

PENGEMBANGAN WEB (TEORI)

LAPORAN EKSPERIMEN MENGENAI PROBLEM HANDLING API PAGINATION AND FILTERING IN RESTFUL SERVICES

Laporan ini disusun untuk memenuhi tugas 1 mata kuliah Pengembangan Web (Teori)



Disusun oleh kelompok B4:

Asri Husnul Rosadi	221524035
Faris Abulkhoir	221524040
Mahardika Pratama	221524044
Muhamad Fahri Yuwan	221524047
Najib Alimudin Fajri	221524053
Septyana Agustina	221524058
Sarah	221524059

Dosen Pengampu:
Joe Lian Min, M.Eng.

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA
PROGRAM STUDI D4 TEKNIK INFORMATIKA
POLITEKNIK NEGERI BANDUNG
2024**

DAFTAR ISI

DAFTAR ISI	i
A. IDENTIFIKASI PROBLEM	1
B. DESKRIPSI PROBLEM	1
C. METODOLOGI EKSPERIMEN	1
D. PELAKSANAAN EKSPERIMEN	2
E. ANALISIS HASIL EKSPERIMEN	2
F. KESIMPULAN	4

Link chatGPT : <https://chatgpt.com/share/7fc18555-a833-4a3d-92ef-b976cacd3e8d>

A. IDENTIFIKASI PROBLEM

Permasalahan yang dihadapi adalah mengoptimalkan kinerja API dalam menangani permintaan data dalam jumlah besar. Fokus utamanya adalah membandingkan kinerja API saat menggunakan pagination versus tanpa pagination dalam pengambilan data besar, terutama pada kategori produk "Electronics" yang memiliki 333,721 entri data.

B. DESKRIPSI PROBLEM

API yang diimplementasikan berfungsi untuk mengambil data produk dari database dengan fitur filtering berdasarkan kategori, harga minimum, dan harga maksimum. Namun, tanpa adanya pagination, API dapat mengalami masalah performa saat menangani permintaan dengan jumlah data besar, seperti mengirimkan seluruh data sekaligus. Oleh karena itu, penting untuk mengukur efisiensi dan performa API dengan pagination dibandingkan tanpa pagination.

C. METODOLOGI EKSPERIMEN

1. Lingkungan Eksperimen:

- a. Framework: FastAPI
- b. Database: PostgreSQL dengan tabel products yang memiliki kolom id, harga, stok, nama_produk, dan kategori.
- c. Jumlah data kategori 'Electronics': 333,721 entri.

2. Metode yang Diuji:

- a. **Dengan Pagination:** Menggunakan parameter page, page_size, category, min_price, dan max_price untuk mengambil data terbatas dalam jumlah tertentu per halaman.
- b. **Tanpa Pagination:** Mengambil seluruh data sesuai filter kategori dan harga tanpa membatasi jumlah data yang diambil.

3. Langkah Eksperimen:

a. Endpoint Pagination:

- i. `http://127.0.0.1:8000/products?page=1&page_size=10000&category=Electronics&min_price=10&max_price=500`

b. Endpoint Tanpa Pagination:

- i. `http://127.0.0.1:8000/products/no-pagination?category=Electronics&min_price=10&max_price=50`

4. Pengukuran:

- Waktu respons.
- Penggunaan memori.
- Efisiensi pengiriman data (jumlah data yang dikirim vs total data yang tersedia).

D. PELAKSANAAN EKSPERIMEN

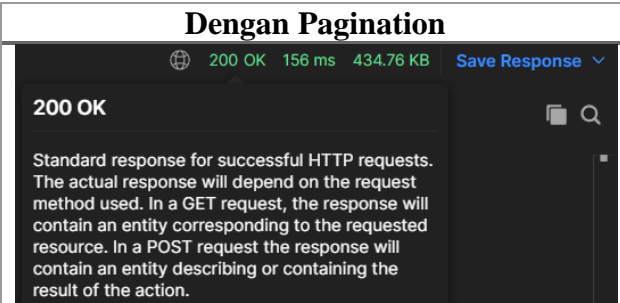
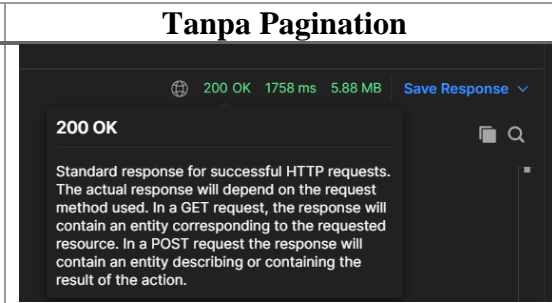
1. Dengan Pagination:

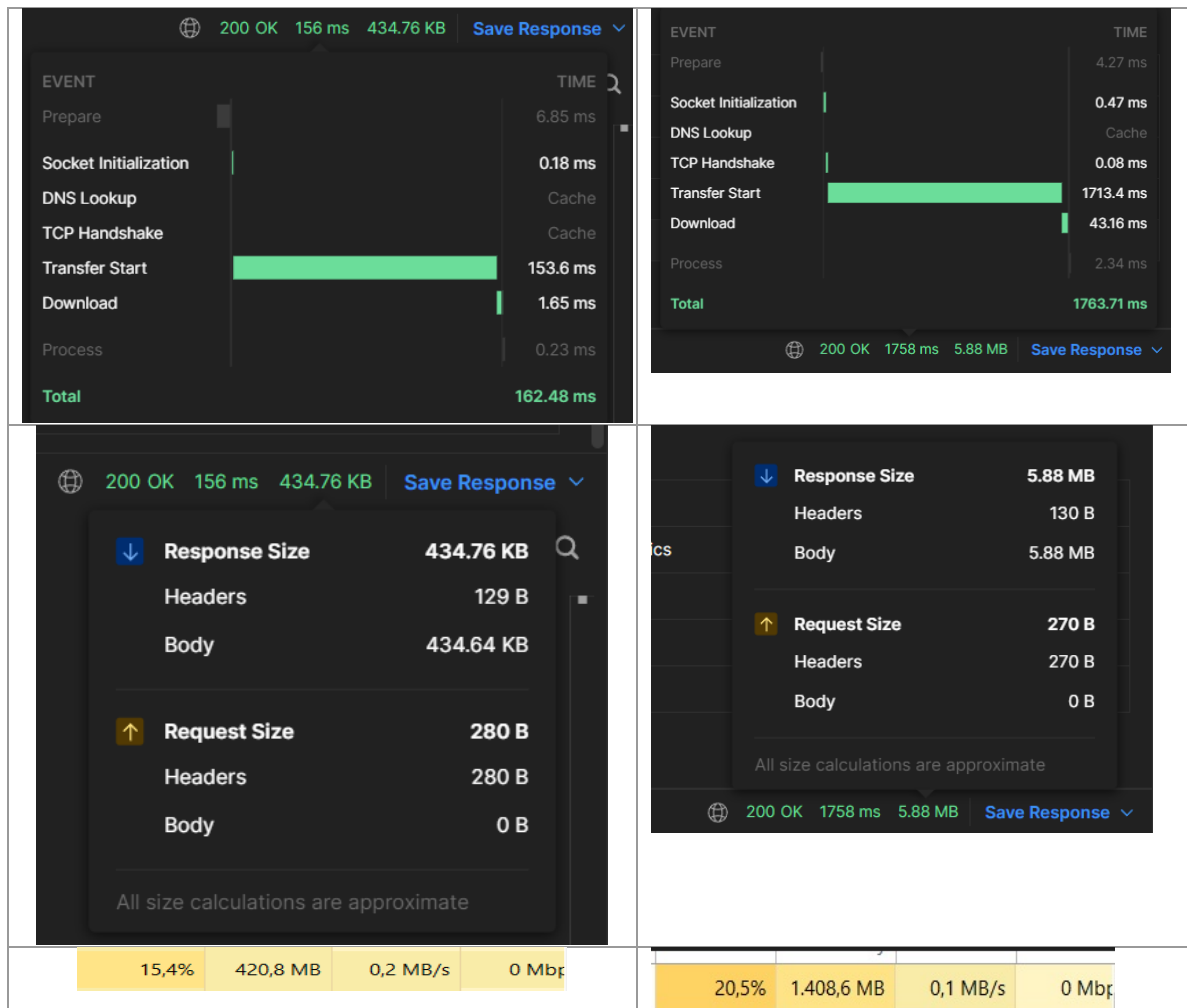
- a. Data diambil dengan `page_size` 10,000 per halaman, untuk produk kategori 'Electronics' dengan rentang harga 10 hingga 500.
- b. Total entri yang tersedia: 333,721 data, sehingga dibagi menjadi beberapa halaman.

2. Tanpa Pagination:

- a. Seluruh data yang memenuhi syarat filter diambil sekaligus tanpa pembagian halaman.
- b. Kondisi ini memungkinkan pengambilan seluruh 333,721 entri dalam satu permintaan jika kategori dan harga sesuai filter.

E. ANALISIS HASIL EKSPERIMEN

Dengan Pagination	Tanpa Pagination
 <p>200 OK 156 ms 434.76 KB Save Response</p> <p>200 OK</p> <p>Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request the response will contain an entity describing or containing the result of the action.</p>	 <p>200 OK 1758 ms 5.88 MB Save Response</p> <p>200 OK</p> <p>Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request the response will contain an entity describing or containing the result of the action.</p>



1. Dengan Pagination:

- Waktu Respons:** Rata-rata respons lebih cepat karena jumlah data yang dikirim per permintaan dibatasi menjadi 10,000 entri.
- Penggunaan Memori:** Penggunaan memori lebih efisien karena server hanya mengelola sebagian kecil data dalam setiap permintaan.
- Efisiensi Pengiriman Data:** API mampu mengontrol jumlah data yang dikirim sehingga tidak membebani server maupun klien.

2. Tanpa Pagination:

- Waktu Respons:** Waktu respons jauh lebih lama karena seluruh data dikirimkan dalam satu permintaan.
- Penggunaan Memori:** Server menggunakan memori lebih besar karena harus menangani dan mengirimkan seluruh 333,721 entri sekaligus.
- Efisiensi Pengiriman Data:** Pengiriman data tidak efisien karena server harus menangani permintaan besar dan klien harus menerima seluruh data sekaligus, yang dapat menyebabkan latency tinggi dan kegagalan pengiriman data jika bandwidth terbatas.

F. KESIMPULAN

Pagination adalah pendekatan yang lebih efisien untuk menangani data besar dalam API. Kelebihan dari pagination meliputi peningkatan kinerja dan pengalaman pengguna, dengan waktu respon yang lebih cepat dan penghematan penggunaan memori. Di sisi lain, tanpa pagination, kinerja server bisa menurun drastis ketika harus memproses dan mengembalikan seluruh data sekaligus.

Kelebihan Pagination:

- Mengurangi waktu respon server dengan membatasi jumlah data yang dikirim dalam satu permintaan.
- Menghemat sumber daya server dan perangkat klien dengan menghindari pengambilan seluruh data secara langsung.
- Meningkatkan pengalaman pengguna dengan pengiriman data yang lebih cepat dan stabil.

Kekurangan Pagination:

- Memerlukan implementasi tambahan pada sisi server dan klien untuk menangani navigasi antar halaman data.
- Pengguna mungkin perlu beberapa kali klik untuk mendapatkan seluruh informasi yang mereka butuhkan.

Kapan Menggunakan Pagination:

- Pagination sangat dianjurkan ketika API harus menangani data besar yang melebihi ribuan atau ratusan ribu entri, seperti pada database produk e-commerce dengan ribuan item.
- Pagination juga tepat digunakan ketika data disajikan kepada pengguna secara bertahap, seperti daftar produk, artikel, atau komentar, di mana pengguna tidak memerlukan semua data sekaligus.
- Namun, jika data yang diambil dalam jumlah kecil atau ada kebutuhan untuk memproses seluruh data dalam satu permintaan (misalnya, laporan analitik), pagination mungkin tidak diperlukan.

Dalam kasus data besar seperti 333,721 entri, pagination terbukti menjadi solusi yang lebih optimal untuk mengelola beban dan menjaga kinerja aplikasi tetap stabil.