurunan penggunaan CPU secara signifikan. Penggunaan CPU menurun sekitar 5%–35% dengan *in-memory cache*, lalu turun lebih jauh sekitar 12%–19% dengan FrankenPHP, dan mencapai penurunan maksimal sekitar 30% dengan kombinasi keduanya. Meskipun penggunaan memori meningkat, kenaikannya masih tergolong stabil dan terkendali, yaitu sekitar 5,2% dengan *in-memory cache*, 9,8% dengan FrankenPHP, dan total peningkatan sekitar 11,6% setelah kombinasi FrankenPHP dan *in-memory cache* diterapkan.

3. Rekomendasi terbaik dalam upaya optimasi performa REST API pada aplikasi Jadi Juara adalah dengan mengganti web server yang digunakan saat ini ke FrankenPHP dan menerapkan Redis sebagai solusi penyimpanan data pada memori melalui implementasi metode *in-memory cache* dan *queue*. Pada endpoint yang sering diakses sebaiknya diimplementasikan metode *in-memory cache* agar data yang sering diakses tidak melakukan *query* ulang ke database. Sedangkan pada endpoint yang diimplementasikan *queue* yaitu endpoint yang dapat berjalan secara asinkron. Penggunaan FrankenPHP dengan *worker mode* sebagai web server juga direkomendasikan karena terbukti mampu meningkatkan *request per second*, menurunkan *time per request* serta menurunkan penggunaan CPU saat terjadi lonjakan pengguna yang tinggi. Oleh karena itu, kombinasi FrankenPHP dan *in-memory cache* serta *queue* ini sangat direkomendasikan pada optimasi REST API aplikasi Jadi Juara agar mengurangi risiko terjadinya *server down*.

5.2. Saran

Dari hasil penelitian dan kesimpulan yang telah dijabarkan, berikut beberapa saran untuk pengembangan dan penelitian selanjutnya.

1. Membandingkan dengan web server yang PHP yang lain seperti swoole dan roadrunner
2. Melakukan pengujian dengan concurrent yang lebih tinggi lagi.

Halaman ini sengaja dikosongkan