**FINAL PROJECT SOFTWARE TESTING & QUALITY ASSURANCE A**

“Laporan Hasil Scalability Testing With Gatling”

****

**Disusun Oleh:**

**Mahendra Kirana M.B**

H071221058

**PROGRAM STUDI SISTEM INFORMASI**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS HASANUDDIN**

**2024**

# KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya, sehingga kami dapat menyelesaikan laporan hasil pengujian performa API ini. Laporan ini disusun sebagai bagian dari kegiatan pengujian performa menggunakan alat Gatling untuk menguji sistem API dan mengukur kemampuannya dalam menangani beban pengguna yang tinggi.

Laporan ini disusun dengan penuh perhatian dan mengikuti standar yang telah ditetapkan. Kami berharap laporan ini dapat memberikan informasi yang jelas dan bermanfaat bagi pengembangan dan pemeliharaan sistem yang diuji.

Saya juga mengucapkan terima kasih kepada Bapak Abdul Kadir Jailani, S.Kom., M.T. yang telah memberikan bimbingan, masukan, dan dukungan selama proses penyusunan laporan ini.

Semoga laporan ini dapat memberikan kontribusi positif dalam peningkatan kualitas sistem yang diuji.

Mahendra Kirana M.B

# DAFTAR ISI

[**KATA PENGANTAR 2**](#_8uhcrdd32wb0)

[**DAFTAR ISI 3**](#_3pyvkuyhnsit)

[**BAB I 4**](#_fpmb6x7155y8)

[**PENDAHULUAN 4**](#_j4i788w77h1z)

[1. Latar Belakang 4](#_iq4uem9a8vfl)

[2. Tujuan Pengujian 4](#_x61pfdng0q7q)

[3. Deskripsi Sistem yang Diuji 4](#_uefhouosdg8f)

[4. Alat dan Teknologi yang Digunakan 5](#_7vq2dfeirm8e)

[**BAB II 6**](#_v12vgxwiryo9)

[METODOLOGI PENGUJIAN 6](#_cphl1bh3e4bf)

[1. Jenis Pengujian 6](#_bngu7gam0u2r)

[2. Kriteria Keberhasilan Pengujian 6](#_rulw5sz0olor)

[3. Pendekatan Pengujian 7](#_mix0coqqwub)

[**BAB III 9**](#_q0q2pq9usv8c)

[**KONFIGURASI LINGKUNGAN PENGUJIAN 9**](#_16wd2dq644hv)

[1. Lingkungan Server 9](#_m9f2vp686ax4)

[2. Lingkungan Klien 9](#_3re7wxuw1xmt)

[3. Pengaturan Simulasi 10](#_wg4f6vwijhj7)

[**BAB IV 12**](#_xbyveeqffllw)

[**HASIL PENGUJIAN DAN ANALISIS 12**](#_o30bmiwqscls)

[1. Metrik Pengujian 12](#_32rm7ysv0j7g)

[2. Analisis Performa 12](#_u872xiwwvelw)

[3. Visualisasi dan Grafik Secara Keseluruhan 22](#_k5255wr88tf4)

[**BAB V 24**](#_mcz445dp2xq5)

[**PENUTUP 24**](#_agn42h8zlfmq)

[**1. Kesimpulan 24**](#_cgz1s2qk0kkk)

[2. Saran 24](#_y5jql1cls3pr)

[**REFERENSI 26**](#_zgusqm6qqenh)

# BAB I

# PENDAHULUAN

## Latar Belakang

Performa suatu sistem API sangat krusial dalam mendukung kelancaran operasional aplikasi berbasis web atau mobile. Salah satu aspek yang penting yang perlu diuji adalah kemampuan API dalam menangani jumlah pengguna yang tinggi atau beban yang besar. Pengujian performa ini betujuan untuk mengidentifikasi batas kemampuan API dan mengukur seberapa baik sistem dapat menangani skenario penggunaan yang realistis.

Seiring dengan perkembangan teknologi dan peningkatan jumlah pengguna aplikasi, pengujian skalabilitas menjadi salah satu tahapan yang tidak bisa diabaikan dalam pengembangan perangkat lunak. Oleh karena itu, penting bagi tim pengembang untuk menggunakan alat yang dapat mensimulasikan beban pengguna secara realistis, serta menganalisis kinerja sistem API dalam kondisi tersebut.

## Tujuan Pengujian

Tujuan dari pengujian ini adalah untuk mengukur dan menganalisis performa API dalam menangani beban pengguna yang meningkat, serta untuk mengidentifikasi potensi masalah yang dapat terjadi pada API ketika diberikan beban yang lebih berat. Secara spesifik, pengujian ini bertujuan untuk :

* Menilai kemampuan API dalam melayani berbagai jenis permintaan (seperti pengambilan data pengguna dan pembuatan data pengguna baru) di bawah tekanan beban yang bervariasi.
* Mengidentifikasi potensi bottleneck yang dapat menghambat kinerja sistem.
* Menghasilkan data yang digunakan untuk melakukan perbaikan terhadap performa sistem.

## Deskripsi Sistem yang Diuji

Sistem yang diuji adalah API yang disediakan oleh **Reqres.in**, yang digunakan untuk mengelola data pengguna. API ini menyediakan berbagai endpoint untuk operasi seperti mengambil daftar pengguna, mengambil data pengguna tunggal, serta membuat pengguna baru, Pengujian ini difokuskan pada pengujian performa API dengan beberapa jenis permintaan yang meliputi :

* Mendapatkan daftar pengguna
* Membuat pengguna baru
* Mengambil data pengguna tunggal
* Mengirimkan permintaan tidak valid

## Alat dan Teknologi yang Digunakan

Alat yang digunakan untuk melakukan pengujian skalabilitas adalag Gatling, sebuah tool open-source yang dirancang untuk melakukan pengujian beban dan performa pada aplikasi web. Gatling digunakan untuk mensimulasikan skenario dengan sejumlah besar pengguna dan mengukur performa API di vawah beban tersebut

Beberapa teknologi yang digunakan selama pengujian ini adalah :

* Gatling : Untuk simulasi beban dan pengukuran performa
* Java : Untuk menulis dan menjalankan simulasi gatling.
* API Reqres.in : Sistem API yang diuji

# BAB II

## METODOLOGI PENGUJIAN

## Jenis Pengujian

Pengujian ini menggunakan pendekatan scalability testing, yaitu pengujian yang berfokus pada kemampuan sistem dalam menangani peningkatan beban secara bertahap. Scalability Testing dirancang untuk mengukur seberapa baik sistem dapat meningkatkan kinerjanya saat sumber data (seperti jumlah pengguna atau permintaan) bertambah. Pengujian ini membantu mengidentifikasi batas kapasitas sistem dan memberikan wawasan tentang perilaku sistem di bawah kondisi beban yang berbeda.

Dalam pengujian ini, simulasi dilakukan untuk meniru kondisi nyata dengan variasi jumlah pengguna yang diakses secara bersamaan. Beban pengguna ditingkatkan secara bertahap (ramp-up) untuk menguji stabilitas dan efisiensi sistem. Selain itu, pengujian ini juga memantau konsistensi performa selama fase steady-state, yaitu ketika jumlah pengguna konstan untuk periode tertentu

## Kriteria Keberhasilan Pengujian

Kriteria keberhasilan dalam pengujian ini ditentukan berdasarkan beberapa metrik kunci yang diukur selama simulasi, yaitu :

1. Waktu Response
   1. Waktu respons harus tetap berada dalam ambang batas yang diterima, misalnya di bawah 500 ms untuk permintaan sederhana.
   2. Waktu respons yang konsisten menunjukkan bahwa sistem dapat menangani peningkatan beban tanpa penurunan performa yang signifikan.
2. Tingkat Kegagalan
   1. Tidak ada kegagalan signifikan selama pengujian, khususnya dalam skenario valid.
   2. Error rate harus berada di bawah 1% untuk menandakan stabilitas sistem dalam menangani permintaan.
3. Konsistensi Performa
4. Selama periode steady-state, performa sistem harus tetap stabil meskipun beban tinggi.
5. Konsistensi ini diukur melalui metrik seperti throughput (jumlah permintaan yang diproses per detik) dan jumlah koneksi aktif yang berhasil dipertahankan.
6. Penggunaan Sumber Daya
7. Konsumsi CPU dan memori harus berada dalam batas wajar selama pengujian.
8. Lonjakan penggunaan sumber daya dapat mengindikasikan potensi bottleneck atau inefisiensi dalam pengelolaan permintaan.

## Pendekatan Pengujian

Pendekatan pengujian dilakukan melalui simulasi berbasis skenario, yang mencerminkan aktivitas pengguna sebenarnya. Setiap skenario menggambarkan interaksi tertentu dengan API dan memiliki variasi beban yang dirancang untuk menguji berbagai aspek sistem. Pendekatan ini mencakup beberapa tahap utama:

1. Tahap Persiapan:
   * Menentukan tujuan pengujian dan skenario spesifik berdasarkan fungsionalitas API yang diuji.
   * Menyiapkan lingkungan pengujian, termasuk konfigurasi server dan alat pengujian seperti Gatling.
2. Tahap Pelaksanaan:
   * Melakukan injeksi beban secara bertahap (ramp-up) di mana jumlah pengguna meningkat dari nol hingga jumlah maksimum yang ditentukan.
   * Setelah mencapai jumlah maksimum, simulasi memasuki fase steady-state, di mana jumlah pengguna konstan selama periode tertentu.
   * Setiap skenario dijalankan secara paralel untuk mensimulasikan situasi di dunia nyata di mana berbagai jenis permintaan terjadi secara bersamaan.
3. Pengumpulan Data dan Monitoring:
   * Selama pengujian, data metrik performa seperti waktu respons, throughput, dan error rate dikumpulkan secara real-time.
   * Log pengujian dan hasil simulasi dianalisis untuk mengidentifikasi potensi masalah.
4. Tahap Analisis:
   * Data yang dikumpulkan dianalisis untuk mengevaluasi apakah sistem memenuhi kriteria keberhasilan.
   * Identifikasi area yang memerlukan perbaikan, seperti bottleneck dalam penanganan beban tinggi atau permintaan tertentu.
5. Pelaporan Hasil:
   * Hasil pengujian disusun dalam laporan yang mencakup temuan utama, grafik metrik, dan rekomendasi perbaikan.
   * Visualisasi performa dalam bentuk grafik membantu memberikan gambaran yang lebih jelas tentang perilaku sistem di bawah beban yang berbeda.

Dengan pendekatan ini, diharapkan pengujian dapat memberikan gambaran komprehensif tentang kemampuan sistem dalam menangani peningkatan beban, serta mengidentifikasi area yang memerlukan optimasi untuk meningkatkan performa dan skalabilitas sistem secara keseluruhan.

# BAB III

# KONFIGURASI LINGKUNGAN PENGUJIAN

## Lingkungan Server

Pengujian dilakukan terhadap Reqres.in, sebuah layanan API publik berbasis cloud yang mensimulasikan data pengguna. Reqres.in dipilih karena menyediakan berbagai endpoint yang umum digunakan dalam pengujian API, seperti operasi GET untuk mengambil data pengguna dan POST untuk membuat data pengguna baru. Beberapa karakteristik penting dari server yang diuji meliputi :

* Jenis Layanan : Layanan API berbasis cloud.
* Aksesibilitias : Layanan ini diakses melalui internet publik, sehingga performanya dapat dipengaruhi oleh faktor eksternal seperti latensi jaringan.
* Endpoint yang Digunakan :
  + /users?page=2 : untuk mengambil daftar oengguna
  + /users : untuk membuat pengguna baru
  + /users/2 : Untuk mendapatkan detail satu pengguna tertentu

Catatan : Karena layanan ini berbasis cloud, hasil pengujian dapat dipengaruhi oleh kondisi server di saat pengujian. Oleh karena itu, pengujian dilakukan beberapa kali untuk memastikan konsistensi hasil.

## Lingkungan Klien

Pengujian dilakukan menggunakan mesin klien dengan spesifikasi berikut :

* CPU : Intel Core i5 Gen 12th
* RAM : 16 GB
* Sistem Operasi : Windows 11 (64-bit)

Rincian Konfigurasi Klien :

1. **Performa Mesin Klien**:  
   Mesin klien harus memiliki daya pemrosesan yang cukup untuk menjalankan simulasi beban tanpa menimbulkan bottleneck dari sisi klien. Spesifikasi CPU dan RAM ini dipilih agar klien dapat menangani pengolahan data dari simulasi Gatling secara efisien.
2. **Jaringan**:

* **Kecepatan Internet**: Minimal 10 Mbps untuk memastikan koneksi yang stabil ke server API.
* **Latensi**: Latensi jaringan harus dipantau selama pengujian karena dapat memengaruhi hasil, terutama saat mengukur waktu respons API.

1. **Perangkat Lunak yang Digunakan**:

* **Java Development Kit (JDK)**: Versi 23.0.1, digunakan untuk menjalankan Gatling.
* **Gatling**: Versi terbaru yang kompatibel dengan konfigurasi Java.
* **Maven**: Untuk manajemen dependensi dan build project Gatling

Pengujian dilakukan pada kondisi yang terkendali, artinya tidak ada aplikasi berat lain yang berjalan di mesin klien selama pengujian untuk menghindari interferensi.

## Pengaturan Simulasi

Pengaturan simulasi dirancang untuk mensimulasikan berbagai skenario beban pengguna. Pengujian ini dilakukan dengan skenario yang melibatkan peningkatan jumlah pengguna secara bertahap untuk mengevaluasi performa sistem di bawah berbagai tingkat tekanan. Rincian pengaturan simulasi adalah sebagai berikut:

1. Jumlah Pengguna
   1. **Minimum**: 100 pengguna.  
      Simulasi dimulai dengan 100 pengguna untuk menguji performa dasar sistem.
   2. **Maksimum**: 500 pengguna.  
      Beban maksimum ditetapkan pada 500 pengguna untuk menguji batas skalabilitas API dan mengidentifikasi potensi bottleneck.
2. Durasi Simulasi
   1. **Total Durasi**: 3 menit.
   2. **Ramp-up Period (Periode Peningkatan)**: 1 menit.  
      Jumlah pengguna bertambah secara bertahap selama 1 menit pertama. Pendekatan ini membantu menilai kemampuan sistem untuk menangani peningkatan beban secara perlahan tanpa mengalami penurunan performa drastis.
   3. **Steady-state Period (Periode Stabil)**: 2 menit.  
      Setelah ramp-up selesai, jumlah pengguna konstan selama 2 menit. Fase ini digunakan untuk mengamati konsistensi sistem saat menangani beban tinggi dalam waktu yang lebih lama.
3. Jenis Beban Pengguna
   1. **Beban Bertahap (Ramp-up)**: Pengguna baru ditambahkan secara bertahap untuk menguji bagaimana sistem merespons peningkatan beban yang progresif.
   2. **Beban Konstan (Steady-state)**: Selama fase ini, sistem diuji dengan beban pengguna tetap untuk mengukur stabilitas dan throughput di bawah kondisi beban maksimum yang berkelanjutan.
4. Skenario yang diuji
   1. **Skenario 1 (Get Users Request)**:  
      Mensimulasikan permintaan untuk mengambil daftar pengguna. Fokus pada pengukuran waktu respons dan throughput.
   2. **Skenario 2 (Create User Request)**:  
      Menguji performa sistem saat menerima permintaan pembuatan pengguna baru, termasuk validasi dan respons.
   3. **Skenario 3 (Get Single User Request)**:  
      Mengukur performa sistem saat mengambil data pengguna tunggal, dengan penekanan pada waktu respons.
   4. **Skenario 4 (Invalid Create User Request)**:  
      Menguji penanganan permintaan tidak valid untuk mengevaluasi ketahanan sistem dan error handling.

**Penyesuaian Pengujian**:

Konfigurasi dapat disesuaikan berdasarkan hasil pengujian awal. Jika ditemukan anomali, simulasi dapat diulang dengan parameter yang berbeda untuk mendapatkan data yang lebih akurat.

# BAB IV

# HASIL PENGUJIAN DAN ANALISIS

## Metrik Pengujian

Dalam pengujian ini, berbagai metrik performa dikumpulkan untuk mengevaluasi kinerja sistem secara menyeluruh. Metrik-metrik ini mencerminkan aspek penting dalam pengujian performa dan skalabilitas:

1. Waktu Response

Waktu respons mengukur berapa lama waktu yang diperlukan oleh sistem untuk memproses permintaan dari klien dan mengembalikan respons. Waktu respons yang ideal tergantung pada jenis aplikasi, namun secara umum, waktu respons yang baik berada di bawah 500 milidetik (ms) untuk API. Metrik ini dianalisis untuk mengidentifikasi apakah ada peningkatan waktu respons seiring bertambahnya jumlah pengguna.

1. Throughput ( Jumlah permintaan per detik )

Throughput menunjukkan jumlah permintaan yang dapat diproses oleh sistem dalam satuan waktu tertentu (misalnya, permintaan per detik atau RPS—Requests Per Second). Throughput yang tinggi menunjukkan bahwa sistem dapat menangani beban dengan efisien. Analisis throughput membantu mengidentifikasi batas kapasitas sistem sebelum performa mulai menurun.

1. Error Rate

Error rate adalah persentase permintaan yang gagal dibandingkan dengan total permintaan yang dikirim. Error bisa terjadi akibat berbagai alasan, seperti overload, kegagalan jaringan, atau kesalahan server. Tingkat kesalahan yang rendah (di bawah 1%) menunjukkan stabilitas sistem yang baik, sedangkan tingkat kesalahan yang tinggi mengindikasikan perlunya perbaikan.

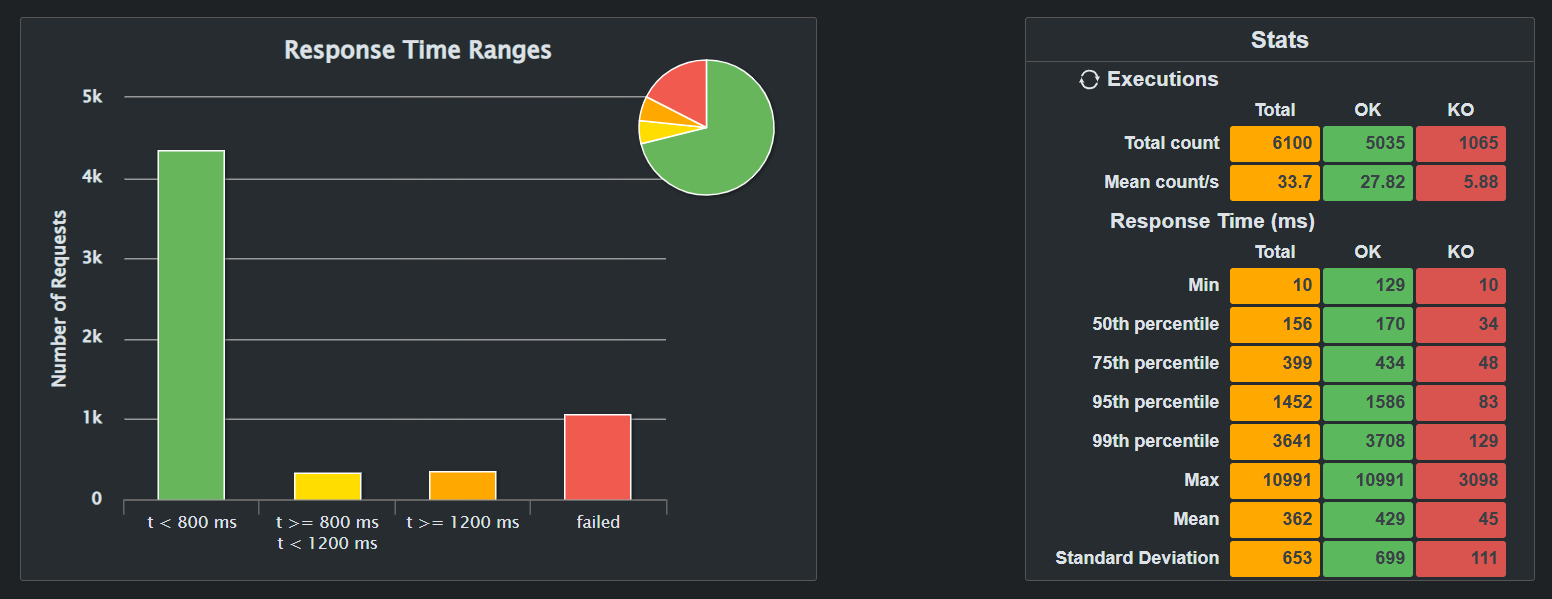
## Analisis Performa

Analisis performa dilakukan berdasarkan hasil pengujian dari setiap skenario yang telah dirancang. Setiap skenario memberikan wawasan yang berbeda mengenai kemampuan sistem dalam menangani beban.

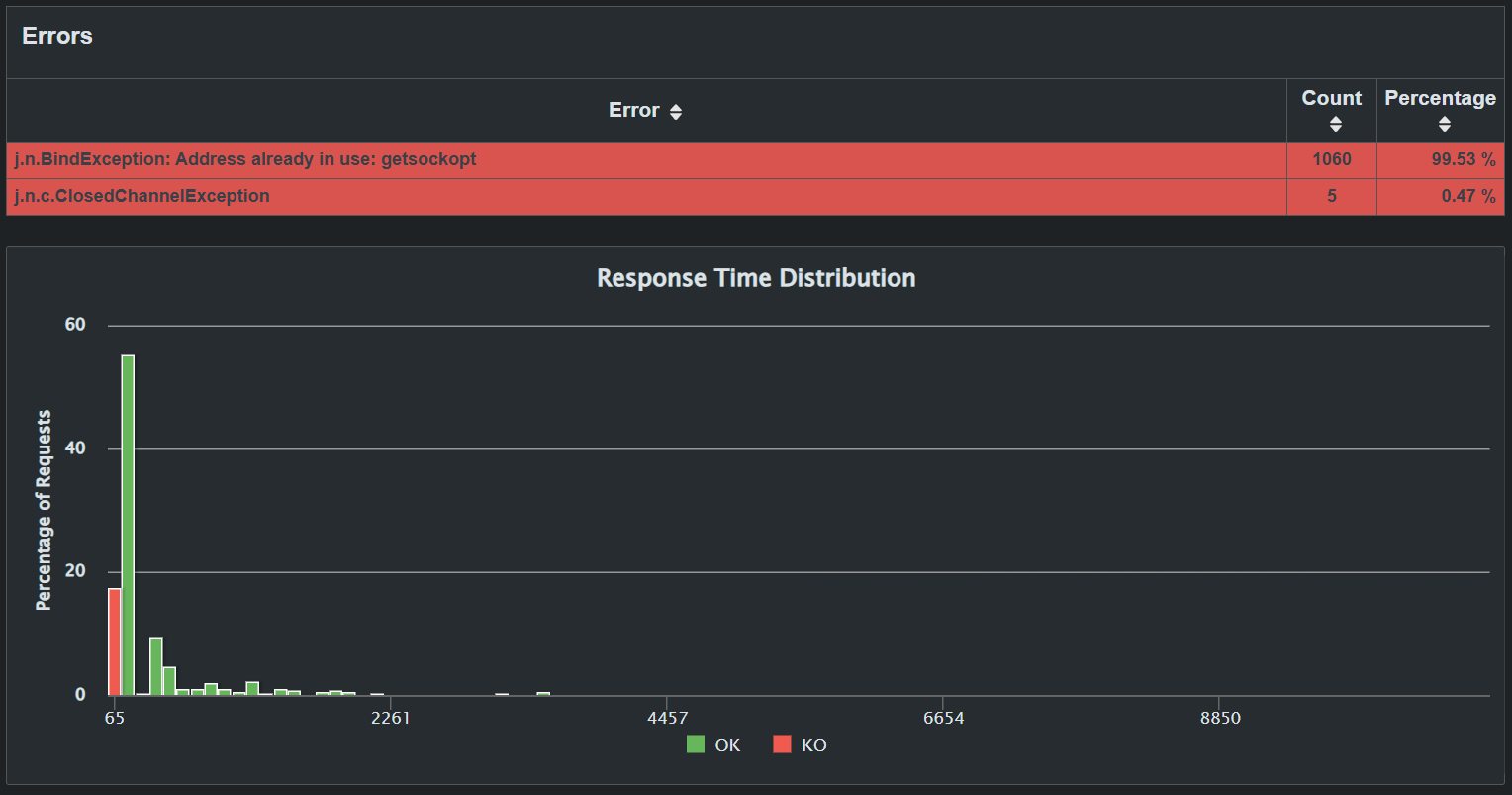
***Skenario 1 : Get Users Request***

Pada skenario ini, sistem dapat menangani permintaan pengambilan data pengguna dengan baik pada beban rendah. Namun, waktu respons meningkat secara signifikan ketika jumlah pengguna aktif melebihi 200 pengguna. Hal ini menunjukkan bahwa sistem memiliki keterbatasan dalam menangani jumlah permintaan tinggi dalam skala besar.

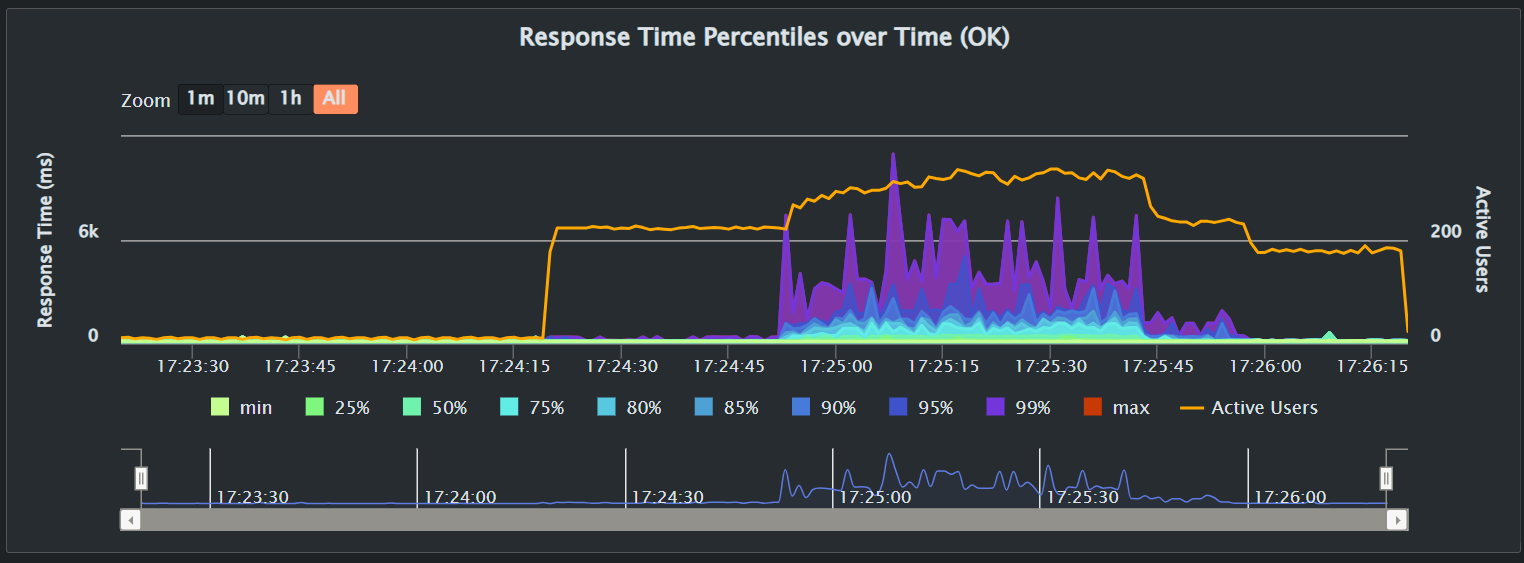
1. ***Response Time Ranges***



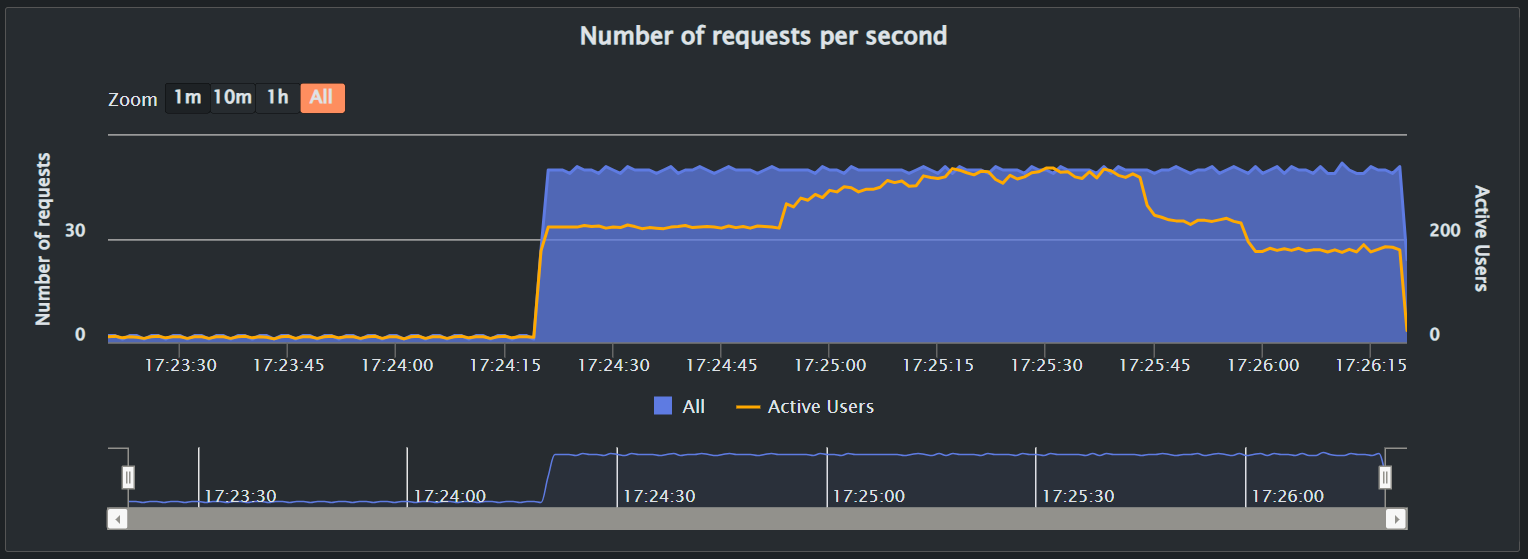
1. ***Error & Response Time Distribution***



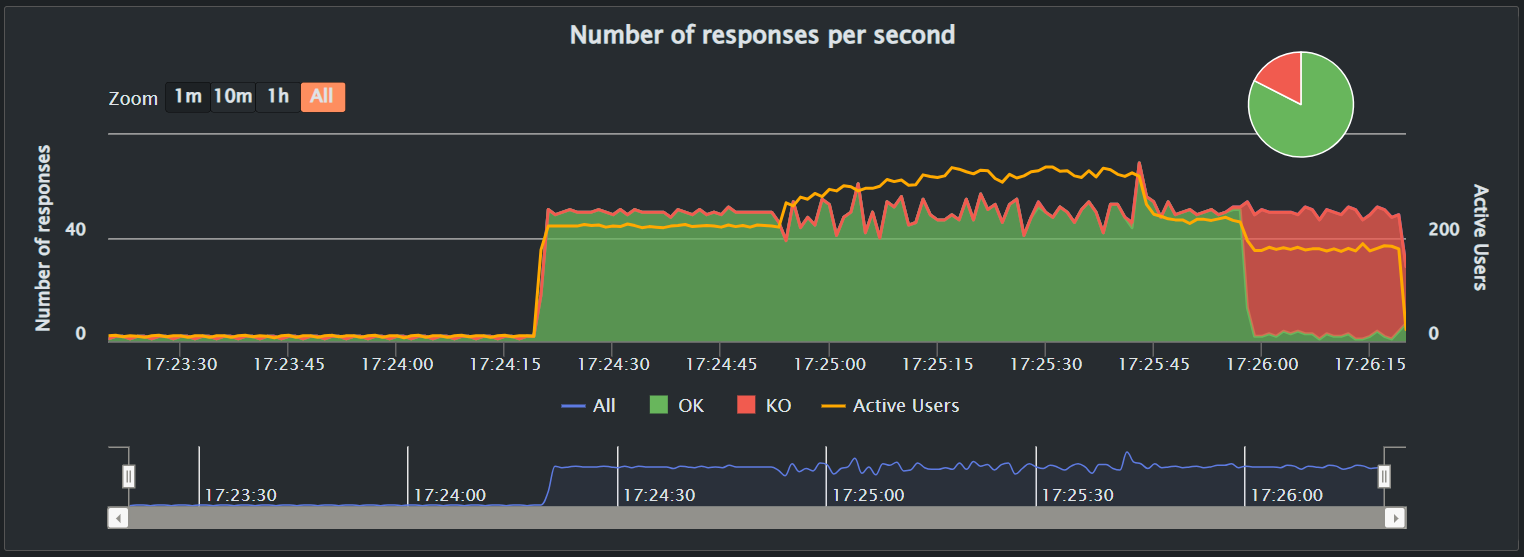
1. ***Response Time Percentiles over Time (OK)***



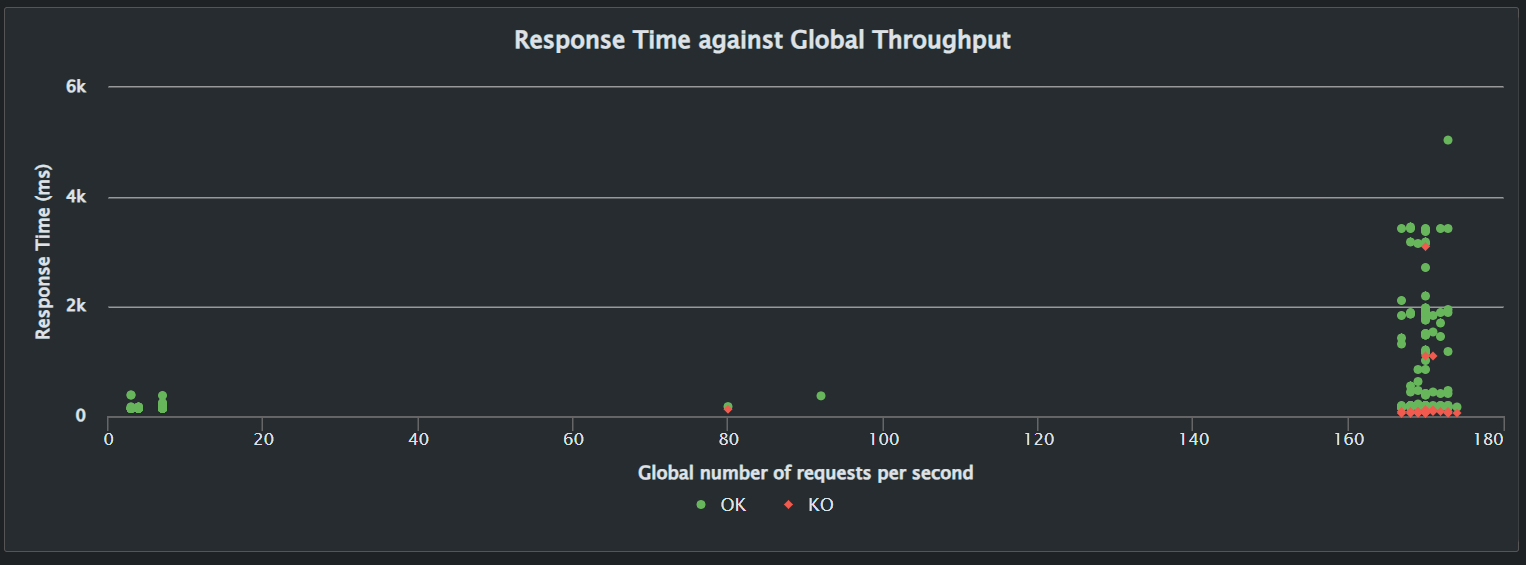
1. ***Number of requests per second***



1. ***Number of responses per second***



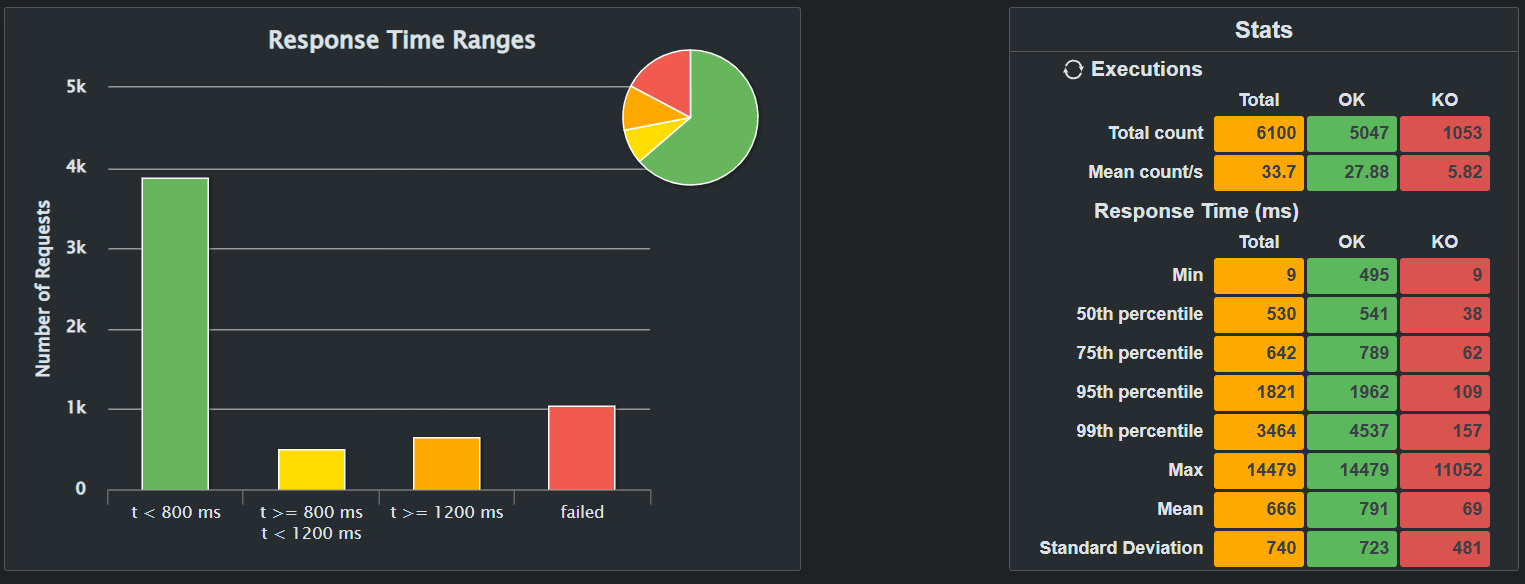
1. ***Response time against Global Throughput***



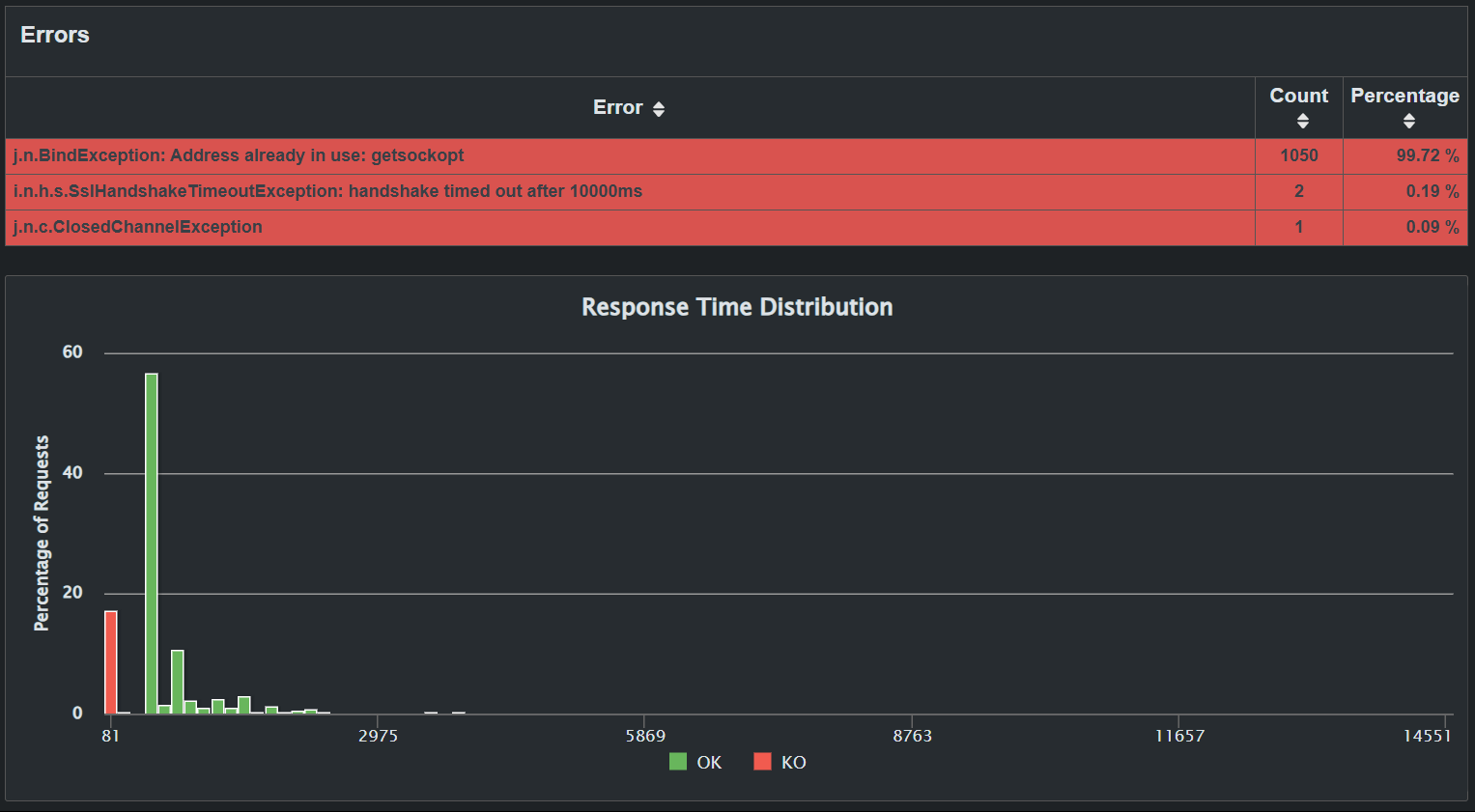
***Skenario 2 : Create Users Request***

Pengujian pembuatan pengguna menunjukkan bahwa waktu respons meningkat tajam setelah 200 pengguna, dan error rate juga mulai meningkat secara signifikan. Puncak error rate terlihat pada skenario ini, yang mengindikasikan bahwa proses pembuatan pengguna lebih berat dibandingkan dengan pengambilan data pengguna.

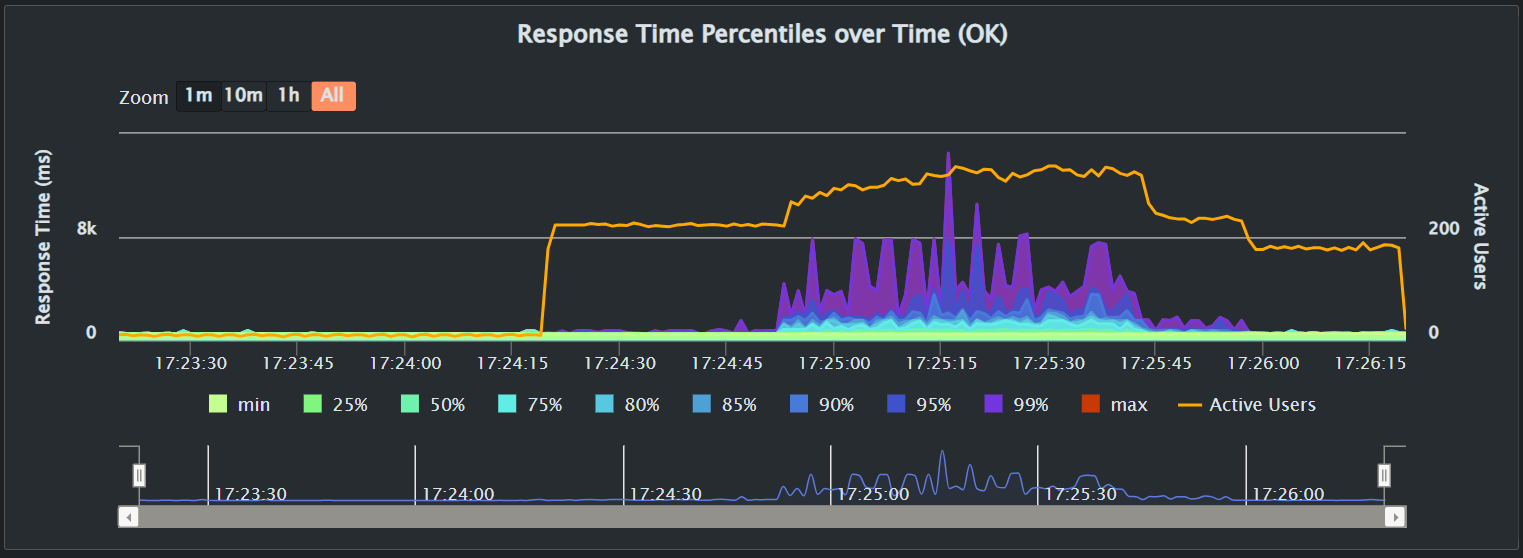
1. ***Response Time Ranges***



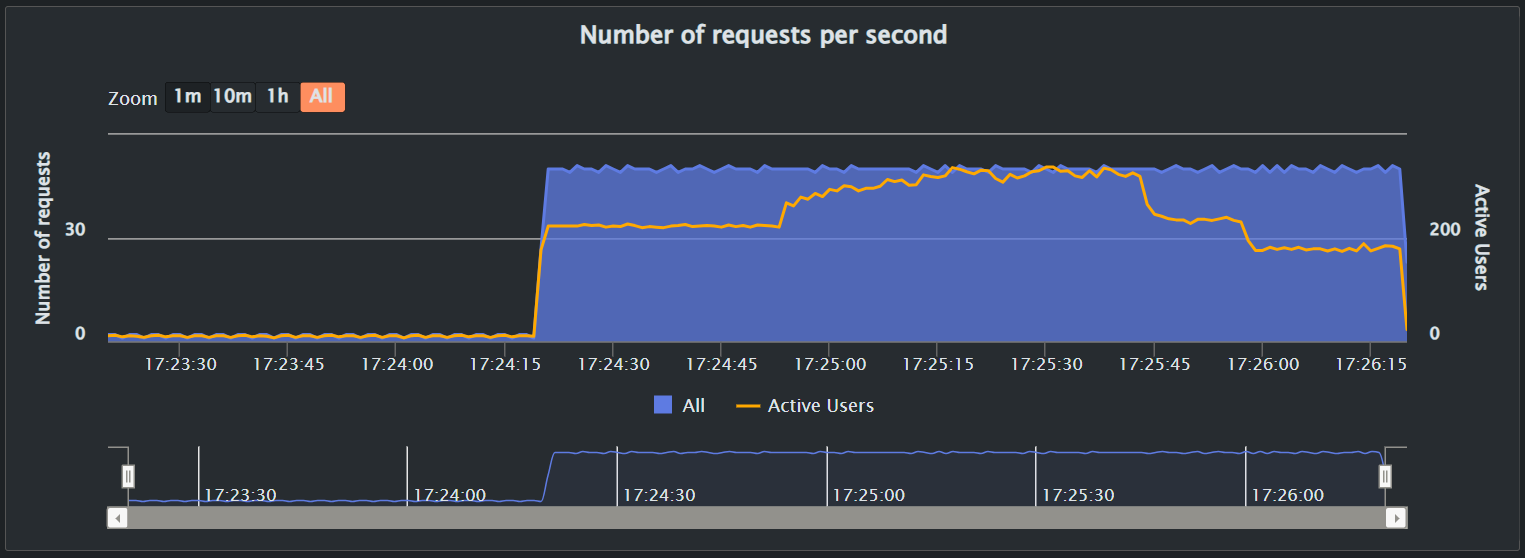
1. ***Error & Response Time Distribution***



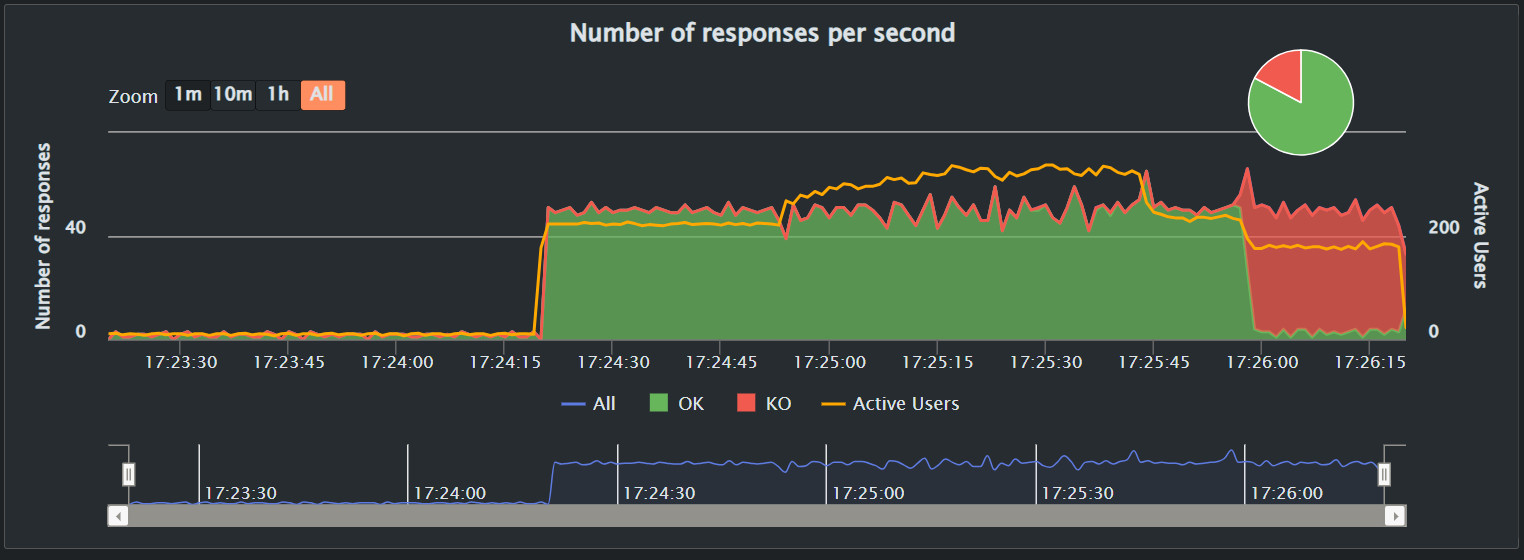
1. ***Response Time Percentiles over Time (OK)***



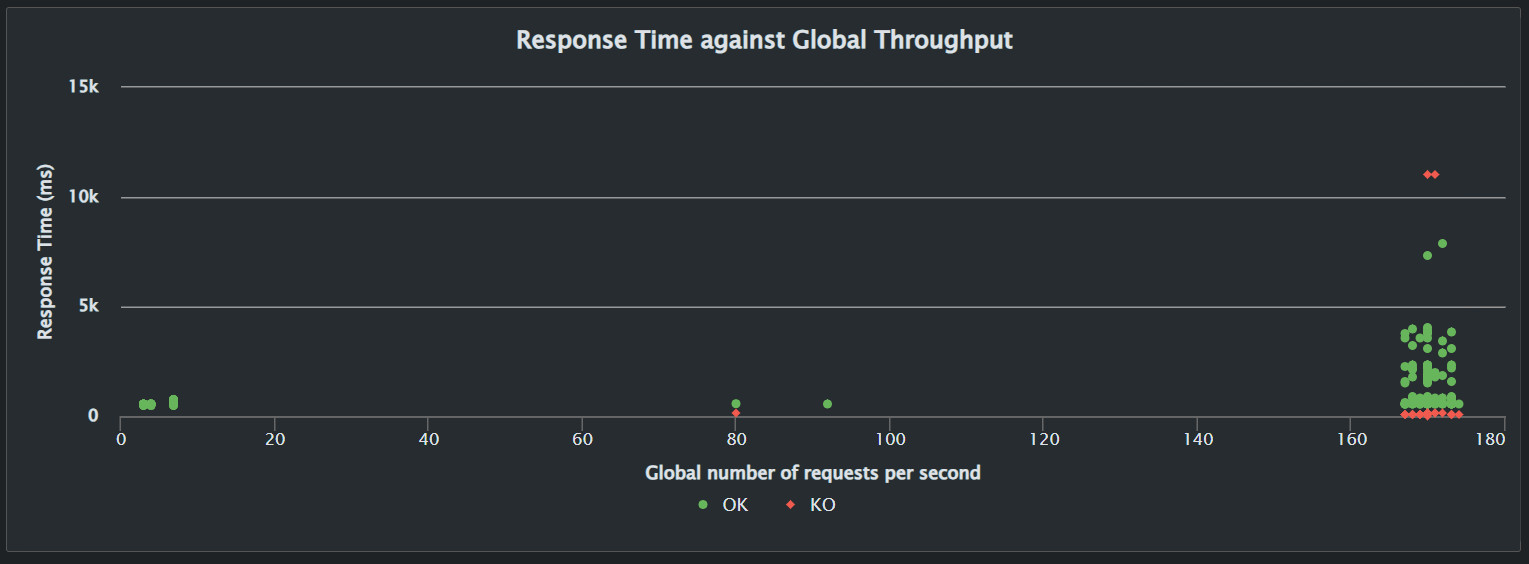
1. ***Number of Requests per second***



1. ***Number of responses per second***



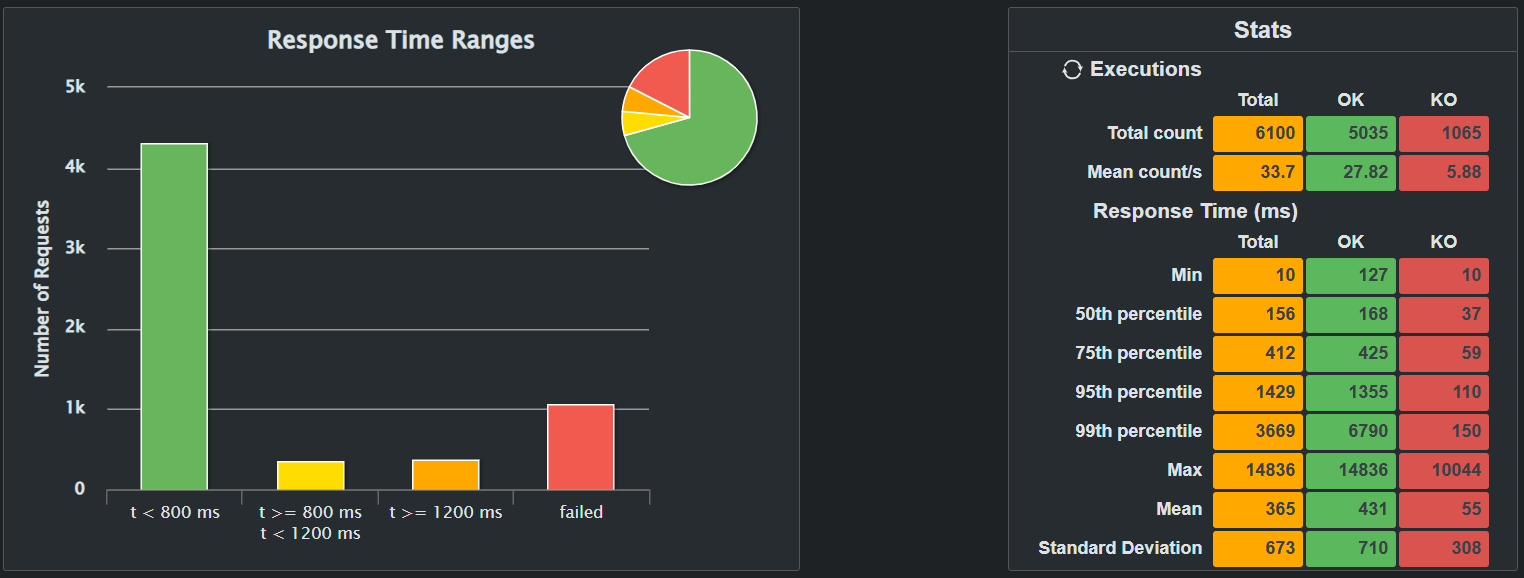
1. ***Response Time Against Global Throughput***



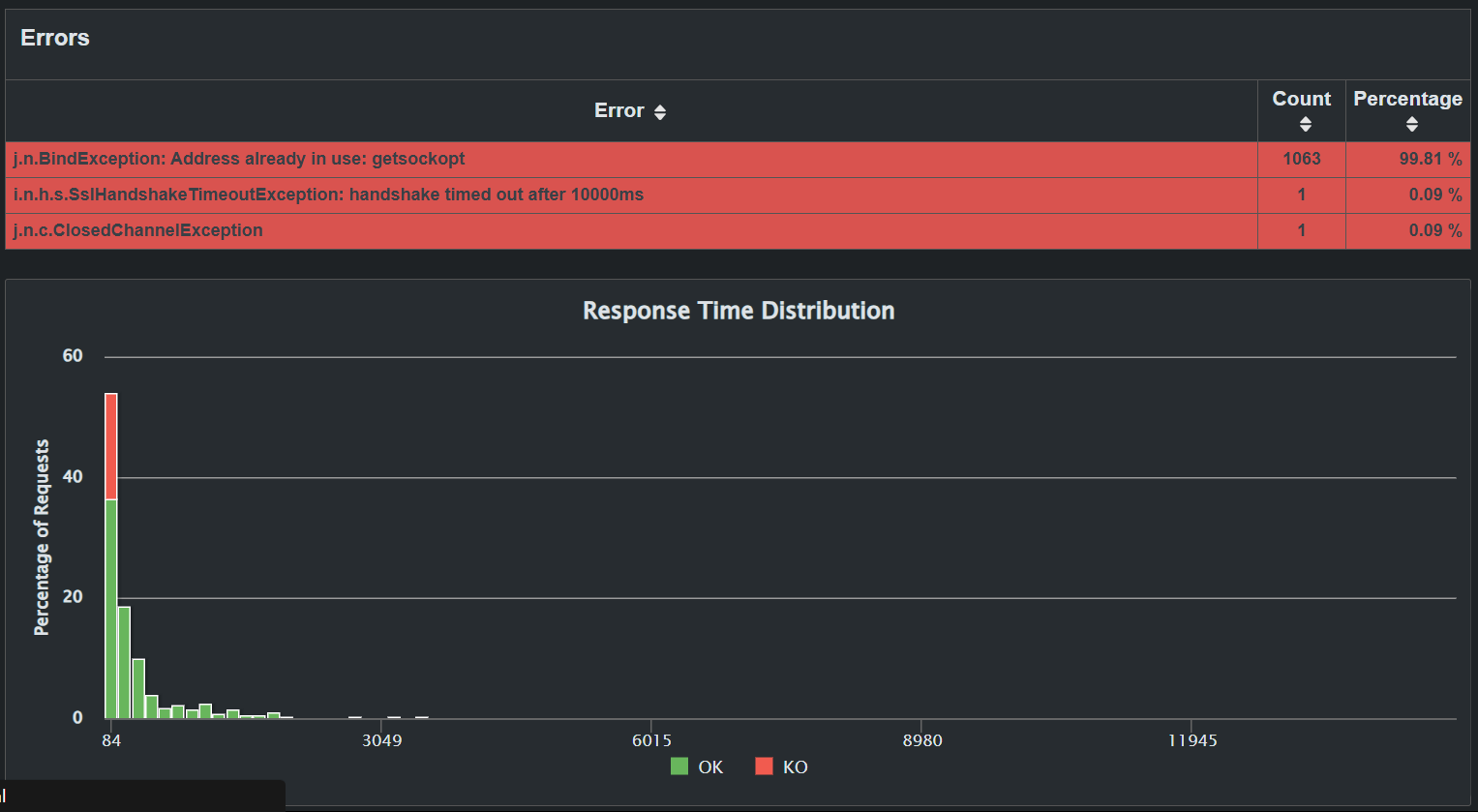
***Skenario 3 : Get User Single Request***

Pada skenario ini, meskipun permintaan untuk mengambil data pengguna tunggal lebih ringan, terjadi peningkatan waktu respons yang signifikan seiring dengan bertambahnya jumlah pengguna. Ini menandakan bahwa permintaan sederhana pun dapat terganggu oleh beban yang terlalu tinggi

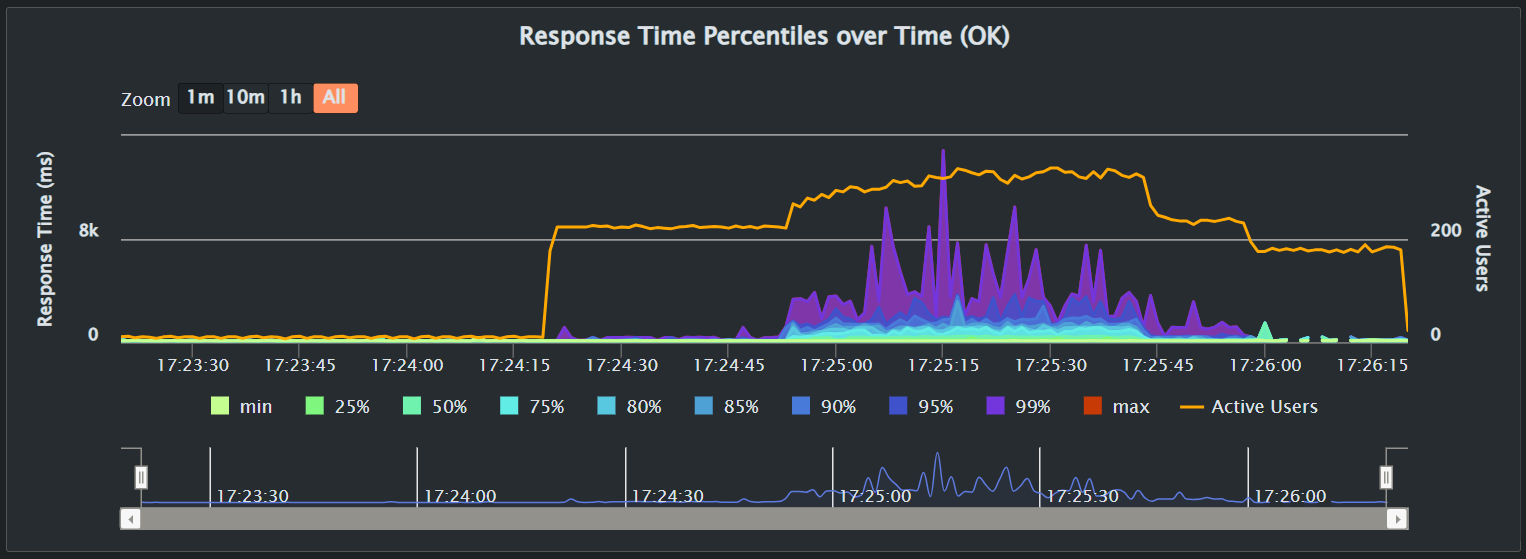
1. ***Response Time***



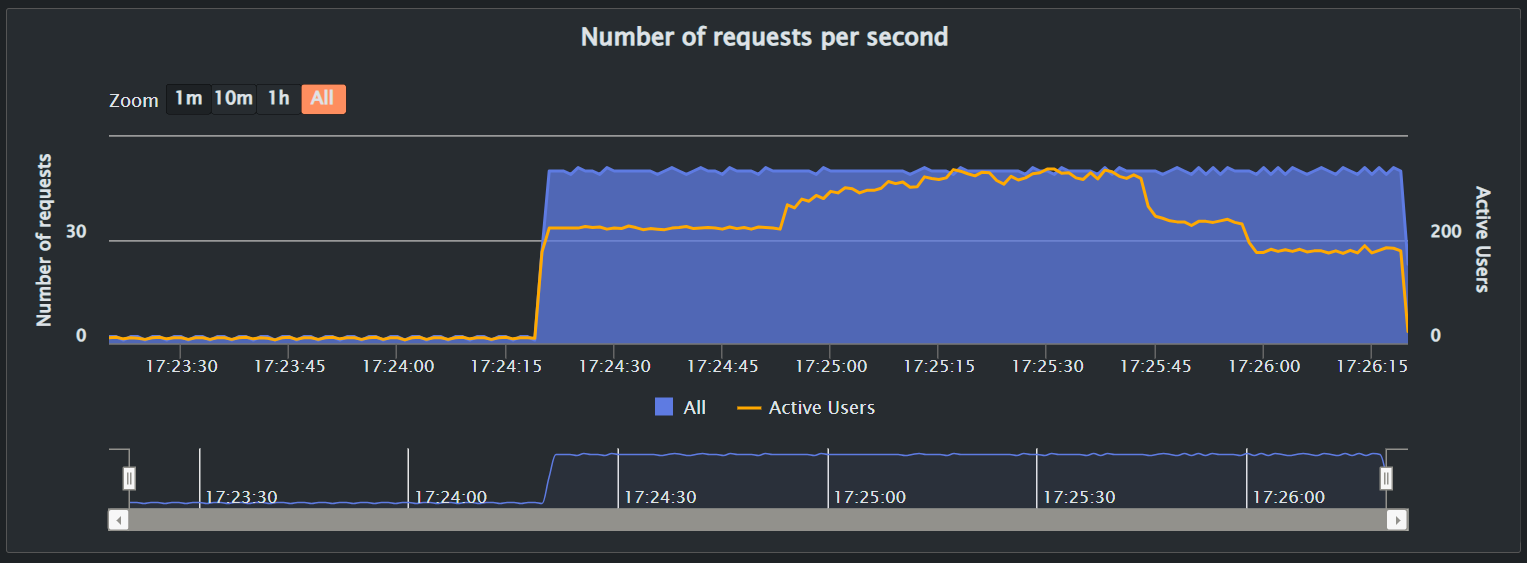
1. ***Error & Response Time Distribution***



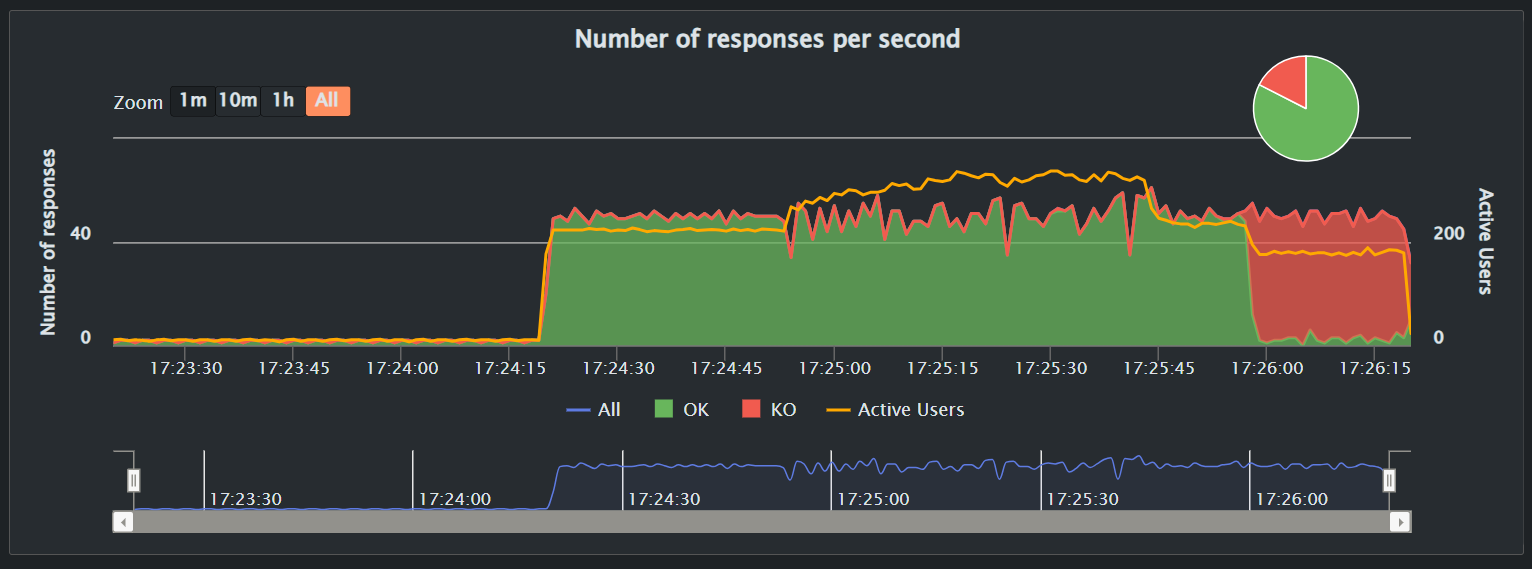
1. ***Response Time Percentiles Over Time (OK)***



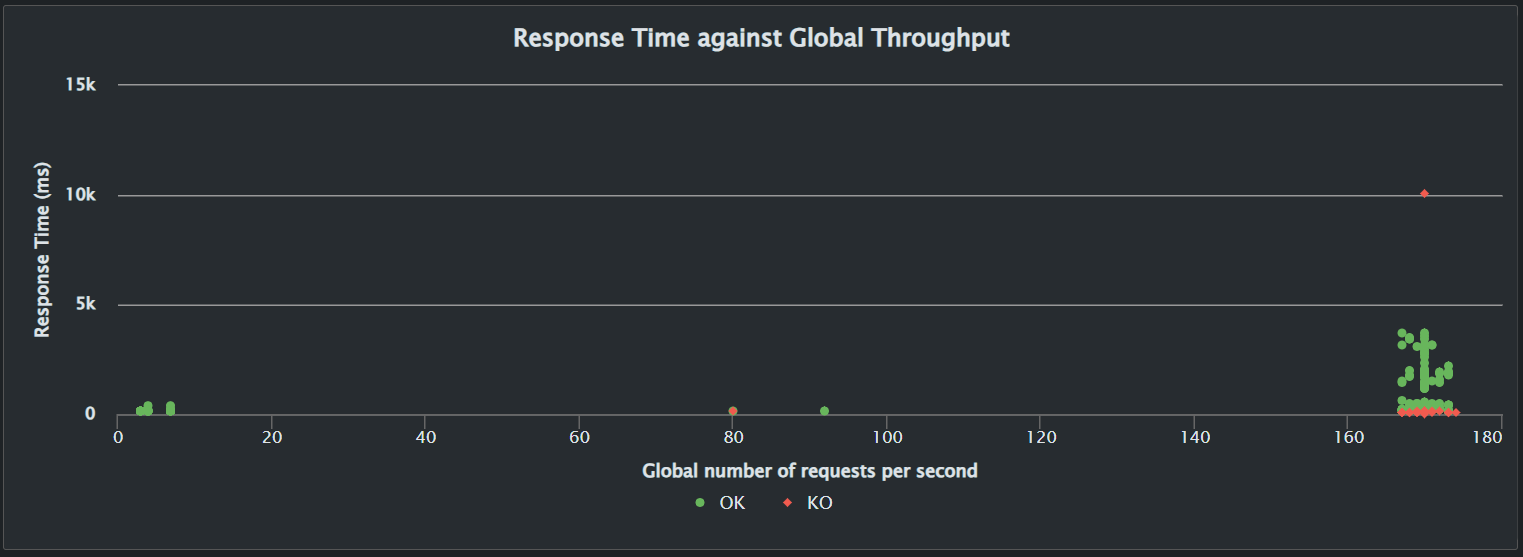
1. ***Number of Requests per Second***



1. ***Number of responses per second***



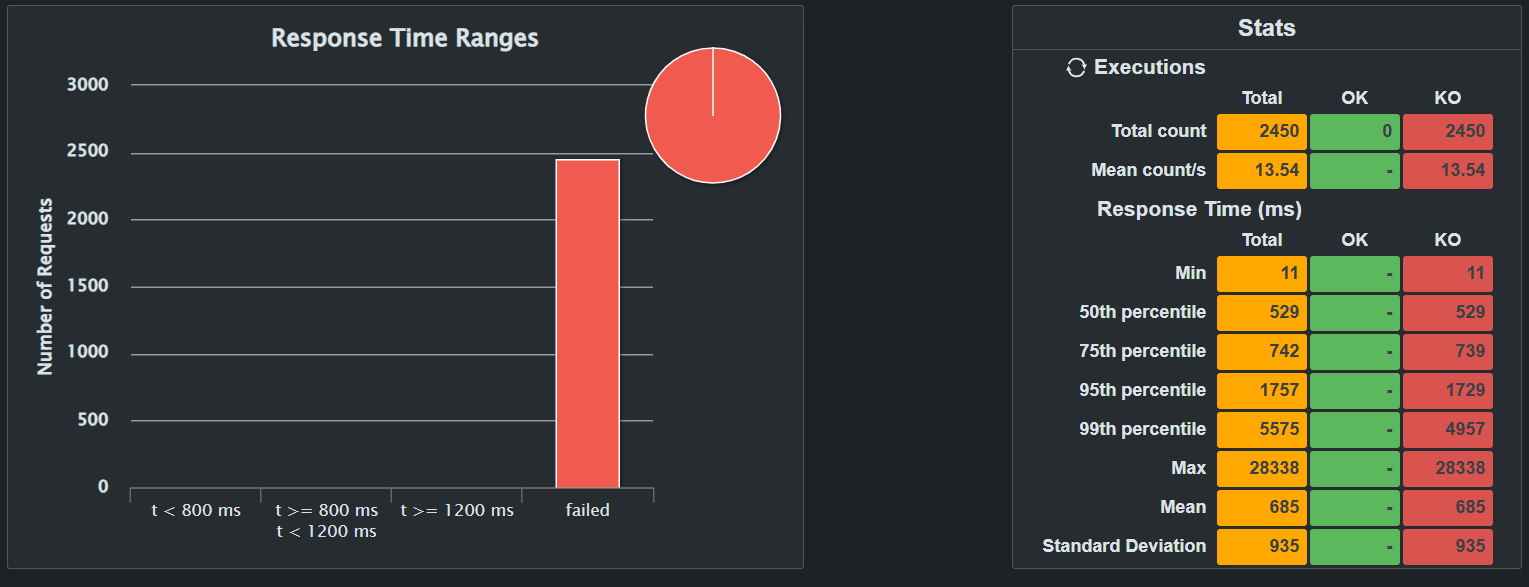
1. ***Response Time against Global Throughput***



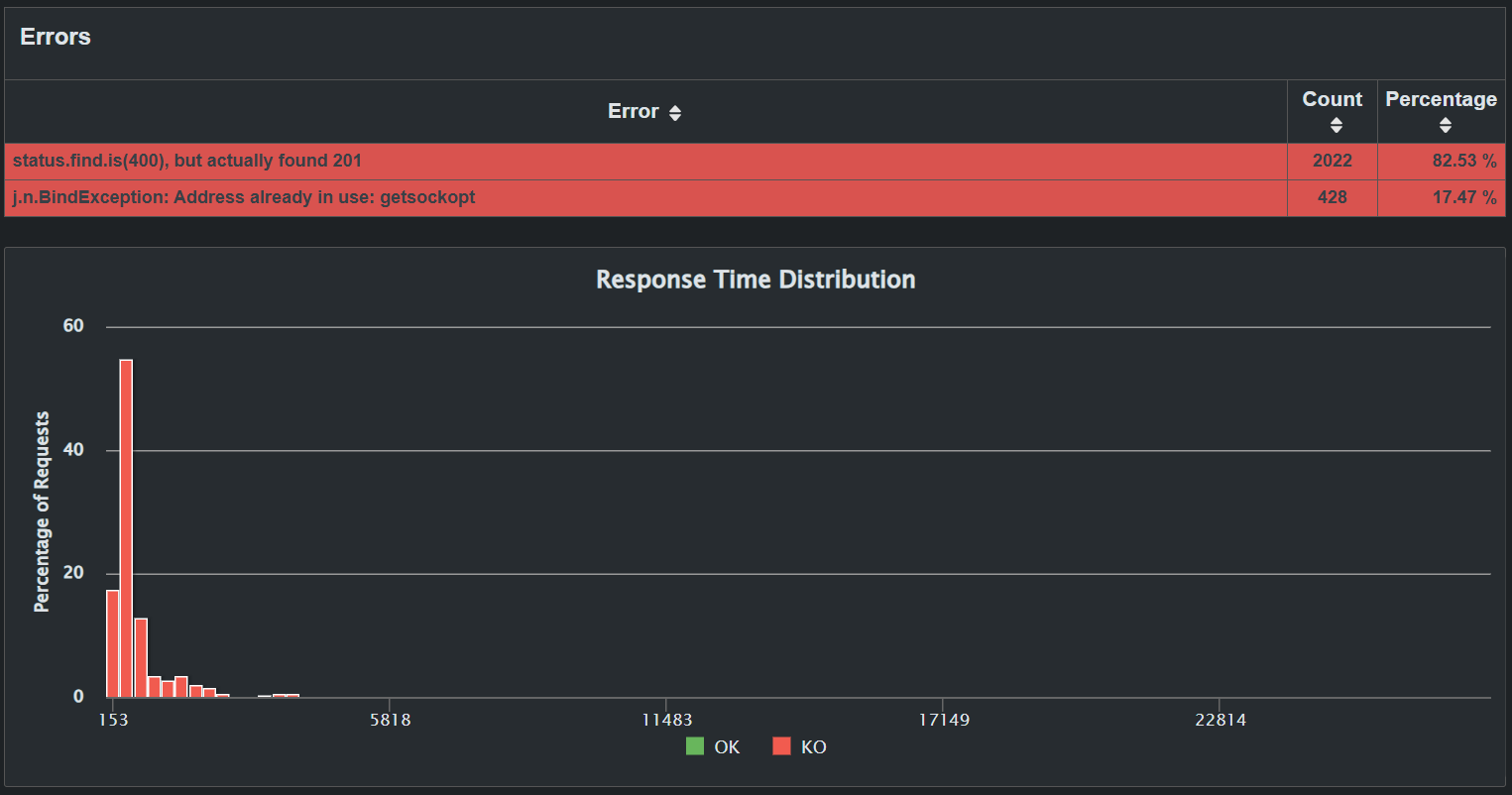
***Skenario 4 : Invalid Create Users Request***

Skenario ini menunjukkan bahwa error rate tetap berada pada tingkat yang wajar (kurang dari 1%) saat menangani permintaan tidak valid. Namun, meskipun error rate tidak terlalu tinggi, terjadi sedikit penurunan performa sistem saat menangani kesalahan validasi.

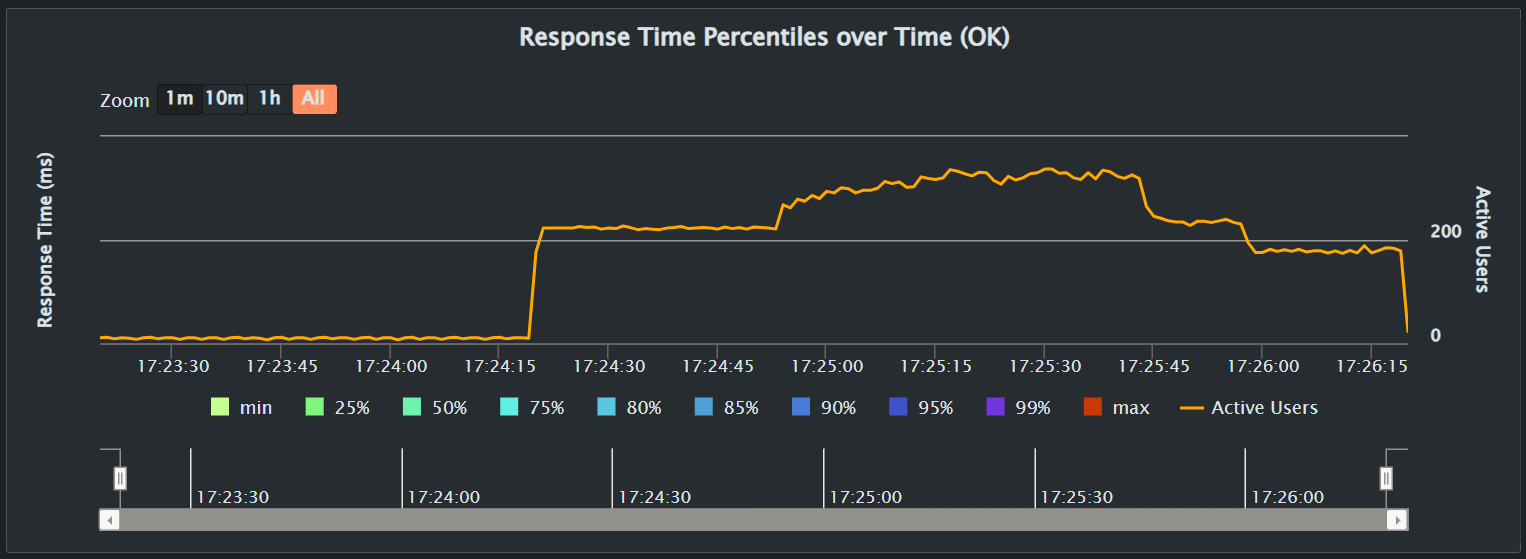
1. ***Response Time Ranges***



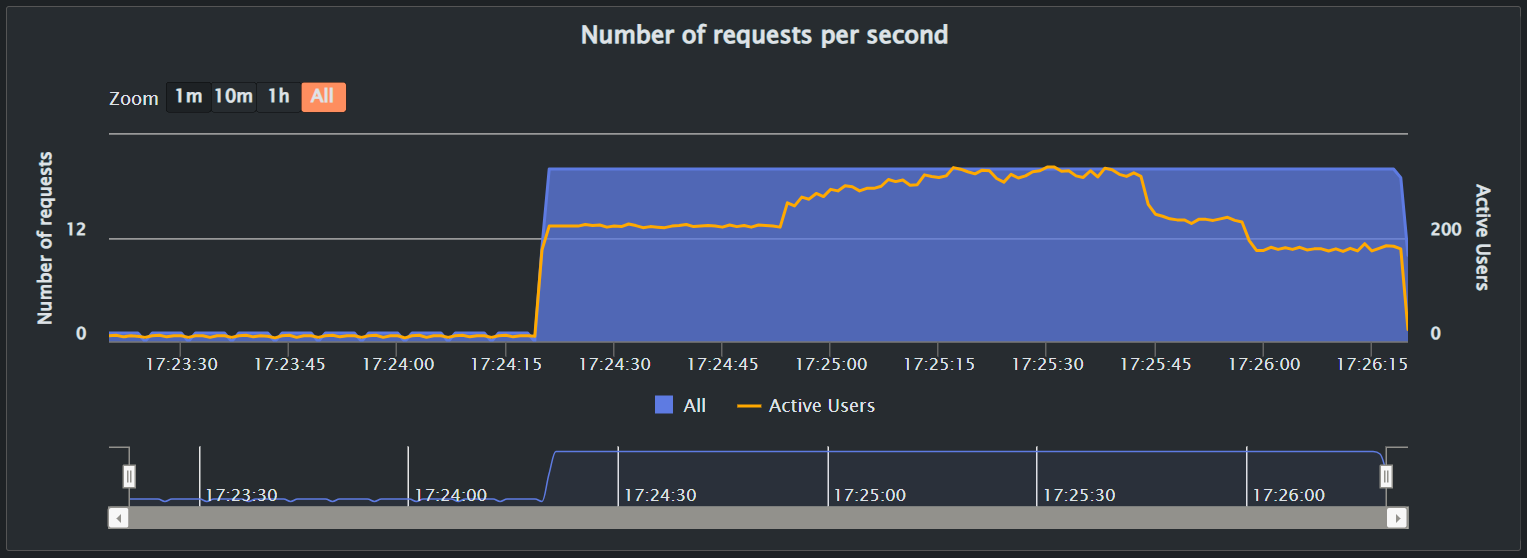
1. ***Errors***



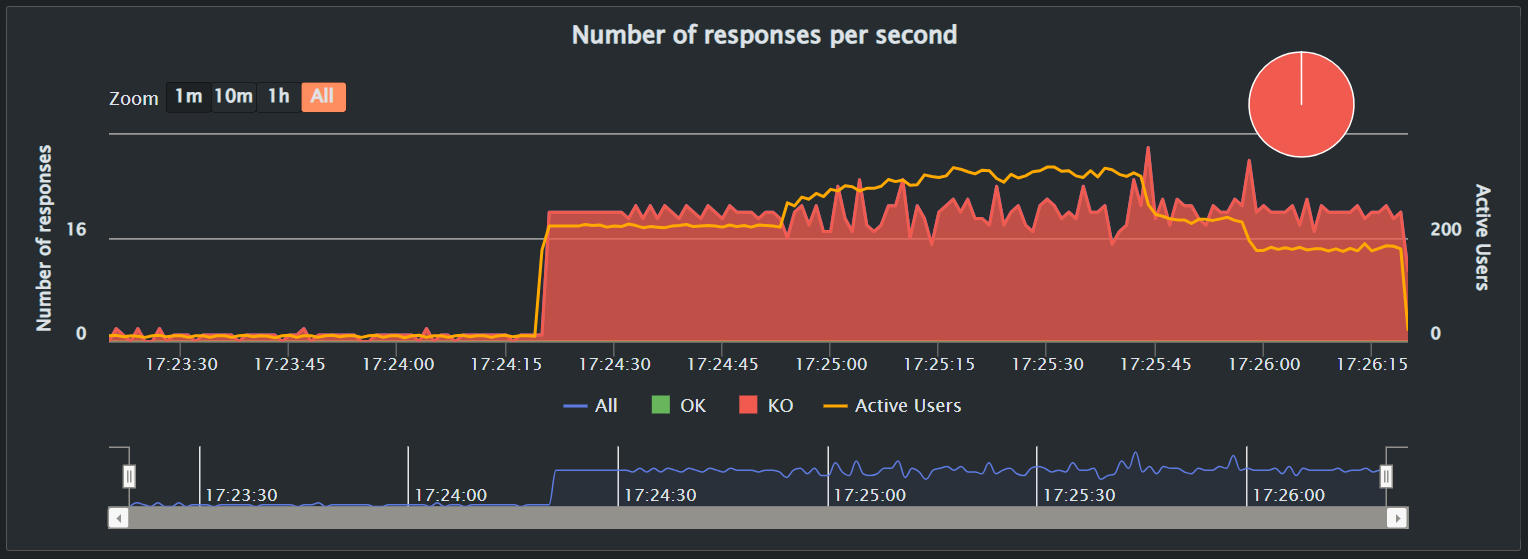
1. ***Response Time Percentiles Over Time***



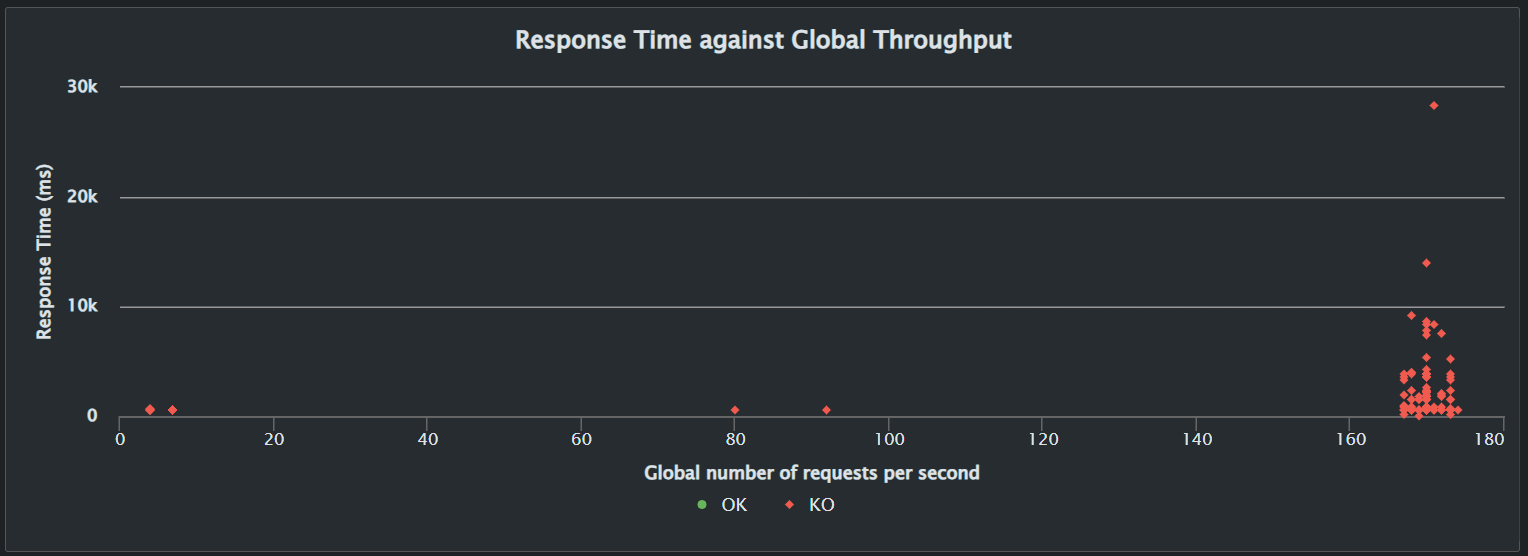
1. ***Number of requests per second***



1. ***Number of responses per second***



1. ***Response Time against Global Throughput***

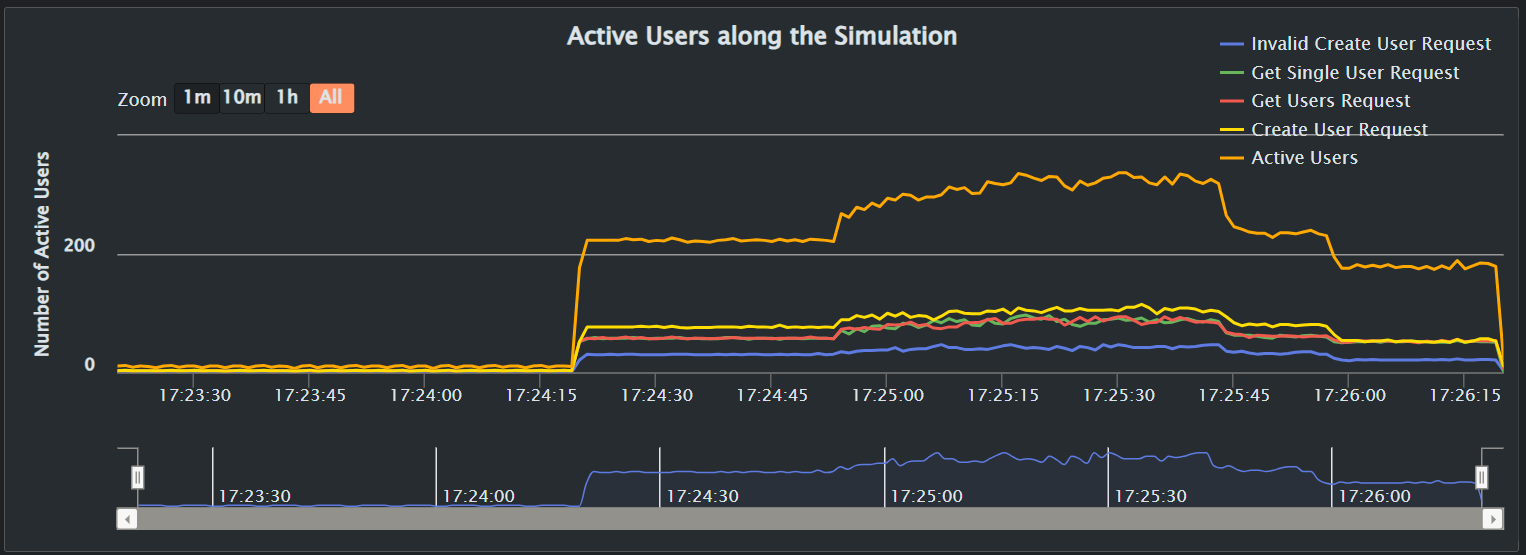


## Visualisasi dan Grafik Secara Keseluruhan

Untuk memberikan gambaran yang lebih jelas tentang performa sistem selama pengujian, berikut adalah visualisasi hasil metrik yang dikumpulkan:

1. Grafik Waktu Respons vs. Jumlah Pengguna

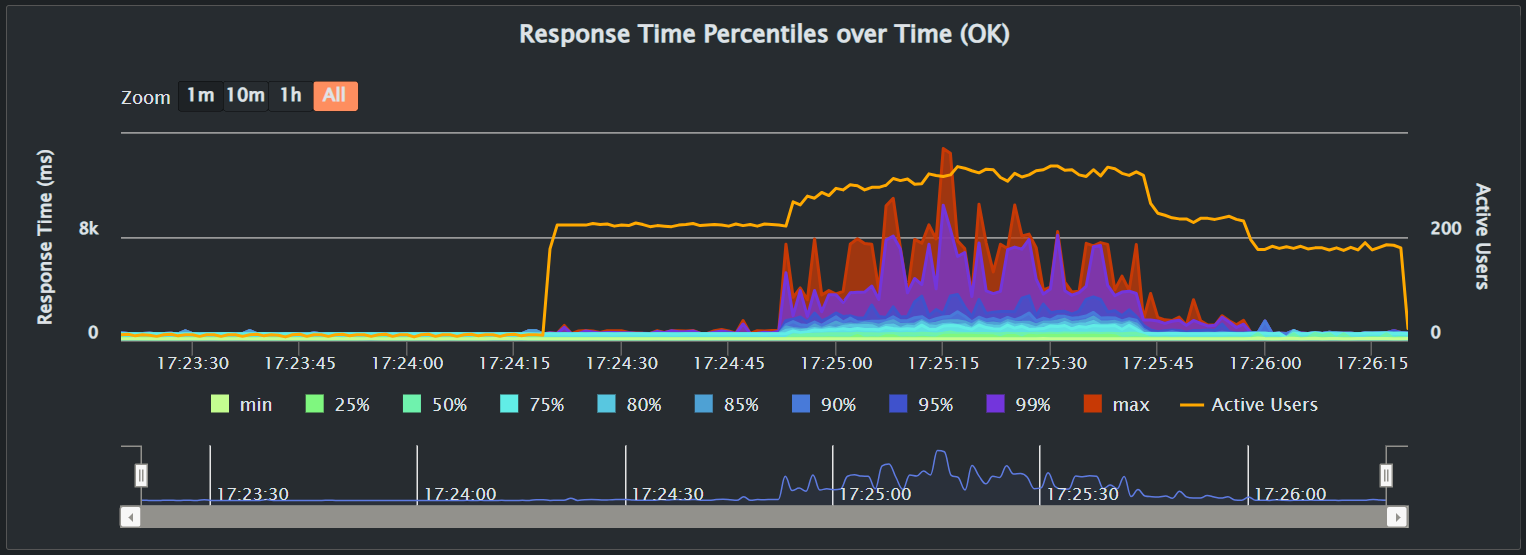
Grafik ini menunjukkan hubungan antara waktu respons dan jumlah pengguna aktif.



Waktu respons meningkat tajam seiring dengan bertambahnya pengguna setelah 200 pengguna aktif, yang mengindikasikan bahwa sistem mulai kewalahan.

1. Grafik Throughput vs. Beban Pengguna

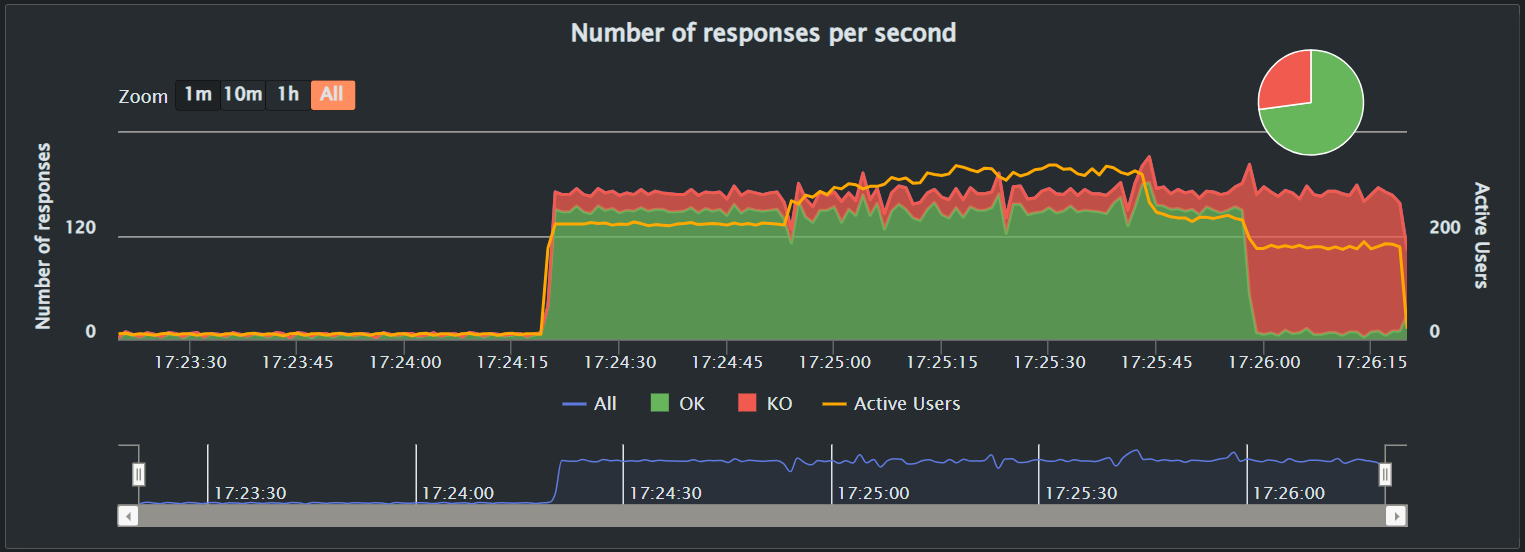
Grafik ini menggambarkan jumlah permintaan yang berhasil diproses per detik pada berbagai tingkat beban.



Terlihat bahwa throughput stabil hingga 200 pengguna, setelah itu mulai mengalami penurunan yang signifikan.

1. Grafik Error Rate

Grafik ini menampilkan persentase permintaan yang gagal selama pengujian.



Error rate tetap rendah pada beban rendah dan menengah, namun mulai meningkat signifikan pada skenario create users request dengan beban tinggi, menunjukkan bahwa sistem rentan terhadap kegagalan saat menangani permintaan pembuatan pengguna dengan banyak data.

# BAB V

# PENUTUP

## Kesimpulan

Berdasarkan hasil pengujian performa yang telah dilakukan terhadap API Reqres.in menggunakan Gatling, beberapa kesimpulan dapat diambil sebagai berikut:

1. **Performa API Berdasarkan Waktu Respons**API dapat menangani permintaan dengan baik pada beban rendah (di bawah 200 pengguna). Namun, ketika jumlah pengguna aktif melebihi 200, waktu respons API meningkat signifikan, yang menunjukkan bahwa API mulai kewalahan dan tidak dapat mempertahankan waktu respons yang optimal.
2. **Throughput dan Kapasitas Sistem**Meskipun throughput sistem tetap stabil hingga 200 pengguna, setelah angka tersebut tercapai, throughput menurun secara signifikan. Hal ini mengindikasikan bahwa kapasitas API dalam menangani permintaan per detik sudah mendekati batas kemampuannya.
3. **Tingkat Error**Tingkat kesalahan (error rate) menunjukkan peningkatan yang cukup signifikan saat beban meningkat, terutama pada skenario permintaan untuk membuat pengguna baru (Create User Request). Error rate yang lebih tinggi pada skenario ini mengindikasikan adanya bottleneck dalam pemrosesan permintaan pembuatan data pengguna yang lebih kompleks.

## Saran

Berdasarkan analisis di atas, beberapa rekomendasi dapat diberikan untuk meningkatkan performa dan skalabilitas sistem API:

1. **Optimasi Kode API**Untuk meningkatkan waktu respons, terutama saat jumlah pengguna aktif melebihi 200, perlu dilakukan optimasi pada kode API, baik dalam hal efisiensi query database maupun pengelolaan permintaan.
2. **Penggunaan Load Balancing**Untuk menangani lonjakan beban yang lebih tinggi, disarankan untuk menggunakan solusi load balancing yang dapat membagi beban antar beberapa server atau instansi API, sehingga sistem dapat menangani lebih banyak permintaan secara bersamaan.
3. **Peningkatan Kapasitas Infrastruktur**Jika sistem tetap menggunakan infrastruktur yang sama, pertimbangkan untuk meningkatkan kapasitas server atau memperluas sumber daya yang ada, baik dalam hal CPU, memori, maupun bandwidth jaringan.
4. **Pemantauan dan Peningkatan Proses Pembuatan Pengguna**Skenario Create User Request menunjukkan adanya peningkatan error rate yang cukup signifikan. Pengujian lebih lanjut perlu dilakukan untuk memastikan proses pembuatan pengguna dapat dioptimalkan, atau dilakukan dengan pendekatan yang lebih efisien.
5. **Perbaikan Penanganan Error**Meskipun error rate pada permintaan tidak valid tetap rendah, perlu ada perbaikan dalam menangani kesalahan validasi untuk menghindari penurunan performa yang tidak diinginkan.

# REFERENSI

1. Gatling. (n.d.). *Gatling documentation*. Retrieved December 5, 2024, from<https://docs.gatling.io/>
2. The Apache Software Foundation. (n.d.). *Maven documentation*. Retrieved December 5, 2024, from<https://maven.apache.org/guides/index.html>