BAB I
PENDAHULUAN

1.1 Latar Belakang

Application Programming Interface (API) adalah antarmuka perangkat lunak yang memungkinkan komunikasi dan pertukaran data antara berbagai sistem atau aplikasi, seperti aplikasi web, *mobile*, atau *database*. Menurut Faza Alameka et al. API berperan penting dalam memfasilitasi akses data yang efisien [1], dengan menyediakan mekanisme standar untuk bertukar informasi dalam format seperti JSON atau XML. Kemampuan API mendukung interaksi lintas platform menjadikannya elemen penting dalam pengembangan aplikasi modern, yang menuntut skalabilitas dan efisiensi untuk memenuhi harapan pengguna.

API dengan performa tinggi sangat penting untuk menjaga kelancaran alur kerja aplikasi dan meningkatkan pengalaman pengguna. Seperti dijelaskan oleh Deepak, API yang kurang optimal dapat menyebabkan keterlambatan *response*, gangguan operasional, dan penurunan kepuasan pengguna, yang berdampak signifikan pada daya saing aplikasi [2]. Penelitian Fadida Zanetti Junaedy dan Untung Surapati menunjukkan bahwa optimalisasi performa API, seperti melalui penyeimbangan beban dan *caching*, dapat meningkatkan waktu *response* dan stabilitas sistem hingga 5 kali lipat, yang menunjukkan bahwa optimalisasi dapat lebih efisien dan responsif terhadap permintaan klien [3].

Di antara berbagai arsitektur API, RESTful API merupakan arsitektur yang umum digunakan karena kesederhanaan, fleksibilitas, dan kompatibilitasnya dengan berbagai platform. RESTful API memanfaatkan protokol dan metode HTTP seperti *GET, POST, PUT,* dan *DELETE*, dengan format data yang ringan seperti JSON, sehingga mendukung komunikasi efisien antara *frontend* dan *backend*. Keunggulan RESTful API sebagaimana dikemukakan oleh Deny Eko Septian dan Hutabri, terletak pada kemampuannya untuk mempercepat siklus pengembangan aplikasi dan beradaptasi dengan perubahan kebutuhan pengguna, sehingga menjadi pilihan yang baik untuk pengembangan API [4].

Untuk memastikan RESTful API dapat permintaan pengguna secara cepat dan stabil, pengujian performa menjadi tahap penting. Pengujian ini bertujuan untuk mengidentifikasi kelemahan sistem, seperti waktu respons yang lambat, tingkat kesalahan (*error rate*) yang tinggi, atau kapasitas sistem yang terbatas (*throughput* rendah). Fadida Zanetti Junaedy dan Untung Surapati menunjukkan bahwa pengujian beban berhasil mengidentifikasi *bottleneck* dalam menangani *request* [3].

Sistem monitoring lalu lintas di Politeknik Negeri Bengkalis yang menjadi fokus penelitian ini umumnya hanya diakses oleh admin atau petugas tertentu. Namun, sistem monitoring tetap menghasilkan beban trafik yang tinggi karena menggunakan teknik *polling* API secara berkala (setiap beberapa detik) untuk memperoleh data *real-time*, seperti jumlah kendaraan dan rata-rata kecepatan. Pola *polling* yang berulang ini membuat jumlah permintaan API terus-menerus tinggi selama sistem beroperasi, meskipun jumlah pengguna terbatas. Kondisi tersebut menuntut API memiliki performa yang andal dan efisien agar tetap mampu merespons secara cepat dan stabil.

Metode *load testing* dinilai efektif dalam menguji kemampuan API pada kondisi beban tinggi. Setiawan, G. H., Adnyana, I. M. B., & Budiarta, K.menyebutkan bahwa *load testing* mensimulasikan beban pengguna dalam jumlah besar untuk mengevaluasi performa API, dengan fokus pada metrik seperti waktu respons (*response time*), jumlah permintaan yang berhasil diproses (*throughput*), dan tingkat kesalahan (*error rate*) [5].

Dalam penelitian ini, Apache JMeter dipilih sebagai alat pengujian karena kemampuannya mensimulasikan beban pengguna secara realistis dan menghasilkan laporan metrik lengkap. JMeter dapat mengatur jumlah pengguna virtual dan jenis permintaan API, serta mendukung analisis hasil pengujian dalam format yang memudahkan evaluasi.

Setelah batas performa API teridentifikasi, langkah optimasi menjadi sangat penting untuk meningkatkan kecepatan, stabilitas, dan efisiensi layanan. Dua teknik utama yang akan diterapkan adalah *caching* dan *query optimization.*

Caching menyimpan data yang sering diakses pada memori sementara seperti Redis untuk mengurangi beban database dan mempercepat waktu respons. Fadida Zanetti Junaedy dan Untung Surapati menunjukkan bahwa penerapan *caching* dapat meningkatkan kinerja API hingga 5 kali lipat dalam hal waktu *response* [3]. Sementara itu, *query optimization* bertujuan menyederhanakan pengambilan data dari *database*, misalnya dengan meminimalkan operasi *join*, memilih kolom tertentu, atau menambahkan indeks untuk mempercepat eksekusi *query*.

Penelitian ini menekankan evaluasi performa RESTful API dengan menggunakan *load testing* sebelum dan sesudah optimasi diterapkan. Pengujian beban menggunakan Apache JMeter akan mensimulasikan permintaan dalam skala besar dan mengukur metrik utama seperti *response time, throughput*, dan *error rate*. Dengan demikian, penelitian ini bertujuan memberikan gambaran nyata tentang dampak teknik optimasi *caching* dan *query optimization* terhadap efisiensi dan keandalan RESTful API, khususnya dalam konteks aplikasi monitoring berbasis data real-time.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang, penelitian ini berfokus pada dua pertanyaan utama untuk meningkatkan performa RESTful API. Pertama, Bagaimana efektivitas teknik caching dan query optimization dalam mengoptimalkan performa RESTful API? Kedua bagaimana perbandingan performa RESTful API dalam menghadapi polling request sebelum dan sesudah penerapan teknik caching dan query optimization?

1.3 Batasan Masalah

Agar penelitian ini terfokus dan terarah, maka ruang lingkup penelitian dibatasi pada hal-hal berikut:

1. Teknik optimasi yang diterapkan hanya mencakup *caching dan query optimization* .
2. Tidak membahasa bagaimana RESTful API dibangun

3. Evaluasi performa API berpusat pada tiga metrik utama, yaitu *response time*, *throughput ,* dan *error rate*.
4. Penelitian ini menguji *endpoint GET* pada RESTful API.

1.4 Tujuan

Penelitian ini bertujuan untuk mengkaji dampak penerapan teknik *Caching* dan *query optimization* terhadap performa RESTful API, dengan tujuan spesifik sebagai berikut:

1. Menganalisis pengaruh teknik *caching* dan *query optimization* terhadap performa RESTful API.
2. Mengidentifikasi kontribusi teknik *caching* dan *query optimization* terhadap efisiensi RESTful API.
3. Mendapatkan wawasan terkait optimasi RESTful API menggunakan *query optimization* dan *caching*.

1.5 Manfaat

Penelitian ini diharapkan memberikan kontribusi baik secara praktis maupun akademis:

1. Hasil penelitian dapat menjadi panduan bagi pengembang aplikasi, untuk menerapkan teknik *caching* dan *query optimization* untuk meningkatkan performa RESTful API.
2. Memperluas pengetahuan di bidang rekayasa perangkat lunak, khususnya dalam konteks penerapan strategi optimasi performa RESTful API