

Izveštaj o testiranju performansi sistema (Locust)

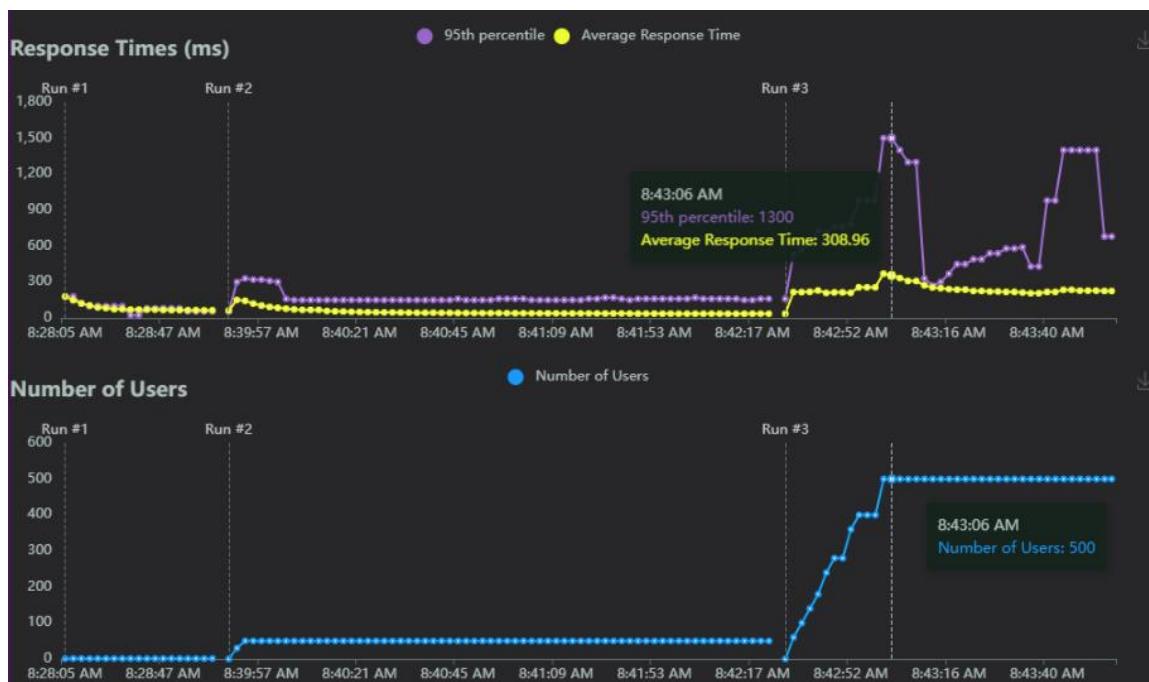
Ovaj izveštaj dokumentuje rezultate testiranja opterećenja sistema korišćenjem alata Locust. Testirano je 10 scenarija koji pokrivaju ključne funkcionalnosti aplikacije. Cilj testiranja bio je da se ispita ponašanje sistema pod različitim nivoima opterećenja: 500 korisnika (lagano), 1000 korisnika (srednje) i 5000 korisnika (jako), uz ramp-up period od 20 sekundi.

Testirani scenariji

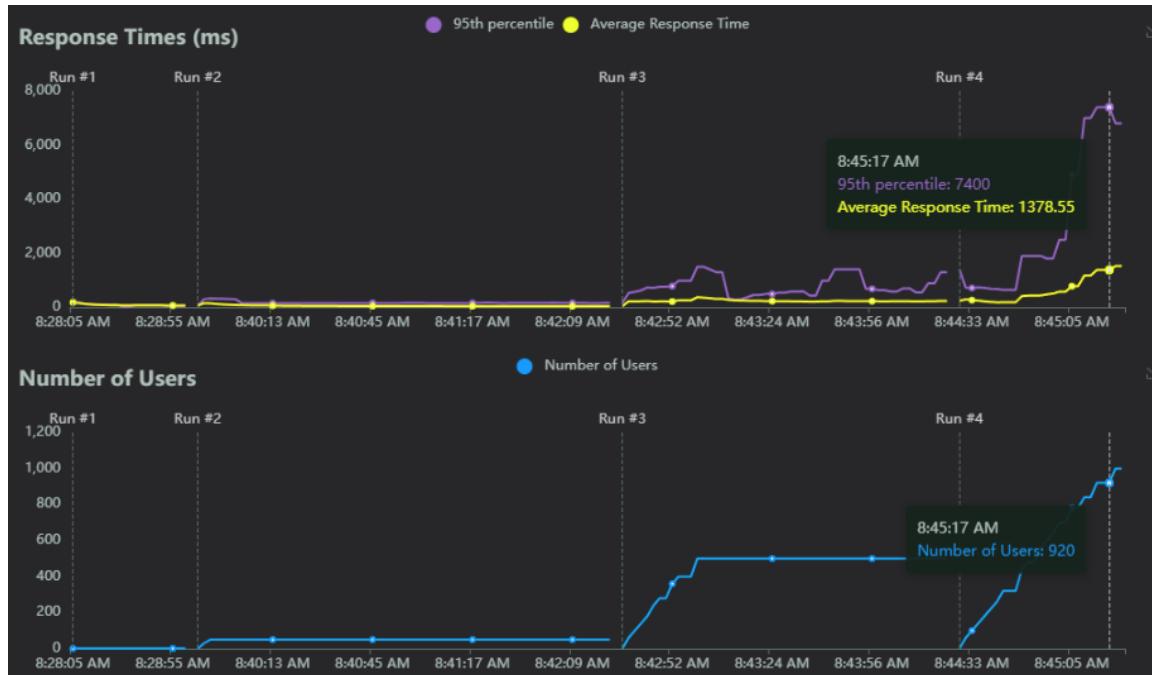
1. Login korisnika
2. Pregled vlasničkih domaćinstava
3. Pregled domaćinstava bez vlasnika
4. Podnošenje ownership requesta
5. Pregled mojih ownership requestova
6. Zakazivanje termina kod službenika
7. Pregled slobodnih termina
8. Admin – lista pending ownership requestova
9. Admin – obrada ownership requesta
10. Pregled termina službenika

Rezultati testiranja

Rezultati pokazuju da je sistem stabilan za 2.5 korisnika po sekundi (500 korisnika/20s), sa prosečnom uspešnošću većom od 95% i prosečnim vremenom odziva oko **300ms**.



Kod 50 korisnika dodatih po sekundi, tj. **stress testinga** (1000 korisnika/20s) uočen je pad performansi u scenarijima podnošenja **ownership requesta** i **zakazivanje termina** kod **službenika**, sa stopom neuspeha od 30–35%, a vreme odziva raste do ~1400ms, tj. dolazi do uskog grla.



Uočene greške

409 Conflict – greška nastaje pri konfliktima u podacima, npr. kada više korisnika pokuša da rezerviše isti termin ili više korisnika podnese ownership request za isto domaćinstvo.

Proširenje izveštaja – uticaj preporučenih promena

- Unique constraint (householdId, userId)**
 - Obezbedilo bi da jedan korisnik ne može više puta podneti zahtev za istu nekretninu.
 - Direktno bi smanjilo broj konflikata i 409 Conflict grešaka, čime bi se smanjio failure rate kod ownership request scenarija.
- Queue sistemi (RabbitMQ / Kafka) za zakazivanje termina**
 - Umesto sinhronog upisa u bazu, zahtevi bi se stavljali u red, a worker bi ih obrađivao jedan po jedan.
 - Ovo eliminiše race condition i smanjuje šansu da više korisnika rezerviše isti termin istovremeno.

- Time bi se drastično smanjile 409 Conflict greške u scenariju zakazivanja termina kod službenika.
 - 3. **Indeksi na kolonama (status, householdId, ownerId)**
 - Pretrage pending requestova i listanje domaćinstava postale bi višestruko brže (naročito pod velikim opterećenjem).
 - Smanjilo bi prosečno vreme odziva GET zahteva, rasteretilo CPU i smanjilo latency pri većem broju korisnika.
 - 4. **Horizontalno skaliranje + Redis cache**
 - Više backend instanci iza load balanca (Nginx ili Kubernetes) obezbedilo bi da sistem podnese nekoliko hiljada korisnika istovremeno.
 - Redis cache za GET rute (/no-owner, /owner, /pending) bi smanjio broj upita ka bazi i ubrzao odziv do 5-10x.
 - U kombinaciji, sistem bi mogao stabilno podržati 5k+ korisnika.
-

Tehnička specifikacija testnog okruženja

Testovi su izvođeni na računaru:

- **Model:** Lenovo Ideapad 5 Pro
- **CPU:** AMD Ryzen 7 (8 jezgara)
- **RAM:** 16 GB DDR4
- **Disk:** NVMe SSD 512 GB
- **OS:** Windows 11 + Docker Desktop (Linux containers)
- **Backend:** Spring Boot 3 + PostgreSQL (Docker) + Nginx proxy
- **Load test alat:** Locust (Python 3.11, web UI)