Napredne veb tehnologije

Izveštaj o testiranju projekta

Autor:

Dušan Bibin

SV65/2020 – student 1

Fakultet tehničkih nauka u Novom Sadu

Datum: 05.10.2025.

Rezime:

Izveštaj detaljno dokumentuje proces testiranja performansi sistema za pregled potrošnje struje u domaćinstvima kao i svih ostalih propratnih funkcionalnosti.

Izveštaj se fokusira na testiranje opterećenja sistema za 10 različitih scenarija korišćenja. Korišćen je alat za testiranje opterećenja sistema Locust. Glavni cilj je bio evaluacija performansi sistema u uslovima sa povećanim brojem korisnika.

Sadržaj

1. Uvod

Ovaj izveštaj dokumentuje proces testiranja softverskog rešenja za pregled potrošnje struje u domaćinstvima i ostalih funkcionalnosti, u skladu sa specifikacijom projekta. Projekat je bio realizovan kao grupni projekat sa tri studenta, pri čemu je svaki student bio odgovoran za određene funkcionalnosti softvera.

Glavni zadatak bio je projektovanje i implementacija softvera koji omogućava vlasnicima pametnih kuća da imaju uvid u potrošnju električne energije u realnom vremenu i pristup trendovima potrošnje iz prošlosti.

U ovom izveštaju će biti detaljno opisani koraci testiranja opterećenosti softverskog rešenja, uključujući testne scenarije, alate i tehnike korišćene za testiranje performansi sistema pametne kuće.

Platforma razlikuje sledeće vrste korisnika:

- Redovan registrovan korisnik (građanin): može da registruje nova domaćinstva, zatraži pristup podacima o potrošnji električne energije nekog domaćinstva, kao i da nadzire potrošnju električne energije domaćistva;
- Službenik: Službenik radi sa strankama i ima uvid u svoj raspored
- Administrator: ima pravo nadzora i upravljanja sistema u realnom vremenu; administrator ne može da izvršava nijednu funkcionalnost koja pripada redovnom korisniku, dok redovni korisnik ne može da izvršava nijednu funkcionalnost koja pripada administratoru;
- Neautentifikovani korisnik: Ima mogućnost registracije i prijave na sistem.

Konkretno moje odgovornosti u okviru projekta su bile:

- Inicijalno stanje sistema
- Registracija I prijava korisnika
- Podnošenje zahteva za potvrdu vlasništva domaćinstva
- Obrada zahteva za potvrdu vlasništva domaćinstva
- Pregled potrošnje električne energije domaćinstva
- Zakazivanje termina kod službenika

Kroz ovaj projekat, ciljamo na evaluaciju performansi sistema pametne kuće u različitim scenarijima korišćenja kako bismo identifikovali eventualne nedostatke ili oblasti za unapređenje, pružajući vlasnicima pametnih kuća pouzdanu platformu za praćenje i upravljanje njihovim uređajima.

2. Metodologija testiranja

Proces testiranja softverskog rešenja za monitoring pametne kuće obuhvatio je korišćenje različitih testnih scenarija, kao i primenu alata za testiranje performansi sistema.

1. Testni scenariji:
    - Scenario 1: Autentikacija korisnika
    - Scenario 2: Registracija korisnika
    - Scenario 3: Pregled detalja o domaćinstvu
    - Scenario 4: Zakazivanje termina kod službenika
    - Scenario 5: Pregled rasporeda službenika u datoj nedelji
    - Scenario 6: Podnošenje zahteva za potvrdu vlasništva domaćinstva
    - Scenario 7: Pregled potrošnje električne energije na mesečnom nivou
    - Scenario 8: Pregled potrošnje električne energije za toku poslednjeg N perioda
    - Scenario 9: Pretraga domaćinstava bez vlasnika putem unosa delimičnih adresa sa sugestijama
    - Scenario 10: Pretraga domaćinstava bez vlasnika putem vidljivog okvira Google mape

2. Alati za testiranje:
    - Locust framework u python. Korišćen je Locust za simulaciju opterećenja sistema kroz različite testne scenarije. Python skripte su razvijene kako bi se definisali i izvršili testni scenariji koristeći Locust. Locust framework je besplatan alat za testiranje performansi i opterećenja koji se koristi za simulaciju HTTP i drugih protokola. Testovi u Locustu mogu se pokrenuti iz komandne linije ili korišćenjem njegovog web korisničkog interfejsa. Protok, vreme odziva i greške mogu se pregledati u realnom vremenu i exportovati radi kasnije analize. Možete importovati standardne Python biblioteke u svoje testove. Locust generiše set test funkcija koje simuliraju veliki broj korisnika. Ovo omogućava određivanje glavne tačke prekida u pogledu performansi, bezbednosti i upravljanja opterećenjem aplikacije.

3. Rezultati testiranja

U ovom delu izveštaja analiziraćemo rezultate testiranja opterećenja Sistema korišćenjem alata Locust. Glavni cilj ovog testiranja bio je određivanje maksimalnog broja korisnika koji sistem može podneti bez gubitka performansi ili prekida rada. Kroz seriju testnih scenarija simulirali smo različite nivoe opterećenja na sistem kako bismo identifikovali tačke prekida i procenili skalabilnost Sistema pod različitim uslovima. Detaljna analiza rezultata svakog testnog scenarija pružiće uvid u performanse sistema u realnim uslovima opterećenja. Aplikacija je pokrenuta, I testovi za datu aplikaciju, na laptopu sa procesorom AMD Ryzen 4700u 2.0Ghz sa 8 pravih I 8 logickih jezgara, I 16 gb RAM memorije sa Ubuntu 22.04.01 sistemom. Zbog toga što su i aplikacija I testovi za istu pokrenuti na istom laptopu, rezultati testova će biti blago lošiji u odnosu na situaciju da je na jednoj mašini pokrenuta aplikacija, a na drugoj testovi. Svaki test je rađen 3 puta sa 20 korisnika (2 spawn rate), 200 korisnika (10 spawn rate) I 2000 korisnika (50 spawn rate).

3.1 Scenario 1: Autentikacija korisnika

Test 1: 20 Users, 2 Spawn rate

Test 2: 200 Users, 10 Spawn rate

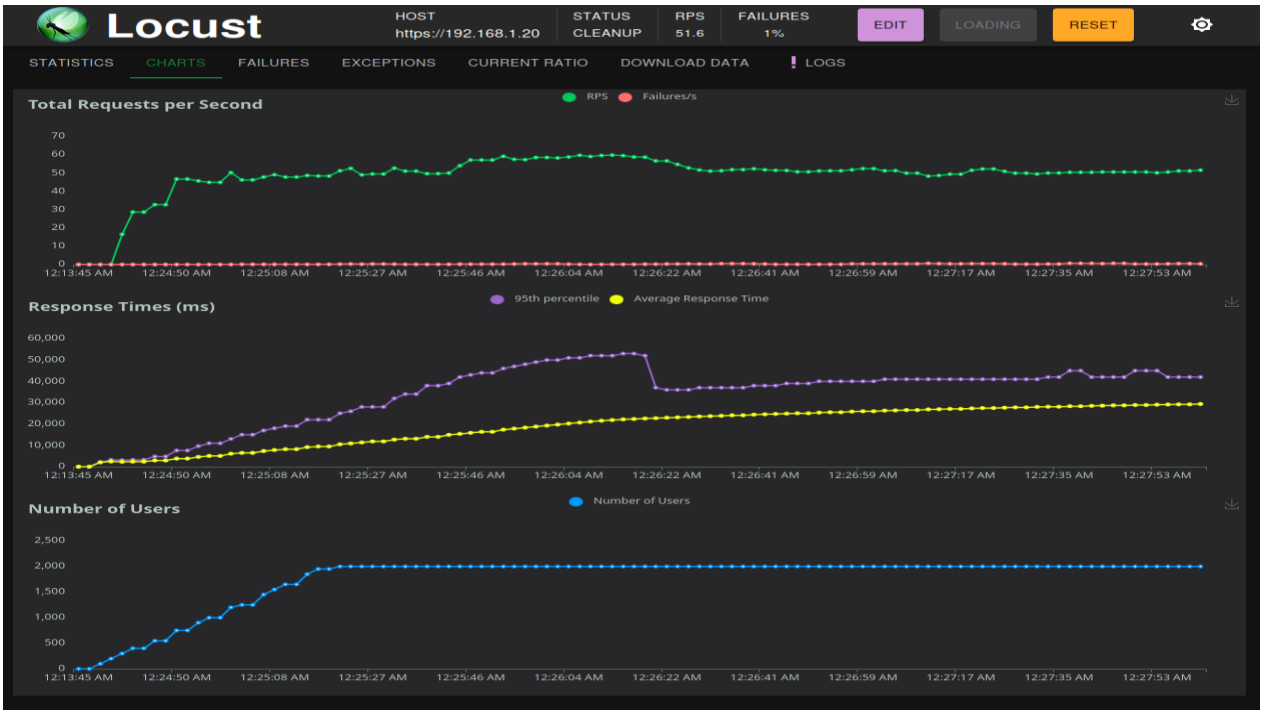
Test 3: 2000 Users, 50 Spawn rate

U autentikaciji je korišćen Bcrypt algoritam za heširanje. Manje više svi algoritmi za heširanje rade tako što namerno usporavaju proces heširanja tako da bi sprečili brute force napade. Vidimo da sa povećanjem istovremenih poziva za autentikaciju povećava vreme odziva za zahtev. Jednu stvar koju sam primetio je da sam 2 puta pristupao bazi nepotrebno što je donekle pomoglo u performansama. Takođe dodato je indeksiranje za mejlove u bazi. BCrypt ima *cost factor* koji je broj iteracija heširanja koji algoritam ponavlja. To je bilo moguće smanjiti ali to onda kompromituje sigurnost lozinki od brute force napada pa je stavljen na vrednost 10.

3.2 Scenario 2: Registracija korisnika

Test 1: 20 Users, 2 Spawn rate

Test 2: 200 Users, 10 Spawn rate



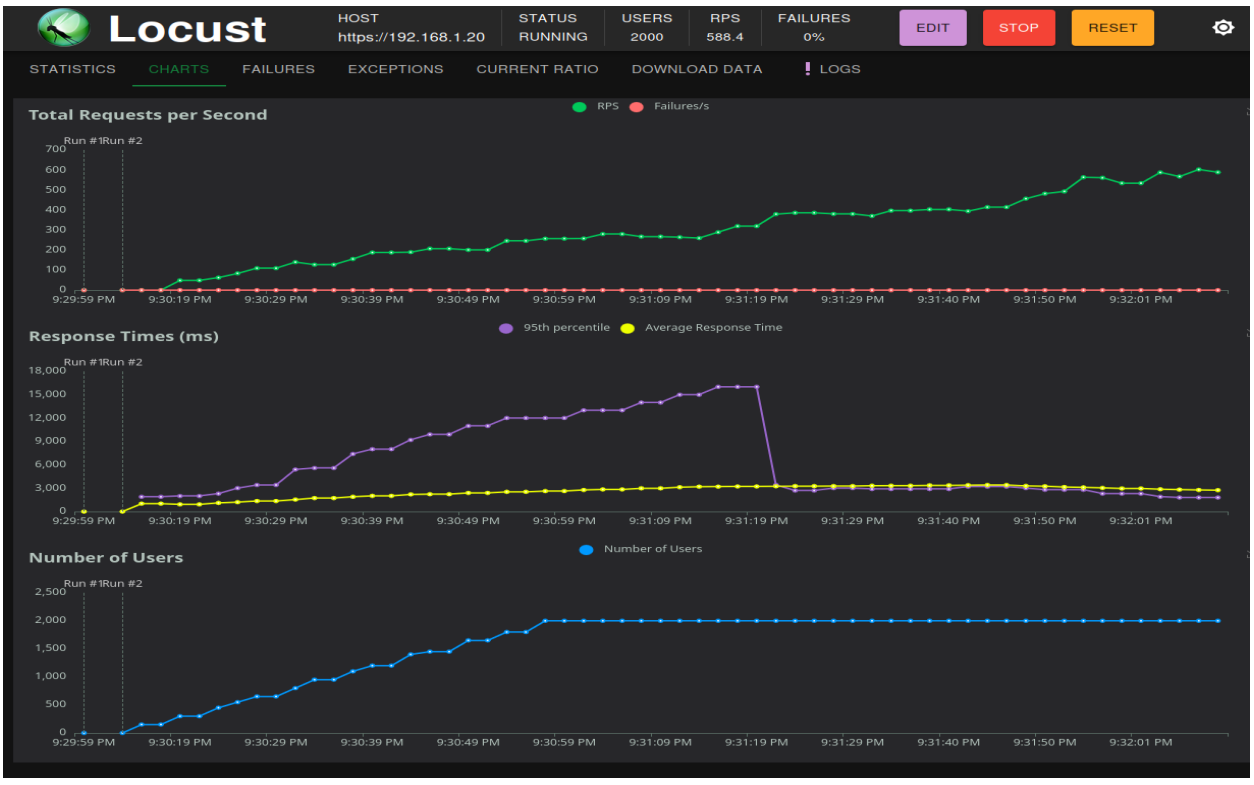
Test 3: 2000 Users, 50 Spawn rate

Registracija zahteva unos profilne slike, pa je ona zajedno sa heširanjem lozinki bila ključna za performnse. Test u locustu je sproveden sa slikom od 512x512 što nije dobar pokazatelj za performanse ali zbog ograničenog skladišta je to bio dobar kompromis jer je tako test mogao malo duže da traje.

3.3 Scenario 3: Pregled detalja o domaćinstvu

Test 1: 20 Users, 2 Spawn rate

Test 2: 200 Users, 10 Spawn rate

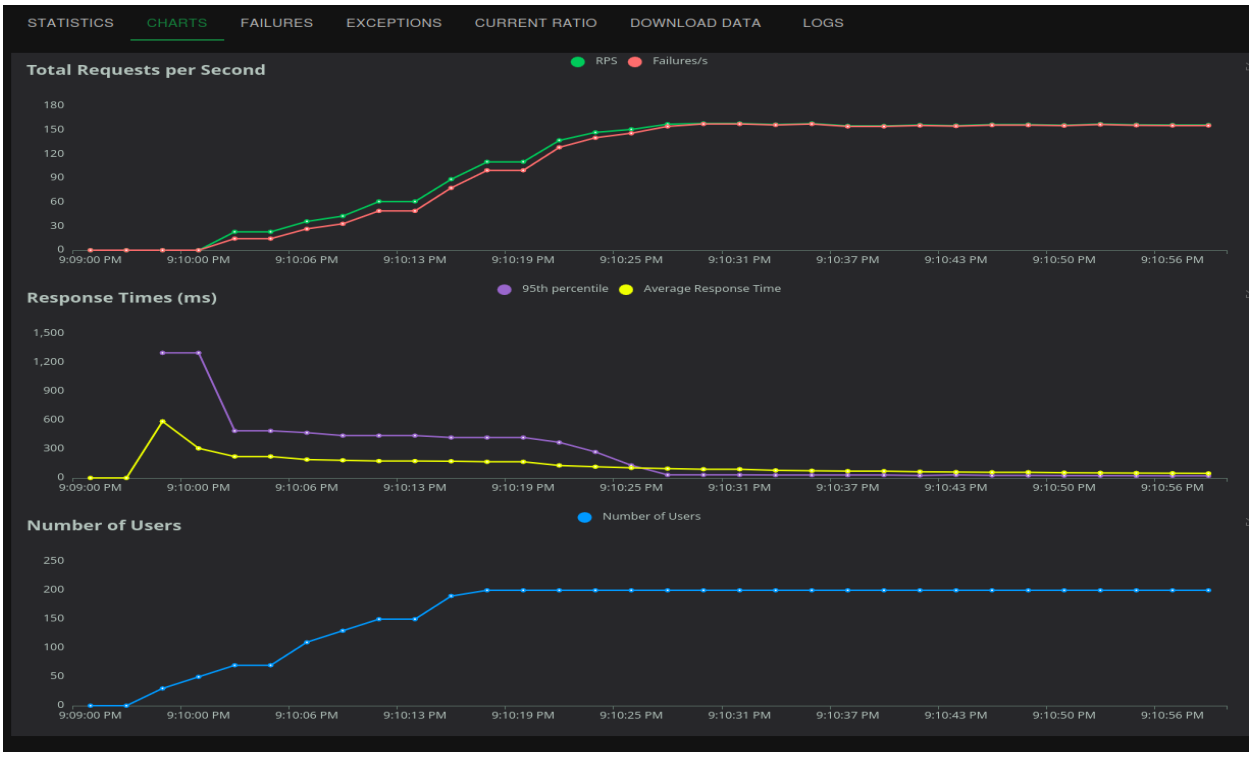


Test 3: 2000 Users, 50 Spawn rate

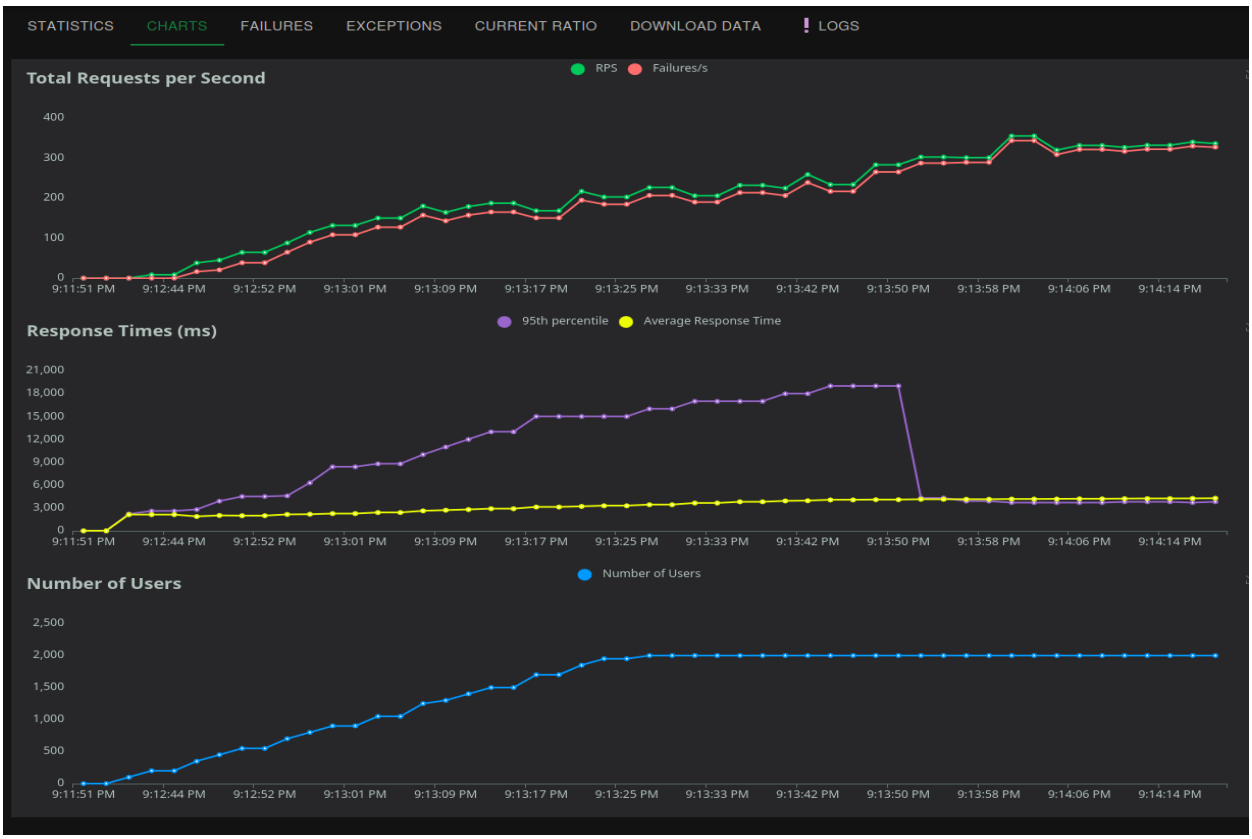
Pošto korisinici u sistemu nemaju mogućnost da menjaju podatke o svojim domaćinstvima, oni se ne menjaju, stoga je ovo bio dobar kandidat za keširanje, gde mnogo korisnika, koji stalo traže isti resurs umesto da sistem stalno traži u bazi podataka, on privremeno skladišti resurs u keš memoriji, koja je brža od obične memorije, na manji vremenski period i time dobijamo bolje performanse prilikom zahteva.

3.4 Scenario 4: Zakazivanje termina kod službenika

Test 1: 20 Users, 2 Spawn rate



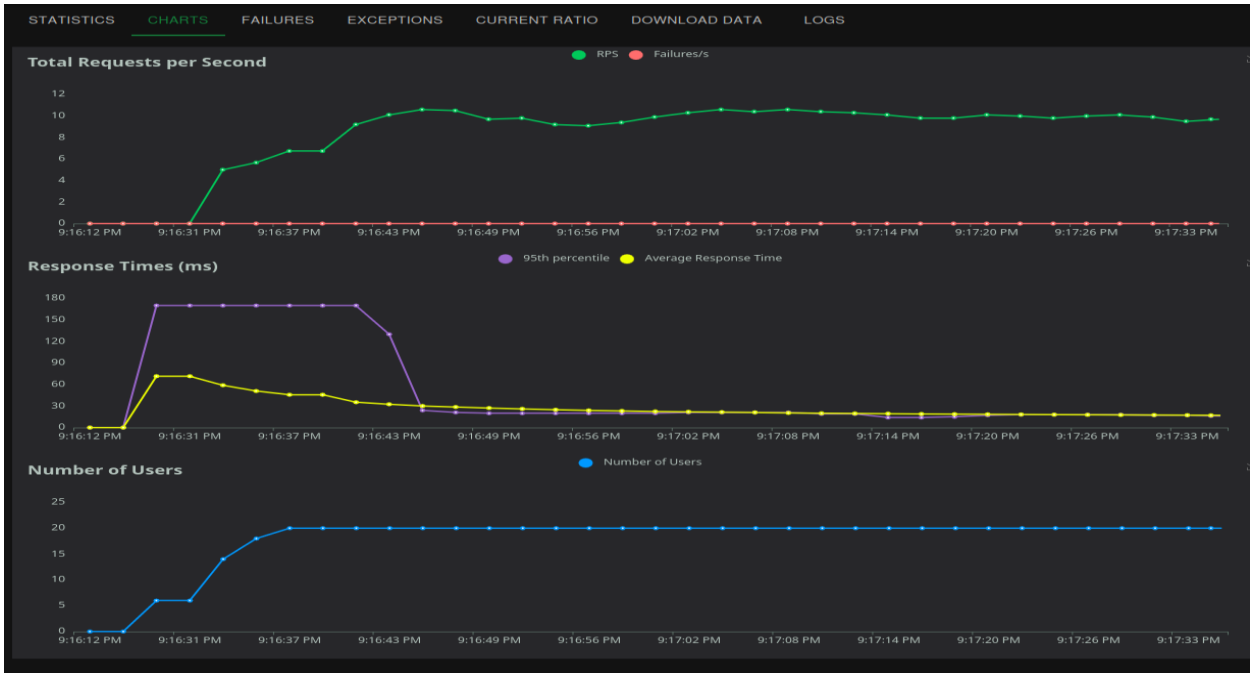
Test 2: 200 Users, 10 Spawn rate



Test 3: 2000 Users, 50 Spawn rate

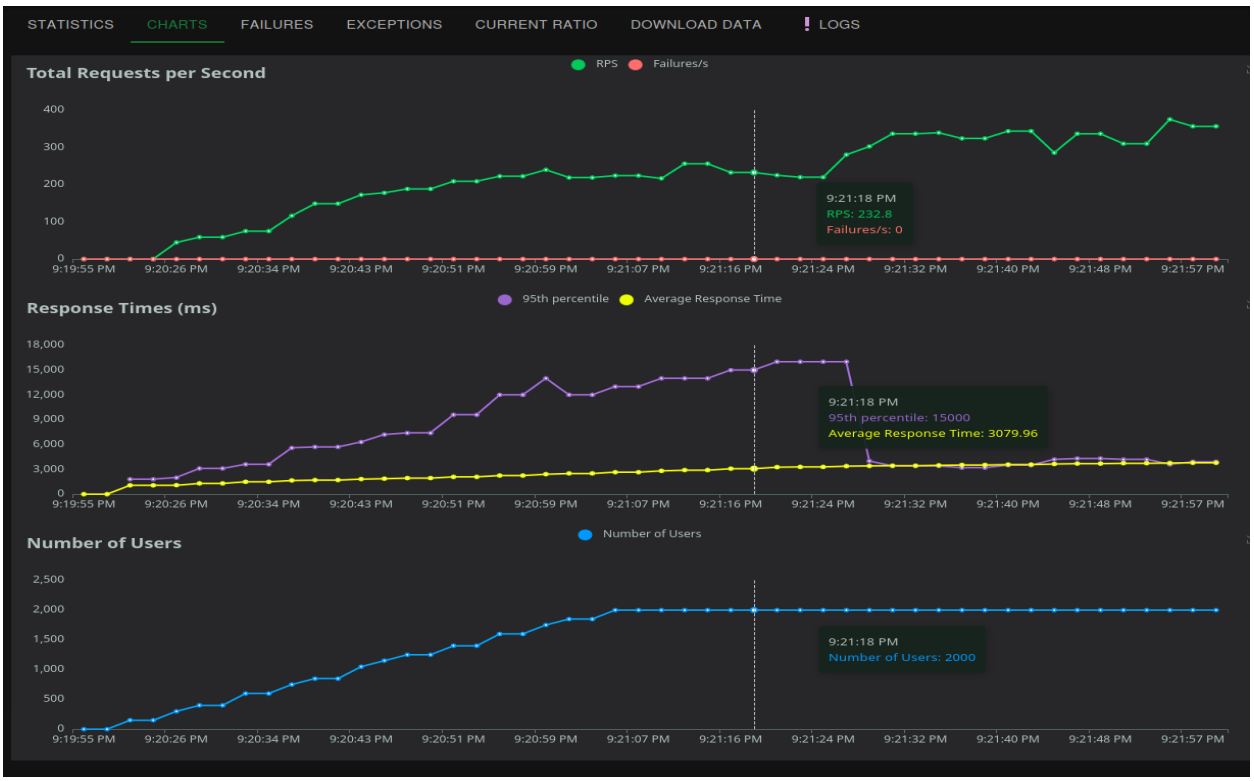
Pošto u ovoj funkcionalnosti, obični korisnici mogu da vide službenikov raspored za radnu nedelju I da rezervišu sastanak sa službenikom, može se dogoditi da 2 korisnika žele da zakažu isti termin, pa tada dolazi do trke podataka, gde se može dogoditi da 2 korisnika zakažu isti termin, a da sistem to ne shvati na vreme. To smo ovde rešili stavljanjem *uniqueConstraint* na par zvaničnikovog id-ja I termina slobodnog sastanka, takođe bismo prilikom zakazivanja novog sastanka umesto da pustimo sistem da sam završi sesiju, mi bi smo je završili odmah nakog zakazivanja

3.5 Scenario 5: Pregled rasporeda službenika u datoj nedelji



Test 1: 20 Users, 2 Spawn rate

Test 2: 200 Users, 10 Spawn rate

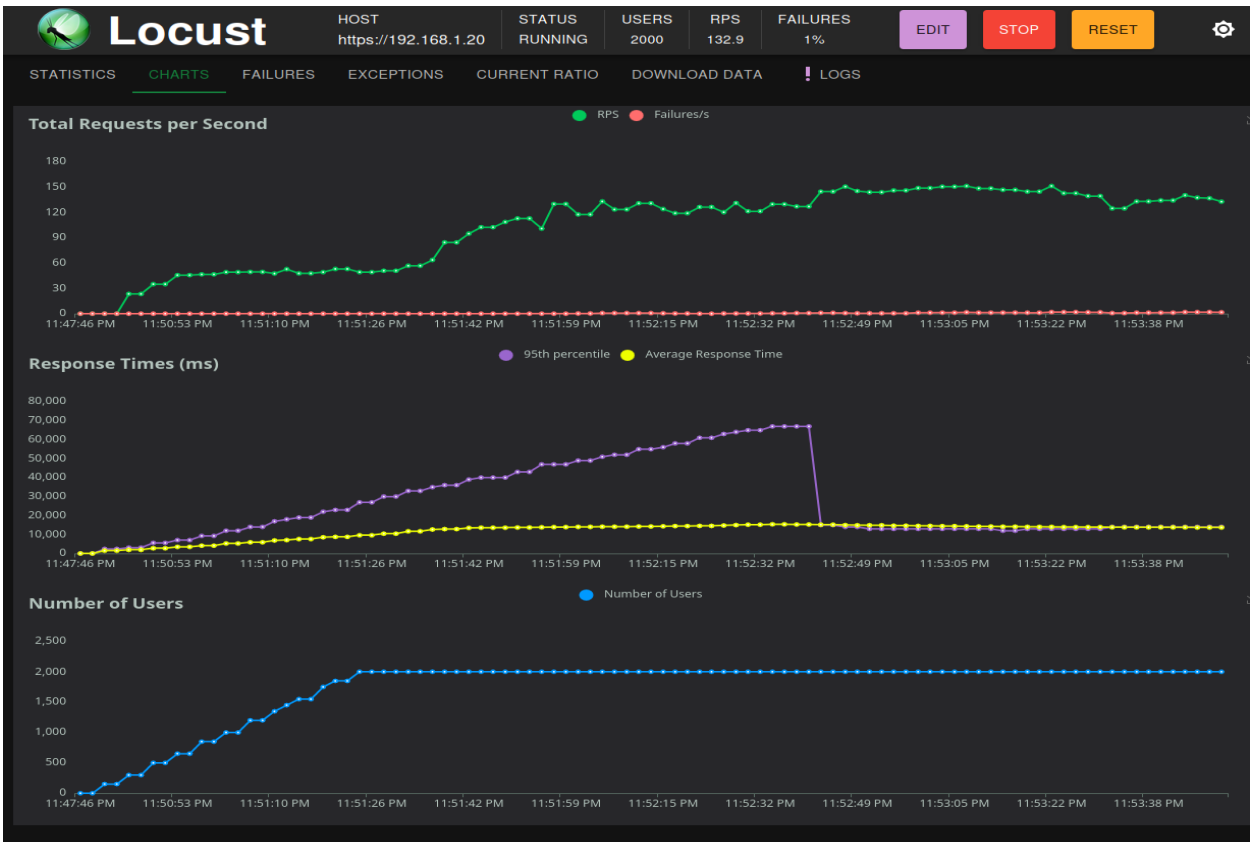


Test 3: 2000 Users, 50 Spawn rate

3.6 Scenario 6: Podnošenje zahteva za potvrdu vlasništva domaćinstva

Test 1: 20 Users, 2 Spawn rate

Test 2: 200 Users, 10 Spawn rate

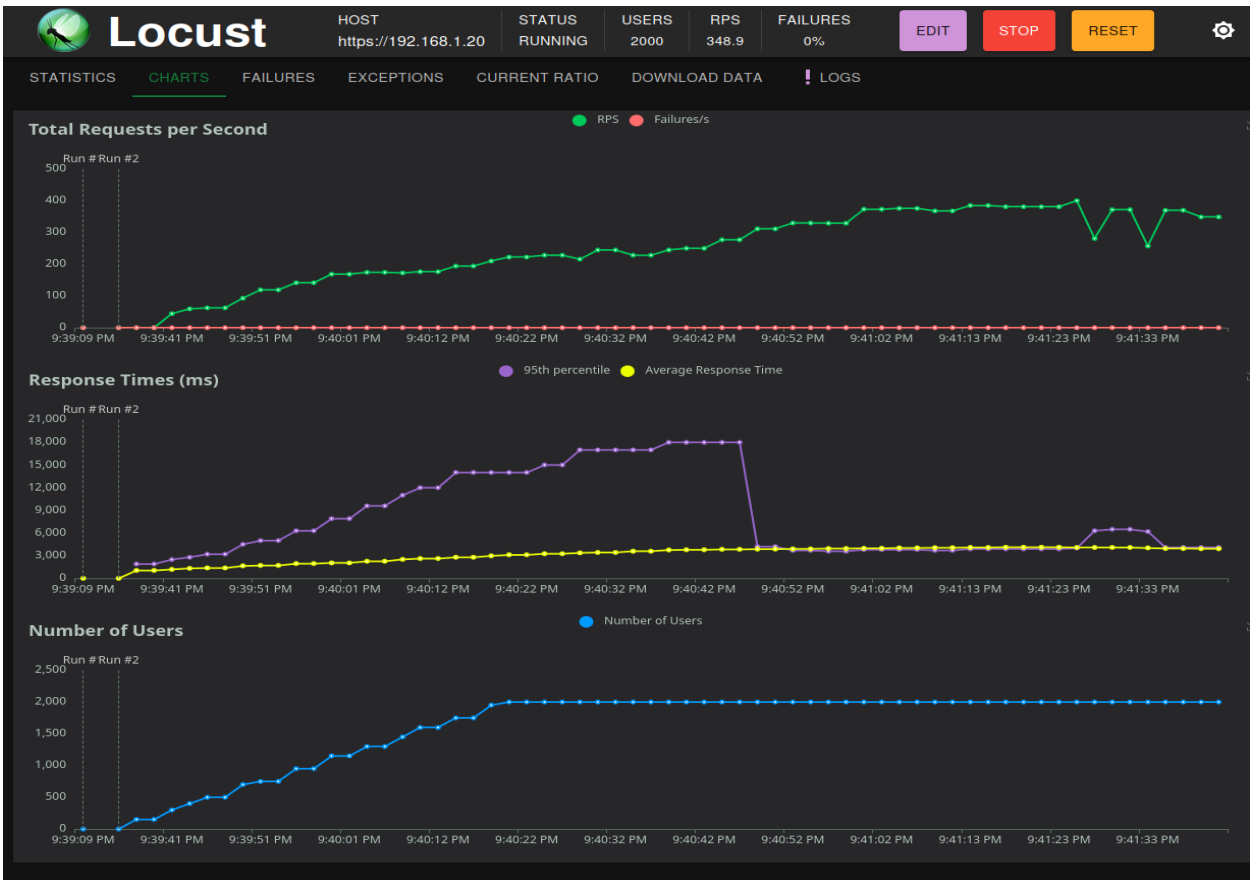


Test 3: 2000 Users, 50 Spawn rate

3.7 Scenario 7: Pregled potrošnje električne energije na mesečnom nivou

Test 1: 20 Users, 2 Spawn rate

Test 2: 200 Users, 10 Spawn rate



Test 3: 2000 Users, 50 Spawn rate

Ova funkcionalnost ima mogućnost da pregledamo potrošnju struje u godini tako što nam se prikazuje bar grafik sa 12 meseci ili potrošnju struje u mesecu. Pošto je manje bilo bitno za preciznost potrošnja na godišnjem nivou nego na mesecnom, tu sam koristio keširanje, kako bi manje imali poziva ka bazi

3.8 Scenario 8: Pregled potrošnje električne energije u datom poslednjem N periodu

Test 1: 20 Users, 2 Spawn rate

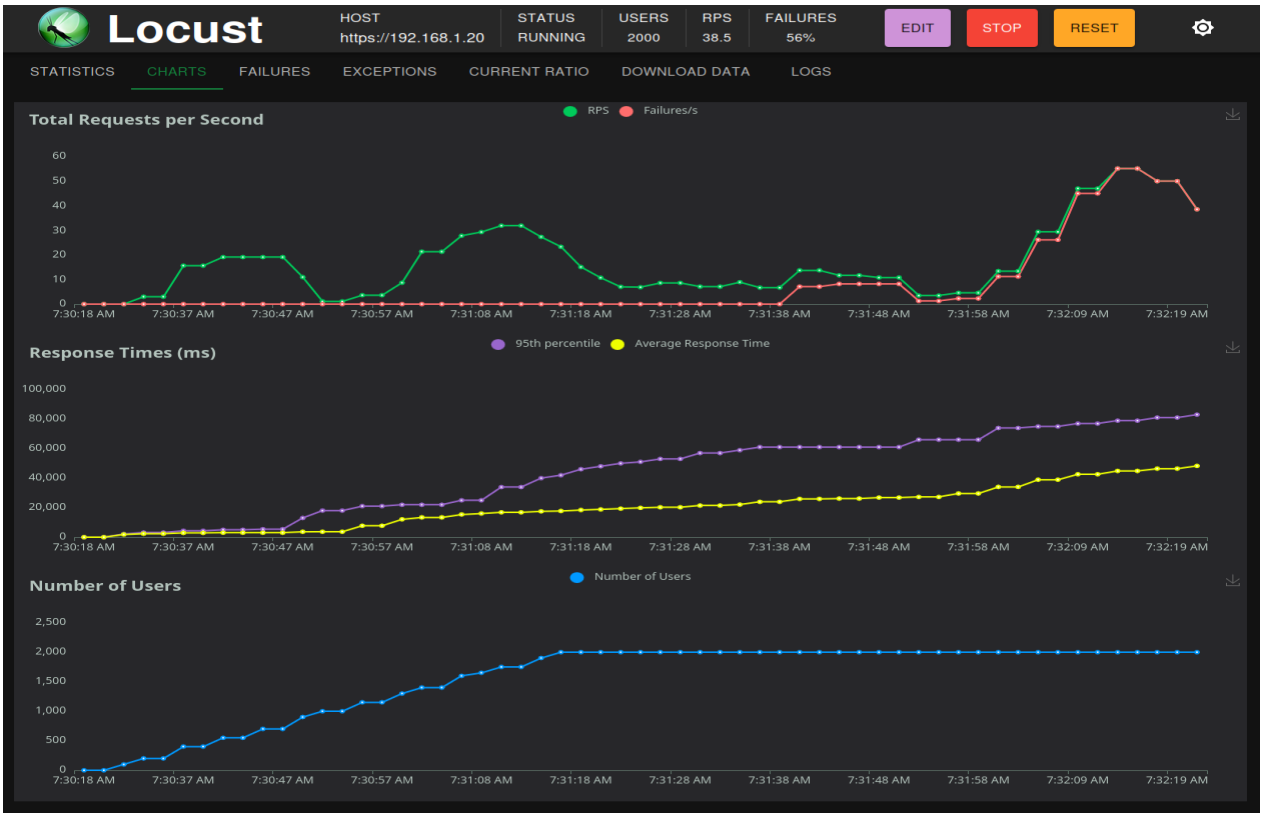
Test 2: 200 Users, 10 Spawn rate

Test 3: 2000 Users, 50 Spawn rate

3.9 Scenario 9: Pretraga domaćinstava bez vlasnika putem unosa delimičnih adresa sa sugestijama

Test 1: 20 Users, 2 Spawn rate

Test 2: 200 Users, 10 Spawn rate



Test 3: 2000 Users, 50 Spawn rate

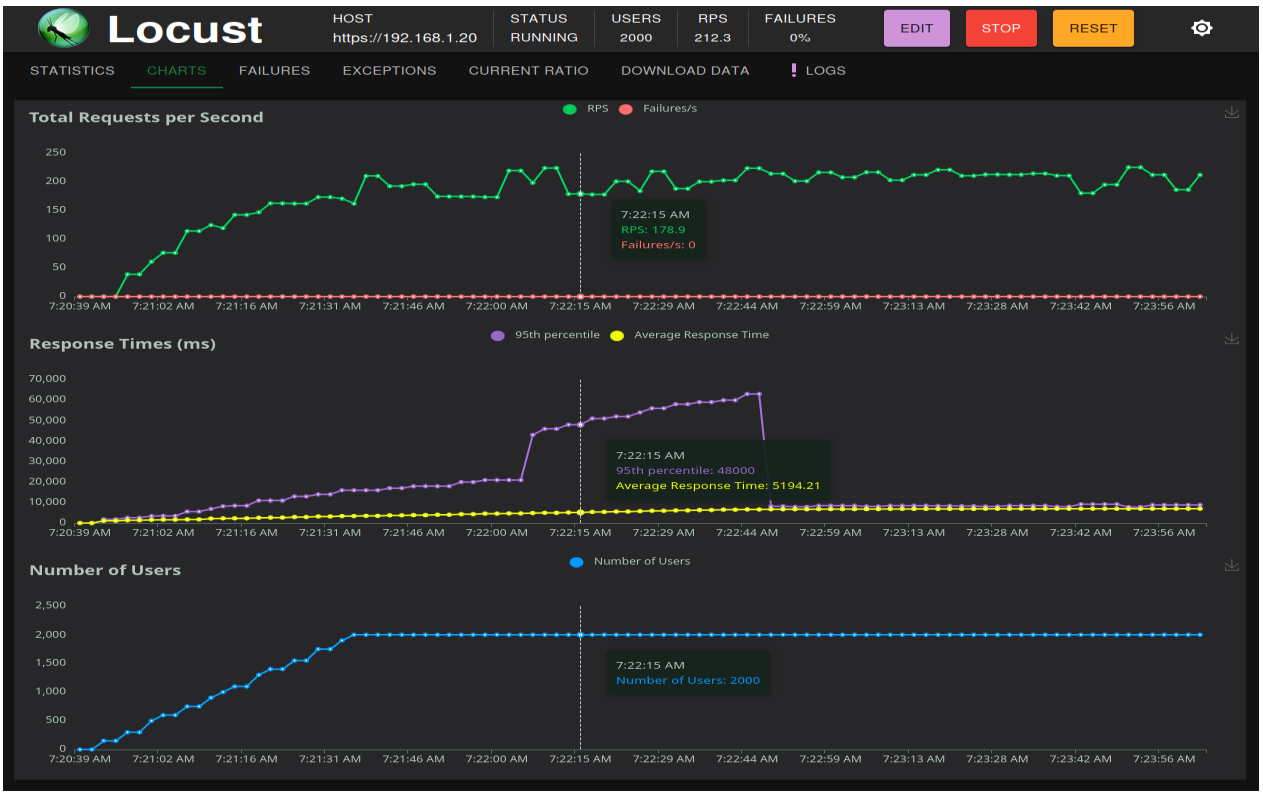
Elasticsearch je sistem za pretragu I analizu podatak u realnom vremenu, baziran na Apache Lucene. Omogućava brzu pretragu teksta. U mom kontekstu je korišćen u 2 slučaja. Prvi slučaj je za pretragu adresa, gde sam imao indeks gde sam čuvao adrese nekretnina I coordinate. Takođe sam dodao indekse za regije, opštine I naselja I vezao sam ih jedne sa drugima kako bih omogućio filtriranje rezultata pomoću regija opština I naselja. Elasticsearch je dosta efektivan po ceni da ima dosta resursa. Obično se stavlja da je alocirana *heap* memorija oko 50% ukupne, međutim kod mene je to bilo na 512mb jer sam bio ograničen resursima, I takođe u kombinaciji sa edgeNgram I fuzzysearch podešavanjima je čak dovelo do pucanja elasticsearch-a

3.10 Scenario 10: Pretraga domaćinstava bez vlasnika putem vidljivog okvira Google mape

Test 1: 20 Users, 2 Spawn rate

I ova funkcionalnost je koristila elasticsearch, ali preko *geospatial queries* gde elasticsearch deli dokumente sa koordinatama u različite ,,kante" kojih ima manje ili više u zavinosti od preciznosti koja joj se zada (u mom slučaju preciznost se definiše na osnovu nivou uveličanja na mapi). Menjao sam preciznost I broj rezultata koje dobijam za svaki bucket jer je bilo dosta slučaja kada je elasticsearch pucao.

Test 2: 200 Users, 10 Spawn rate



Test 3: 2000 Users, 50 Spawn rate