Napredne veb tehnologije

Izveštaj o testiranju sistema SmartHome

Autor:

Bogdan Janošević

SV65/2020 – student 2

Fakultet tehničkih nauka u Novom Sadu

Datum: 01. 31. 2024.

Rezime:

Izveštaj detaljno dokumentuje proces testiranja performansi sistema SmartHome u dva ključna segmenta: testiranje opterećenja sistema za česte scenarije korišćenja i testiranje performansi sistema prilikom rada sa simulatorima pametnih uređaja.

Prvi deo izveštaja fokusira se 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.

Drugi deo izveštaja istražuje performanse sistema prilikom rada sa simulatorima pametnih uređaja. Identifikovani su delovi sistema koji mogu dovesti do problema sa performansama prilikom povećanja broja uređaja koji komuniciraju sa platformom.

Sadržaj

1. Uvod...............................................................................................................3
2. Metodologija testiranja...................................................................4
3. Rezultati testiranja..........................................................................5
   1. Scenario 1: Registracija pametne nekretnine.........................................................5
   2. Scenario 2: Obrada zahteva za registraciju nekretnine...........................................7
   3. Scenario 3: Prikaz izveštaja o dostupnosti pametnih uređaja..............................11
   4. Scenario 4: Prikaz izveštaja o istoriji vrednosti lampe..........................................13
   5. Scenario 5: Prikaz izveštaja o istoriji izvršenih akcija kapije..................................15
   6. Scenario 6: Prikaz izveštaja o istoriji izvrpenih akcija sistema prskalica...............18
   7. Scenario 7: Dodavanje validne tablice kapiji........................................................19
   8. Scenario 8: Izmena režima kapije..........................................................................21
   9. Scenario 9: Uklanjanje validne tablice iz kapije....................................................23
   10. Scenario 10: Izmena vremena aktivacije prskalice................................................24
   11. Utisci o testiranju sa alatom Locust…………………………………………………………………..26
4. Rezultati testiranja sa simulatorima..............................................27
   1. Simulator: Lampa..................................................................................................27
   2. Simulator: Kapija...................................................................................................28
   3. Simulator: Prskalice...............................................................................................30
   4. Utisci o testiranju simulatora................................................................................31
5. Zaključak........................................................................................32
6. Uvod

Ovaj izveštaj dokumentuje proces testiranja softverskog rešenja za monitoring pametne kuće unutar pametnih gradova, 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 prate stanje svojih uređaja u realnom vremenu putem platforme u oblaku.

Softversko rešenje je dizajnirano za podršku pametnim uređajima unutar pametnih kuća. 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 registrovani korisnik: Može registrovati pametnu nekretninu i uređaje unutar nje, nadgledati njihov rad i upravljati njima.
* Administrator: Ima pravo nadzora i upravljanja sistemmom u realnom vremenu. Ne može izvršavati funkcionalnosti redovnog korisnika, niti obrnuto.
* Neautentifikovani korisnik: Može se registrovati i prijaviti na sistem.

Konkretno moje odgovornosti u okviru projekta su bile:

* Registracija pametne nekretnine
* Obrada zahteva za registraciju nekretnine
* Pregled dostupnosti pametnih uređaja
* Upravljanje SPA uređajem – Lampa
* Upravljanje SPA uređajem – Kapija za vozila
* Upravljanje SPA uređajem – Sistem prskalica

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.

1. Metodologija testiranja

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

1. Testni scenariji:

* Scenario 1: Registracija pametne nekretnine
* Scenario 2: Obrada zahteva za registraciju nekretnine
* Scenario 3: Prikaz izveštaja o dostupnosti pametnih uređaja
* Scenario 4: Prikaz izveštaja o istoriji vrednosti lampe
* Scenario 5: Prikaz izveštaja o istoriji izvršenih akcija kapije
* Scenario 6: Prikaz izveštaja o istoriji izvršenih akcija sistema prskalica
* Scenario 7: Dodavanje validne tablice kapiji
* Scenario 8: Izmena režima kapije
* Scenario 9: Uklanjanje validne tablice iz kapije
* Scenario 10: Izmena vremena aktivacije prskalice

1. Alati za testiranje:

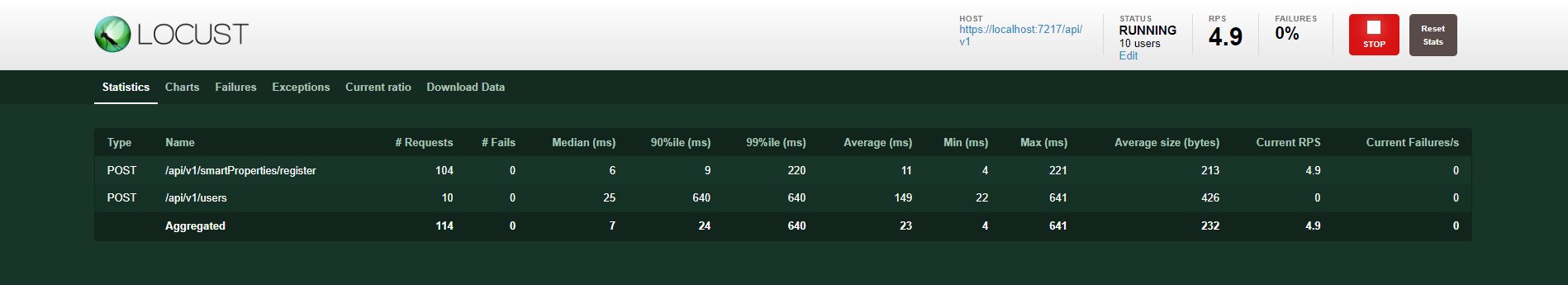
* Locust framework u python. Korišćen je Locust za simulaciju opterećenja sistema kroz različite testne scenarije. Python skripte su razvijene kako bi se definisali i izvršili testni scenariji koristeći Locust. Locust framwork je besplatan alat za testiranje performansi i opterećenja koji se koristi 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 pogledau performansi, bezbednosti i upravljanja opterećenjem aplikacije.

1. Rezultati testiranja

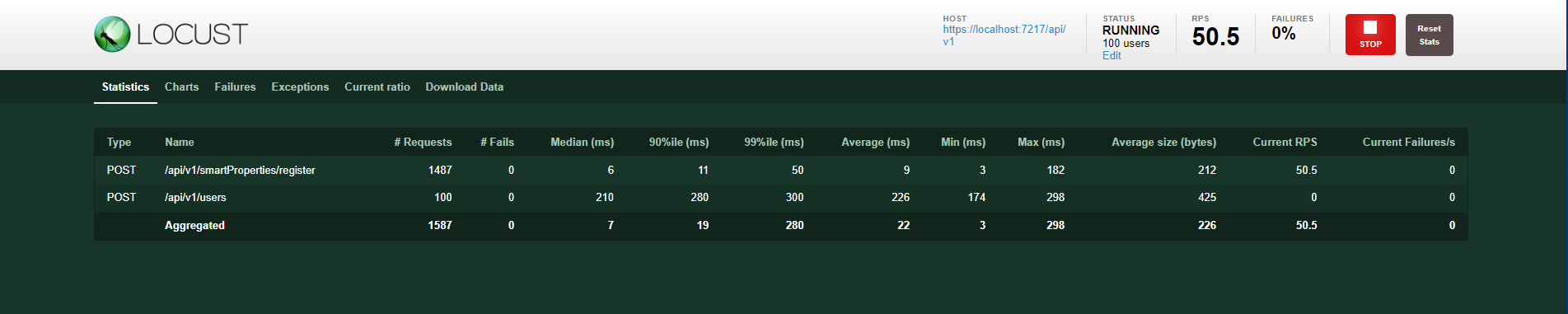
U ovom delu izveštaja analiziraćemo rezultate testiranja opterećenja Sistema korišćenjem alata Locust. Glavni cilj ovog testiranja bio je određivanje maksimalnog broja korisnika koji sistem može podneti bez gubitka performansi ili prekida rada. Kroz seriju testnih scenarija simulirali smo različite nivoe 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.

3.1 Scenario 1: Registracija pametne nekretnine

Scenario 1 testira slučaj korišćenja u kojem se korisnik login-uje kao User i zatim registruje pametnu nekretninu sa nasumičnim podacima. Ovaj scenario sam testirao sa različitim opterećenjima. Testiran je sa 10, 100, 1000, 2000 i 10000 korisnika.

Test1: 10 Users, 3 Spawn rate

Vreme izvršavanja u testu 1 je izuzetno malo, takođe nema grešaka. To se može očekivati zato što je broj korisnika takođe veoma mali.

Test2: 100 Users, 10 spawn rate

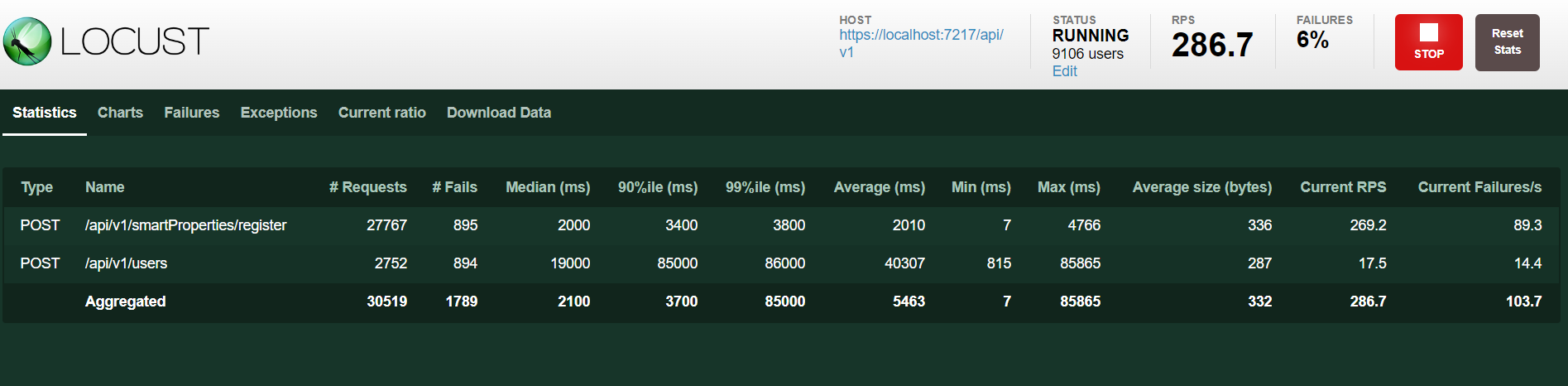
Vreme izvršavanja u testu 2 je takođe veoma malo. Zato što je broj Usera idalje mali. Nema grešaka. Ipak počinje da se vidi razlika u vremenu potrebnom za login u odnosu na prethodni test.

Test 3: 1000 Users, 100 Spawn rate

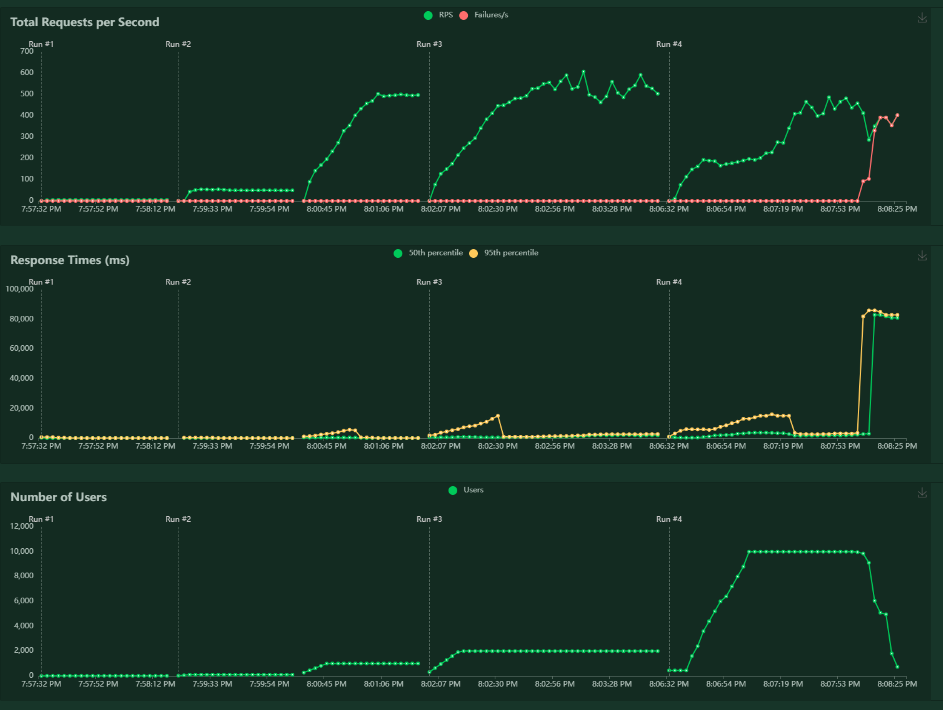
U testu 3 se vide velike razlike u odnosu na prethodna 2 testa. Iako idalje nema grešaka sistem je znatno sporiji. Vidimo da je prosečno vreme preko 10 puta veće nego u prethodnim testovima.

Test 4: 2000 Users, 160 Spawn rate

U testu 4, iako nema grešaka sistem postaje previše sport. Prosečno vreme je preko 5 puta veće od prethodnog testa.

Test 5: 10000 Users, 500 Spawn rate

U testu 5, možemo da vidimo pojavu greške. Prosečno vreme je bilo previše dugačko, spawn rate je bio izuzetno velik i sistem nije mogao da izdrži toliko opterećenje.

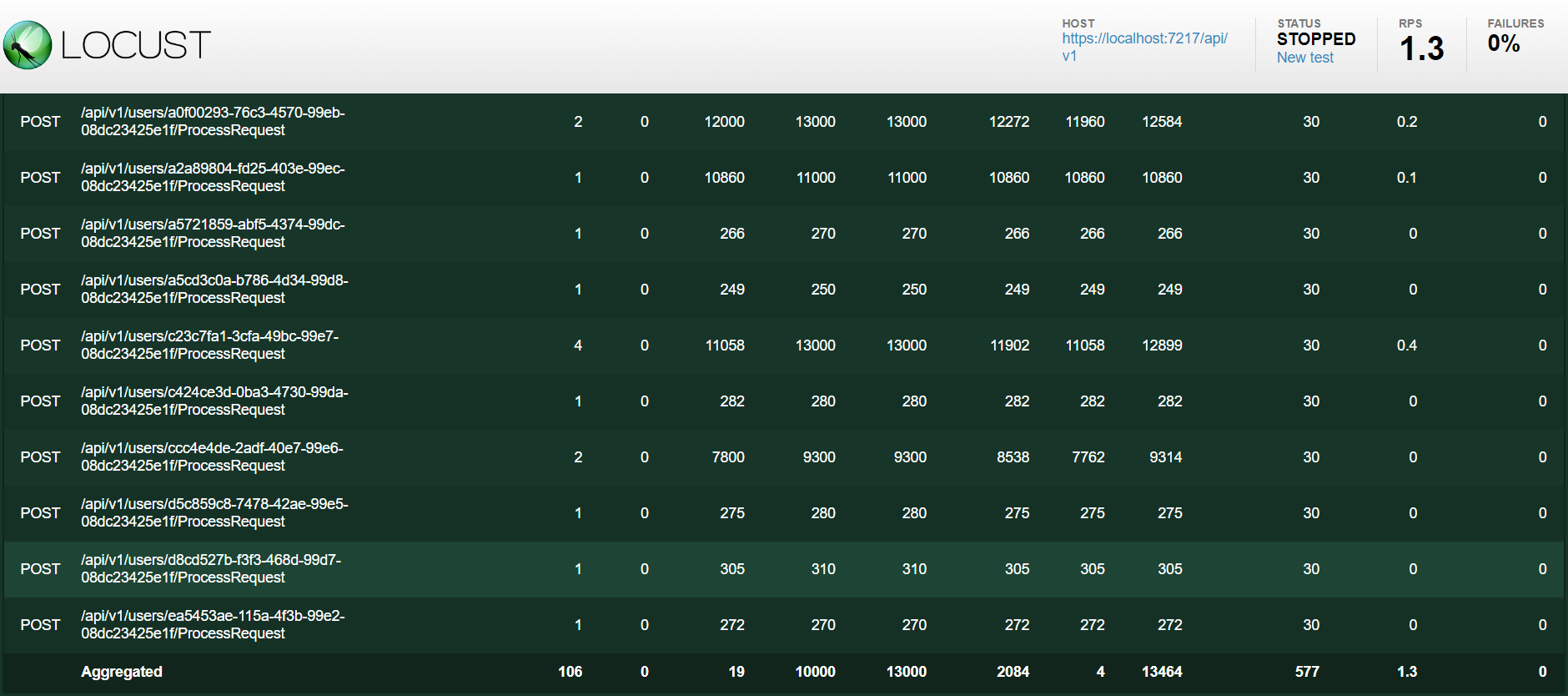
Grafikon

Na datom grafiku možemo da vidimo razlike među testovima. Vidi se da nema velike razlike u količini request-ova po sekundi u testovima sa 1000 i 2000 korisnika. Ipak Response time je znatno veći u testu sa 2000 korisnika. Dok u testu sa 10000 postaje prevelik za korišćenje. Smatram da su performanse scenarija 1 zadovoljavajuće. Smatram da bi jači računar mnogo bolje podneo testove i da je unapređenje računara najbolji način da se ovaj scenario poboljša.

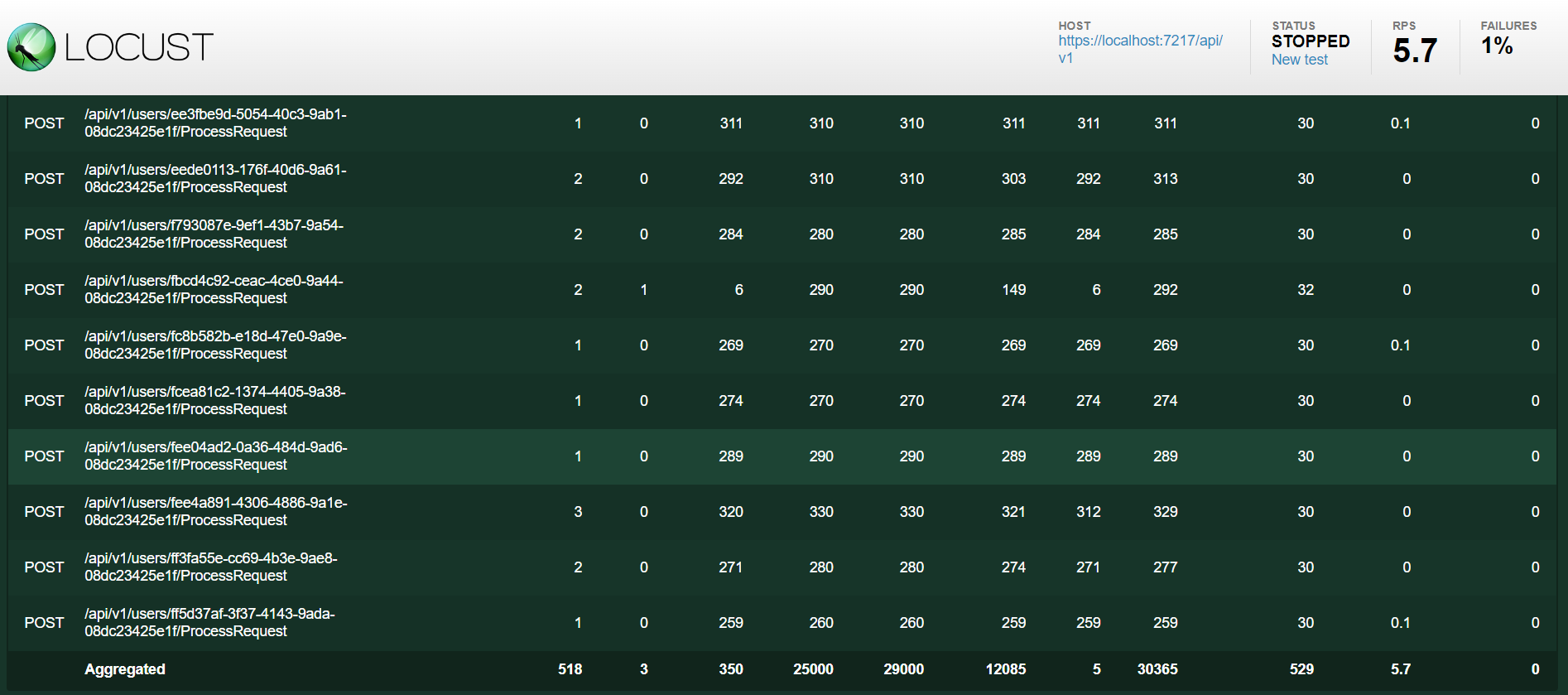
3.2 Scenario 2: Obrada zahteva za registraciju nekretnine

Scenario 2 testira slučaj korišćenja u kojem se korisnik login-uje kao admin, dobavlja sve pametne nekretnine koje imaju status Pending i zatim odobrava ili odbija zahtev. Ovaj scenario sam testirao sa različitim opterećenjima. Testiran je sa 10, 100, 300 i 500 korisnika.Test nakon scenarija 3.1

Kada sam prvi put pokrenuo scenario 3.2 nakon scenarija 3.1 bila je potrebna ogromna količina vremena kako bi se test pokrenuo. Razlog tome je taj što scenario 3.1 napravi preveliku količinu zahteva za pravljenje nekretnine. Uspeo sam značajno da smanjim vreme potrebno za izvršavanje scenarija 3.2 nakon scenarija 3.1 dodavanjem paginacije. Potrebno vreme je idalje veliko kao što se vidi na slici iznad, ali u realnom sistemu nikada ne bi došlo do slučaja da postoji preko 100000 nekretnina na čekanju. Paginacija je znatno smanjila potrebno vreme za izvršavanje testa, ali ne dovoljno. Naredni testovi biće izvršeni sa znatno manjim brojem nekretnina na čekanju.

Test1: 10 Users, 1 Spawn rate

Ovo je prvi test sa manjim brojem nekretnina, Vidi se da je ovaj scenario znatno sporiji od prethodnog. Na 10 usera idalje nema grešaka, jedini problem je brzina programa.

Test 2: 100 Users, 10 Spawn rate

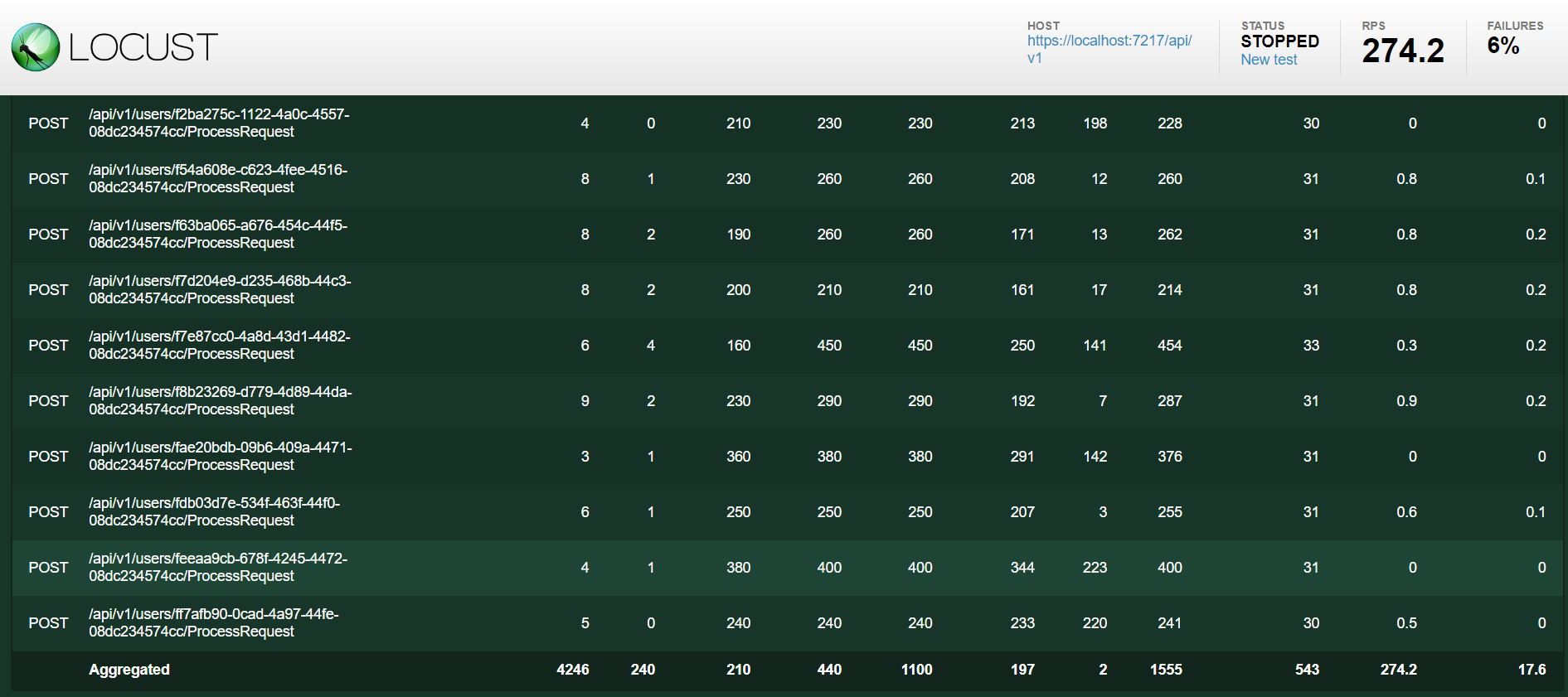
Test sa 100 usera je znatno sporiji nego sa 10, ali verujem da je do toga došlo zbog mnogo veće količine nekretnina nego u prethodnom testu. Takođe vidimo 3 failure-a. Ipak to nije problem zato što su to samo Bad requestovi 400. Sistem ne puca zbog toga, već su samo procesirane nekretnine pokušane da budu procesirane ponovo. To se može videti na slici ispod.

Test 2: Failures



Test 3: 300 Users, 50 Spawn rate, Error

U testu 3 je na 300 usera došlo do errora koji je prekinuo rad sistema. Problem je bio prevelika količina memorije koju sistem nije mogao da izdrži.

Test 4: Users 300, Spawn rate 50

Kao što se vidi na slici iznad pokrenut je test 4 koji ima identičan broj user-a i spawn rate kao i test 3. Ipak vreme je mnogo manje, errori su bili samo bad request. Nije došlo do errora koji je prekinuo rad sistema. Razlog tome je znatno manji broj nekretnina koje su bile na čekanju u testu 4, nego u testu 3.

Test 5: Users 500, spawn rate 50

Količina nekretnina na čekanju je morala da bude veća za ovaj test s obzirom da imamo 500 user-a koji te nekretnine procesiraju. Čak i kada sam smanjio količinu nekretnina idalje je dolazilo do errora koji je prouzrokovao rušenje sistema. Na osnovu svih ovih testova možemo da zaključimo da same funkcije za procesiranje nisu problem, kratko traju i ne troše puno memorije. Najveći problem ovog scenarije predstavlja dobavljanje svih nekretnina koje su u statusu čekanja. Performanse su obrnuto proporcionalne količini nekretnina koje dobavljamo. Iako performanse nisu ni blizu zadovoljavajuće kao u prethodnom scenariju, smatram da je to u redu. Razlog je to što je kritična operacija dobavljanje nekretnina koju vrši admin. U realnom sistemu Nećemo imati 500 admina koji pokušavaju u isto vreme da dobavljaju sve nekretnine isto tako nećemo imati preko 3000 nekretnina na čekanju tako da ne bi trebalo da dođe do gore navedenih problema.

3.3 Scenario 3: Prikaz izveštaja o dostupnosti pametnih uređaja

Scenario 3 testira slučaj korišćenja u kojem se korisnik login-uje kao user, dobavlja svoje pametne nekretnine zatim dobavlja svoje pametne uređaje i uzima id random uređaja. Zatim dobavlja izveštaj o dostupnosti uređaja u poslednjih 6, 12, 24 sata kao i poslednjih nedelju i mesec dana. Takođe dobavlja izveštaj za random vremenski period u poslednjih mesec dana.

Test 1: Users 10, Spawn rate 1

Možemo da vidimo da se test1 izvršava izuzetno brzo i bez problema. To je i očekivano sa malim brojem user-a.

Test 2: Users 100, Spawn rate 10

Razlika između testa 1 i testa 2 je neprimenta. Idalje dobre performanse i bez grešaka.



Test 3: 1000 Users, 100 Spawn rate

Test 3 je značajno sporiji nego prethodna 2. Takođe memorija koja se zauzima je značajno veća. Ipak idalje nemamo problem sa greškama. Test iako sporiji nije previše spor za rad na sistemu.



Test 4: 2000 Users, 180 Spawn rate

Test 4 je prvi u kojem sistem puca. Dolazi do greške jer je 2000 user-a koji traže izveštaj previše za sistem da izdrži. Memorija koja se zauzima je takođe veća kao i vreme potrebno za izvršavanje. Moja mašina nije mogla da izdrži toliku količinu memorije.

Scenario 3 je imao zadovoljavajuće performanse. Nije bilo razlike na 10 i 100 Usera. Ipak na 1000 se videla velika razlika u odnosu na 100, kako u vremenu tako i u količini memorije. Ipak sa grafika se vidi da vreme potrebno za dobavljanje izveštaja ostaje isto i u testu 3 i u testu 4. Dakle login je funkcija kojoj vreme potrebno da se izvrši značajno raste sa porastom user-a.

3.4 Scenario 4: Prikaz izveštaja o istoriji vrednosti lampe

Scenario 4 testira slučaj korišćenja u kojem se korisnik login-uje kao user, dobavlja svoje pametne nekretnine zatim dobavlja svoje pametne uređaje i proverava da li je uređaj lampa. Uzima id random lampe. Zatim dobavlja izveštaj o istoriji vrednosti lampe u poslednjih 6, 12, 24 sata kao i poslednjih nedelju i mesec dana. Takođe dobavlja izveštaj za random vremenski period u poslednjih mesec dana.

Test 1: Users 10, Spawn rate 1

Test 1 je prošao odlično. Sa brojem od samo 10 usera dobavljanje izveštaja ne predstavlja problem.

Test 2: Users 100, Spawn rate 10

Test 2 je prošao dobro kao i test 1. Ne vide se značajne razlike u performansama. Nije bilo grešaka.

Test 3: Users 1000, Spawn rate 100

Test 3 iako je počeo dobro kada se ostavi da traje malo duže dolazi do pucanja što će se videti i na narednoj slici. Performanse su znatno lošije od prethodna 2 testa u svakom pogledu.

Test 4: Users 1000, Spawn rate 100 Error

Test 4 je pokrenut pod istim uslovima kao i test 3, ipak on je pušten da radi duže. Nakon kratkog vremena došlo je do pucanja sistema. Istraživanjem sam video da do greške dolazi kada imamo previše konekcija ka jednom endPointu. Pokušao sam da izmenim maximalnu količinu koja je dozvoljena ali to je dovodilo do drugačijih problema u sistemu. Te sam ostavio ovakvu konfiguraciju. Performanse scenaria nisu dobre kao od nekih prethodnih ali takođe nisu ni toliko loše.

Grafikon

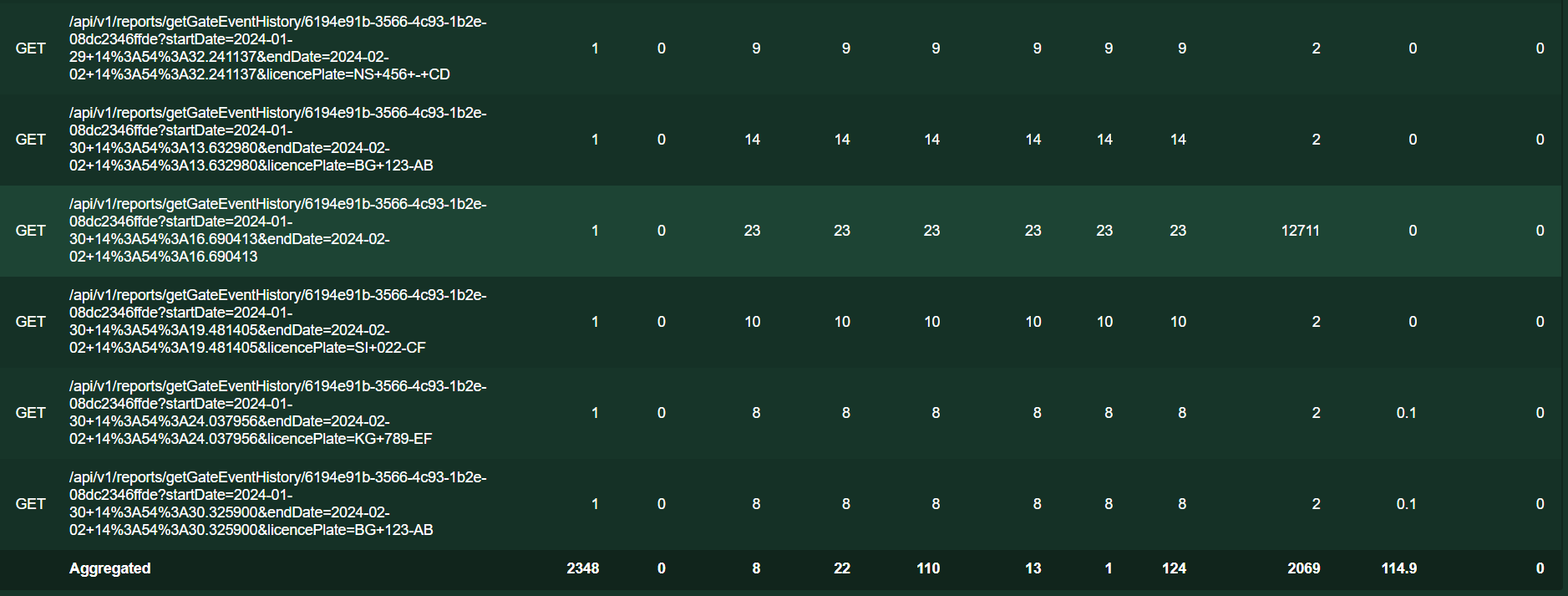
Vidimo da kao i do sada sama pretraga istorije nije toliko spora operacija. Login usporava aplikaciju najviše od svega. Na grafiku vidimo koliki pad ima response time onda kada se svi Useri loginuju.

3.5 Scenario 5: Prikaz izveštaja o istoriji izvršenih akcija kapije

Scenario 5 testira slučaj korišćenja u kojem se korisnik login-uje kao user, dobavlja svoje pametne nekretnine zatim dobavlja svoje pametne uređaje i proverava da li je uređaj kapija. Uzima id random kapije. Zatim dobavlja izveštaj o istoriji vrednosti lampe u poslednjih 6, 12, 24 sata kao i poslednjih nedelju i mesec dana. Takođe dobavlja izveštaj za random vremenski period u poslednjih mesec dana kao i izveštaj za random tablicu koja se nalazi u kapiji.

Test 1: Users 10, Spawn rate 1

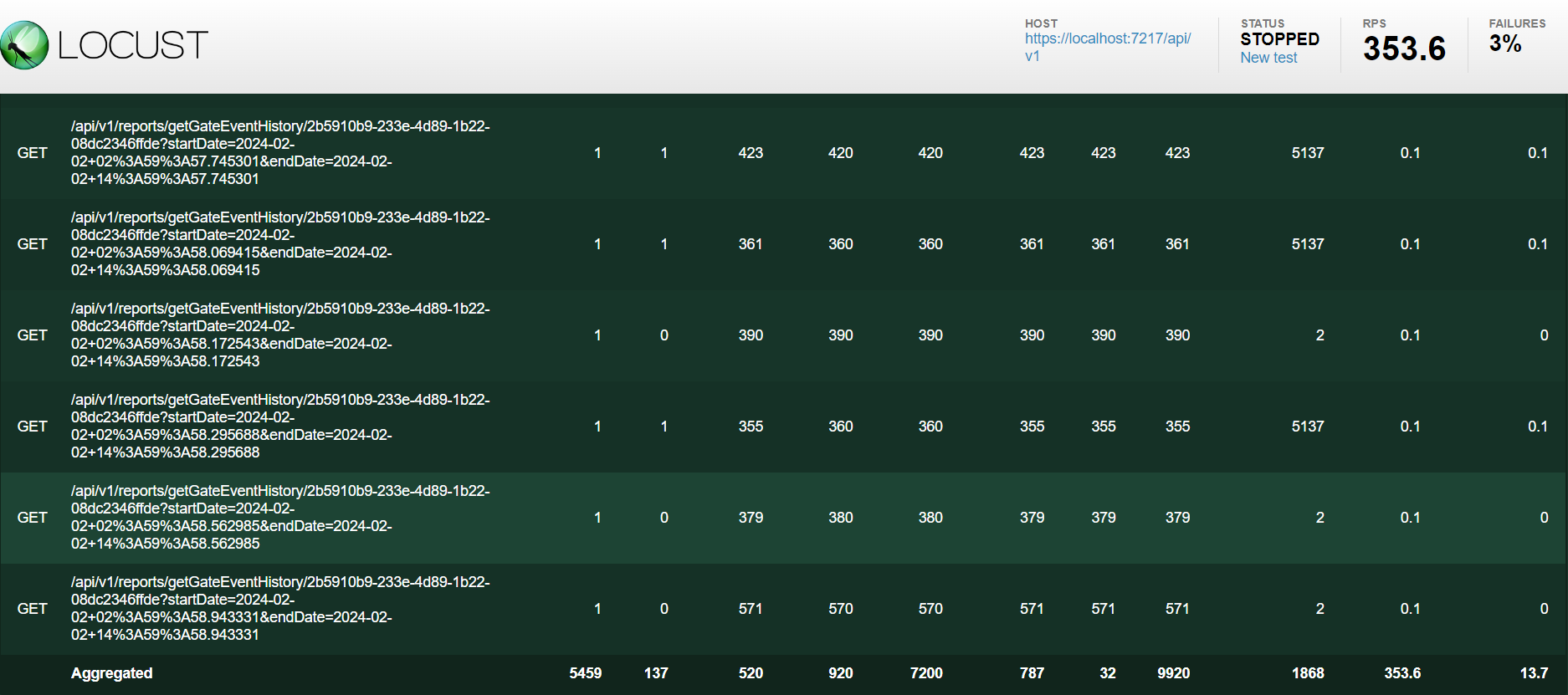
Prvi test kao i do sad prolazi bez greške.

Test 2: Users 100, Spawn rate 10

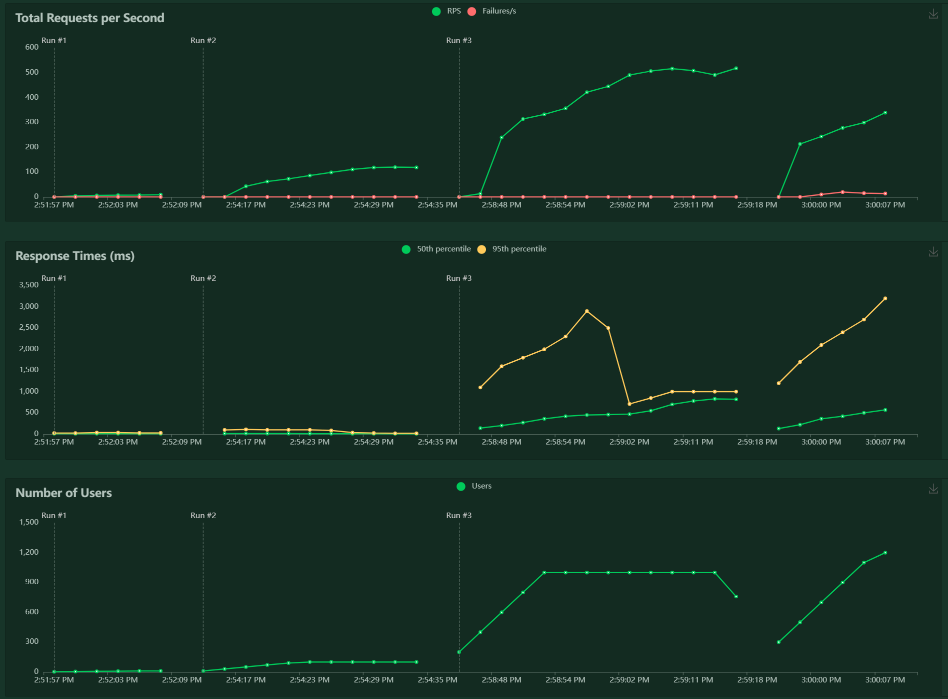
Drugi test sa poprilično sličnim performansama kao i 1. prolazi bez greške.

Test 3: Users 1000, Spawn rate 100

Test 3 je znatno sporiji i zauzima mnogo više memorije nego protehodna 2 testa. Nije bio dugo pokrenut ali nije dolazilo do greške.

Test 4: Users 1200, Spawn rate 100

Test 4 je pukao odmah pri početku. Isti eror kao i u prethodnom scenariju. Naš backend deo aplikacije puca, verujem da to nema veze sa količinom User-a. Sva 4 testa su trčala na istoj instanci backend aplikacije, tako da verujem da smo prekoračili limit istih endpointa koji su pozvani te da je zato došlo do pucanja aplikacije.

Grafikon

Na grafikonu vidimo da je login ponovo operacija kojoj je potrebno najviše vremena da se izvrši. Ipak pad nije toliko velik kao u prethodnom slučaju. Razlog tome je taj što neke od kapija imaju veoma veliku količinu podataka koju vraćaju. U testovima se moglo primetiti da neke kapije vraćaju po 1000 puta više memorije nego neke druge. Razlog tome je što neke kapije imaju veliku količinu podataka koju dobavljamo preko naših testova. Tako da je količina memorije zavisna od količine podataka koju dobavljamo iz kapije.

3.6 Scenario 6: Prikaz izveštaja o istoriji izvršenih akcija sistema prskalica

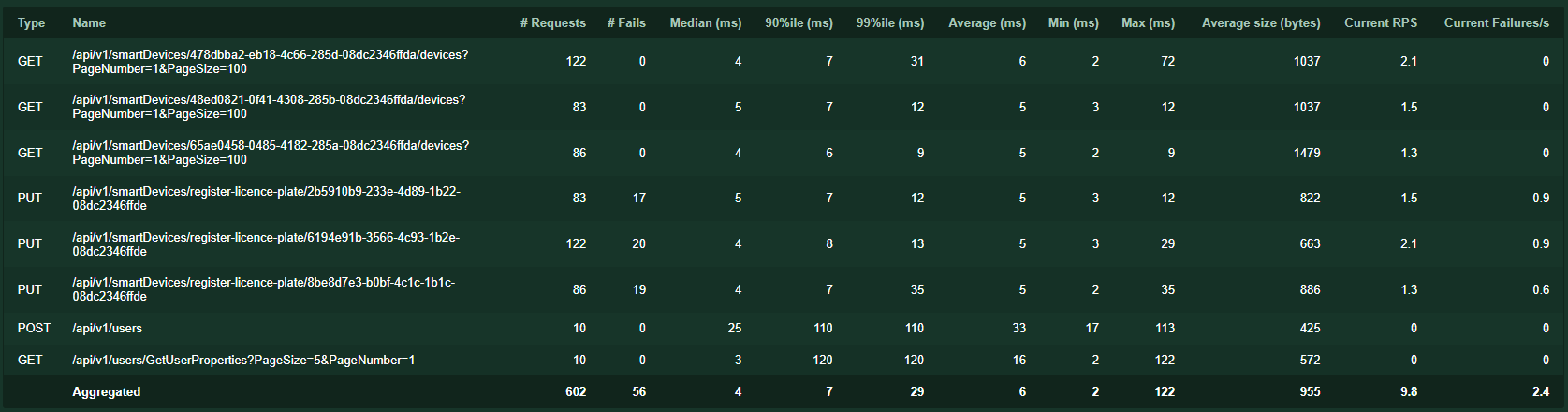
Scenario 6 je radio isto kao i prethodni scenario samo ovaj put sa sistemmom prskalica. Smatram da nema potrebe za dodatnim grafikonima s obrzirom da su performanse iste kao kod kapije. Jedina razlika je ta što je količina podataka u prskalicama manja, te je memorija manja takođe. Ista greška izlazi kada se isti endpoint pozove dovoljan broj puta. Do tog broja poziva sistem ne puca čak ni na 1500 User-a. U narednim testovima performanse sistema će biti značajno lošije, što će se i videti na slikama.

3.7 Scenario 7: Dodavanje validne tablice kapiji

Scenario 7 je prvi scenario u kojem testiramo i mqtt. U ovom scenariju korisnik se loginuje, dobavlja id kapije i unosi novu tablicu koju želi da prosledi kapiji. Taj endpoint šalje poruku na topik simulatoru.

Test 1: Users 5, Spawn rate 1

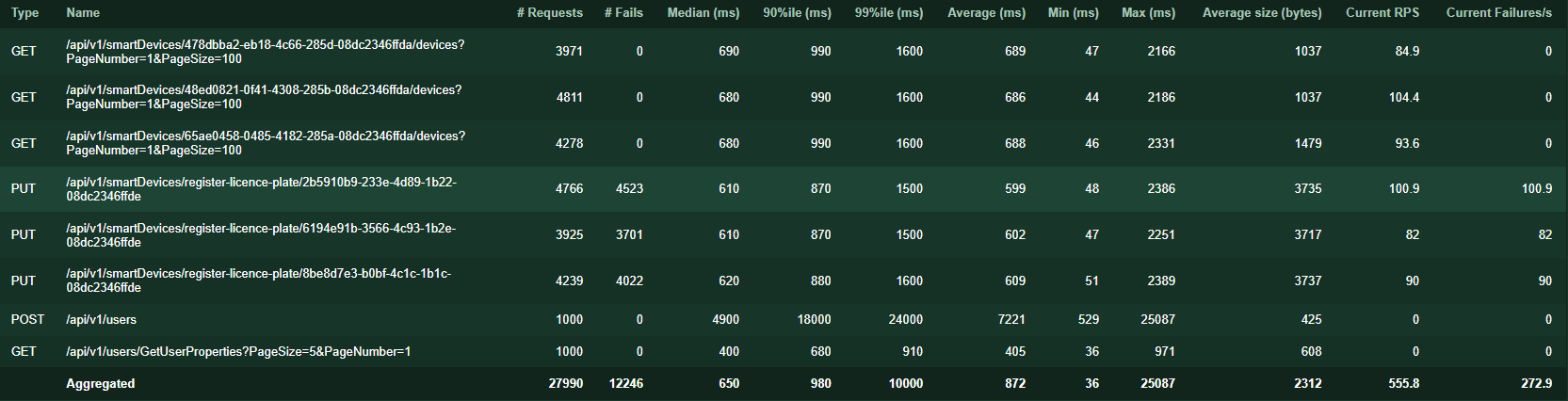
Test 1 je prošao dobro. Nije došlo do pucanja sistema. Performanse su takođe dobre.

Test 2: Users 10, Spawn rate 1

Test 2 je ipak počeo da izbacuje grešku. Greška se periodično desi i kaže da klijent nije konektovan. Defintivno nije dobro da dobijamo tu grešku na samo 10 User-a, ali sistem idalje radi čak i kada dobijamo ovu grešku tako da ona nije fatalna po naš sistem.

Test 3: Users 160, Spawn rate 30

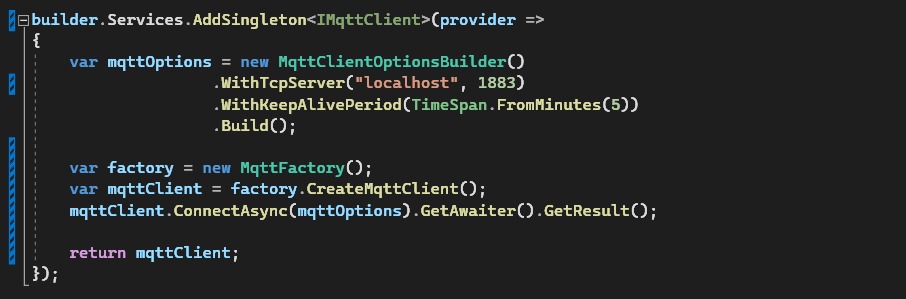
Test 3 se ponaša isto kao i prethodni, greške su učestalije ali nijedna od njih nije srušila ceo sistem i onemogućila rad nad njim. Sve greške koje smo dobili su bile da klijent nije konektovan.

Test 4 Users 1000, Spawn rate 150

Test 4 je neuspešan. Iako nijedna greška nije u potpunosti srušila sistem možemo videti sa desne strane da kada je narastao broj user-a da je svaki pokušaj dodavanja tablice bio neuspešan. U prethodnim testovima iako je dolazilo do greške sistem je u većini slučajeva uspevao da doda tablicu što u poslednjem testu nije slučaj.

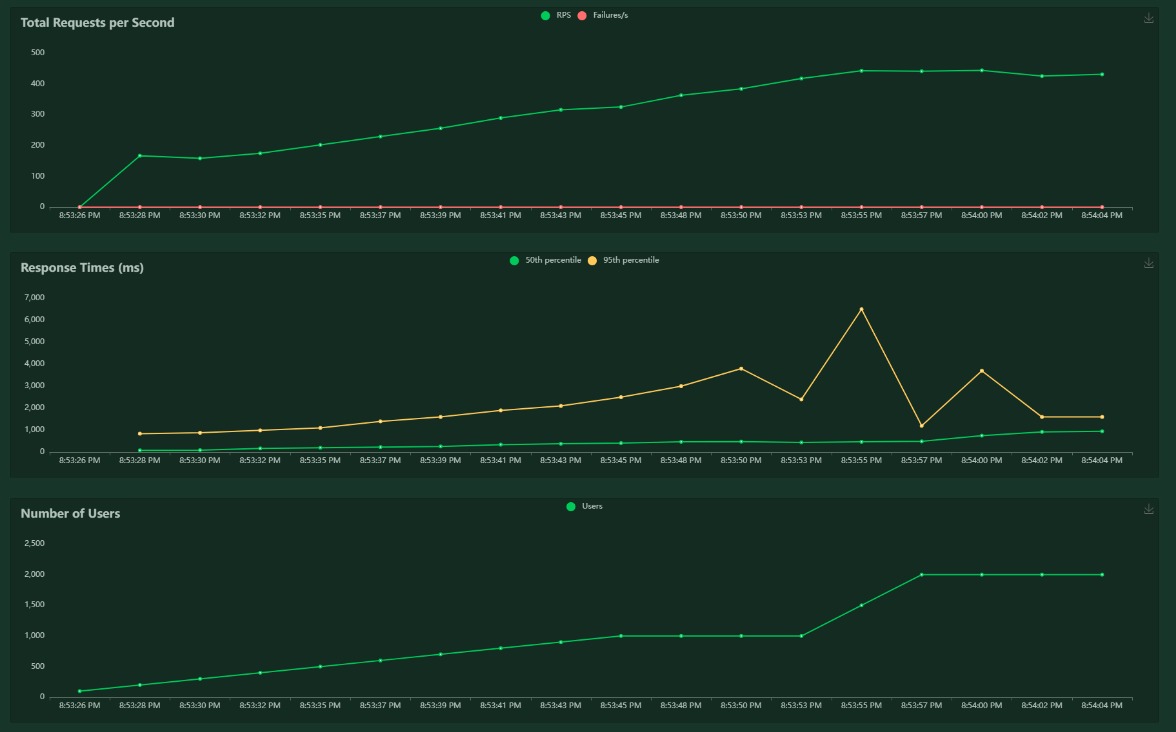
Kod mqtt testa sa kapijom imamo mnogo više grešaka nego u prethodnim testovima. Ipak nijedna od tih grešaka nije potpuno srušila sistem te ćemo ih tolerisati.

Kasno uveče, dan pred rok za izradu, kolega je pronašao grešku zbog koje smo dobijali mqtt error. Greška je uklonjena tako što je stavljeno da se mqtt konektuje samo na početku aplikacije, kao singleton. Ispod možete da vidite ispravljen kod, kao i performanse sistema.

Linija koja je ispravila grešku

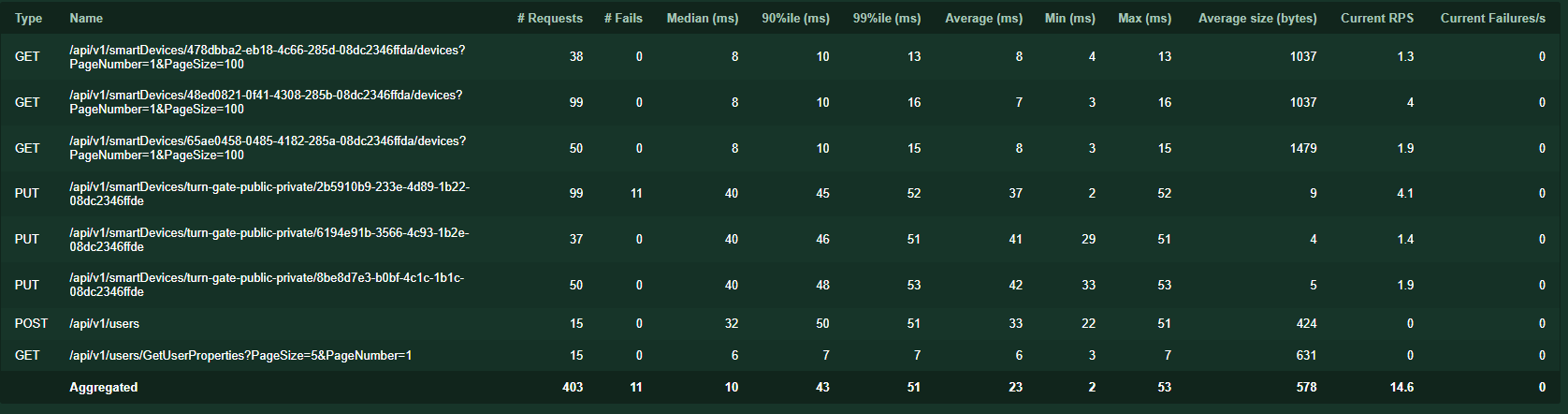
Users 1500, Spawn rate 200

Kao što se vidi ispravka greške je značajno poboljšala performanse sistema i više ne dolazi ni do jednog errora.

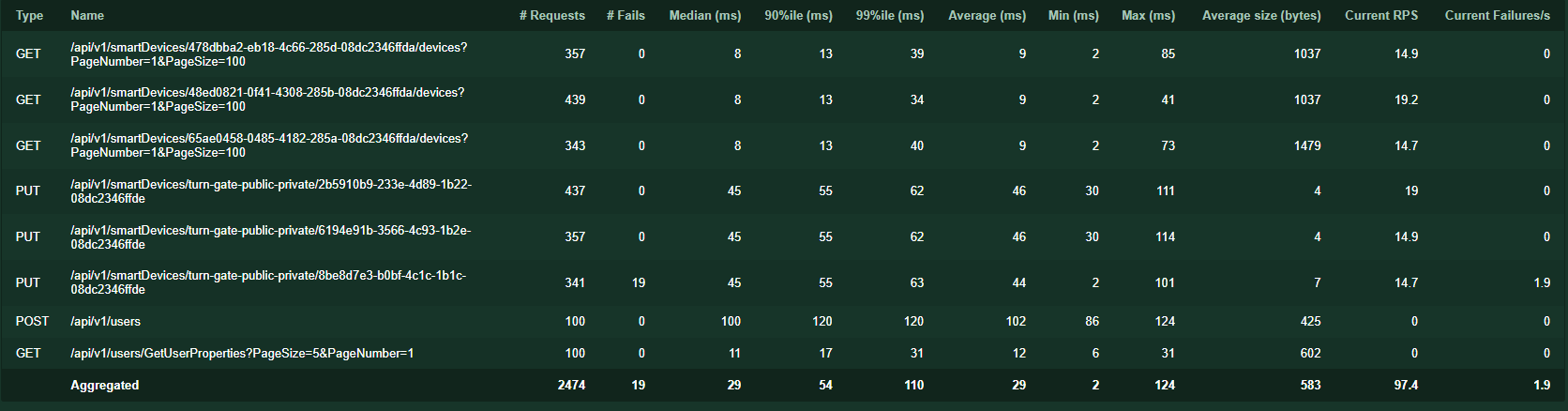
Grafik ispravljenog scenarija

3.8 Scenario 8: Izmena režima kapije

Scenario 8 se odvija tako što se korisnik uloguje kao User, nađe svoje propertije, zatim kapiju sa random id-jem (Isto kao i ranije kada smo imali scenario sa kapijom). Korisnik kada nađe svoju kapiju šalje zahteve da se specijalni režim kapije uključi ili isključi. To je isti endpoint sa True ili False vrednošću.

Test 1: Users 15, Spawn rate 2

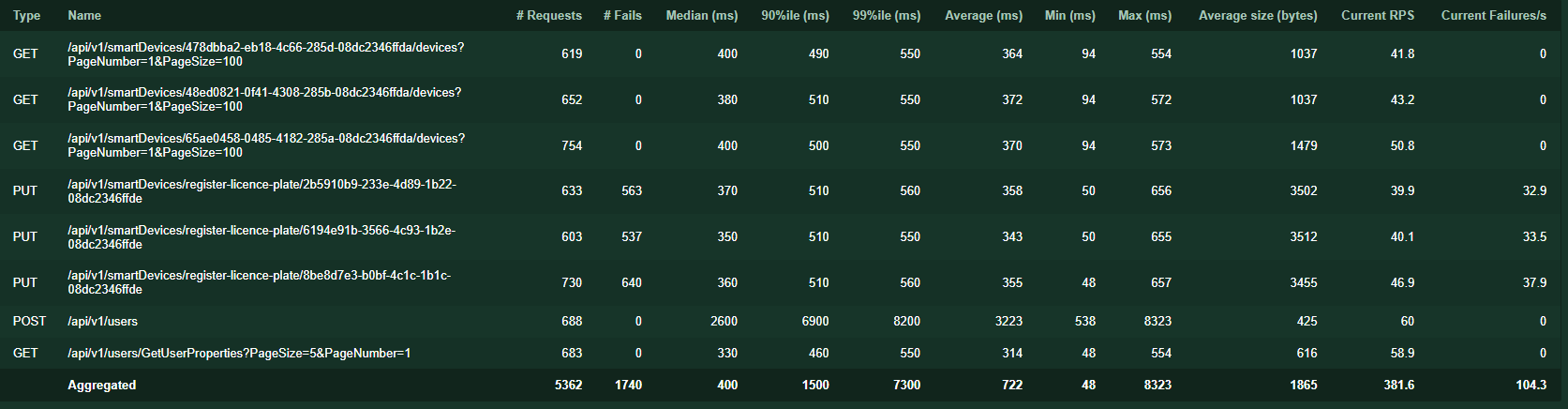
Test 1 je prošao dobro. Greške su sve tipa 404. Te greške ne ruše sistem. Kapija nije bila online pa test nije mogao da joj izmeni specijalni režim. Test se brzo obavljao.

Test 2: Users 100, Spawn rate 10

Test 2 je prošao dobro kao i test 1. Jedine greške su bile 404 koje nisu problem. Vidi se da je sistem malo sporiji kod logina, ali to smo primetili u svakom scenariju do sada.

Test 3: Users 1000, Spawn rate 100

Test 3 je prošao odlično. Bez grešaka sa 1000 user-a. Problem je bio vreme. Login na 1000 user-a je znatno sporiji, ali i ostale funkcionalnosti su se značajno usporile u poređenju na prethodne testove. Ipak mqtt funkcionalnost sa 1000 user-a bez grešaka je zadovoljavajuća.

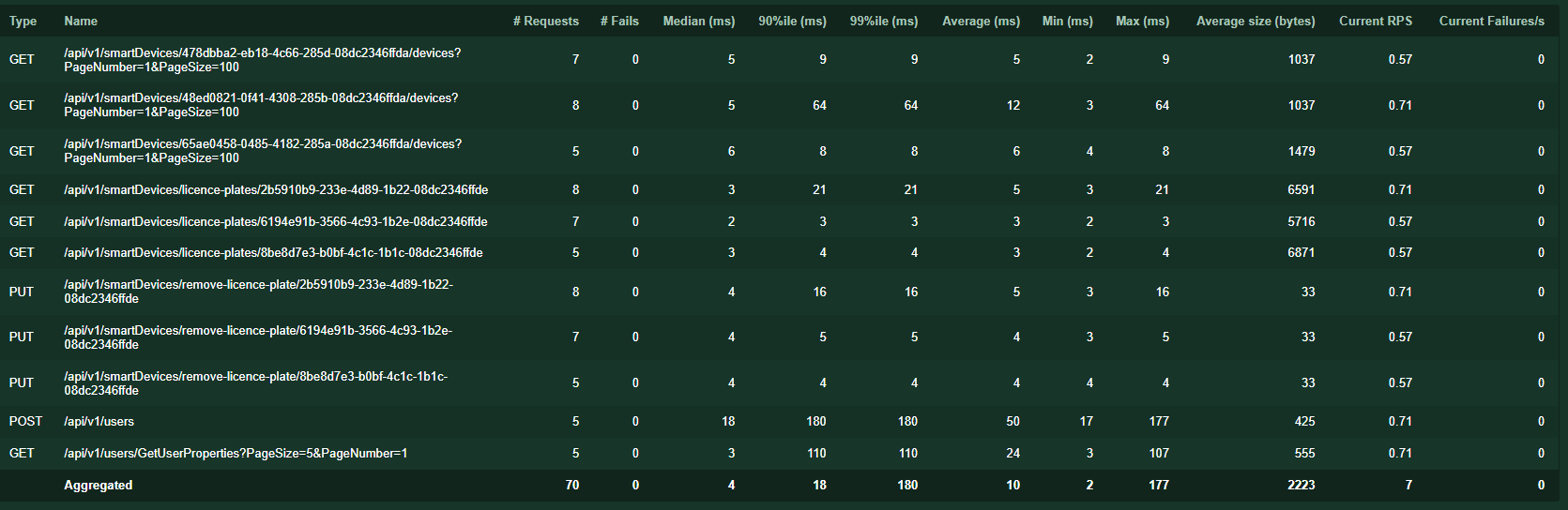
Test 4: Users 1500, Spawn rate 150

Test 4 je prvi test sa greškama. Odmah nakon pokretanja su se desile greške kao u prethodnom scenariju. Klijent nije povezan. Čak nije ni stiglo da se loguje preko 700 usera. Tako da ovaj test nije zadovoljavajuć. Ispravljanjem koje je gore pomenuto ovaj test takođe radi bez grešaka.

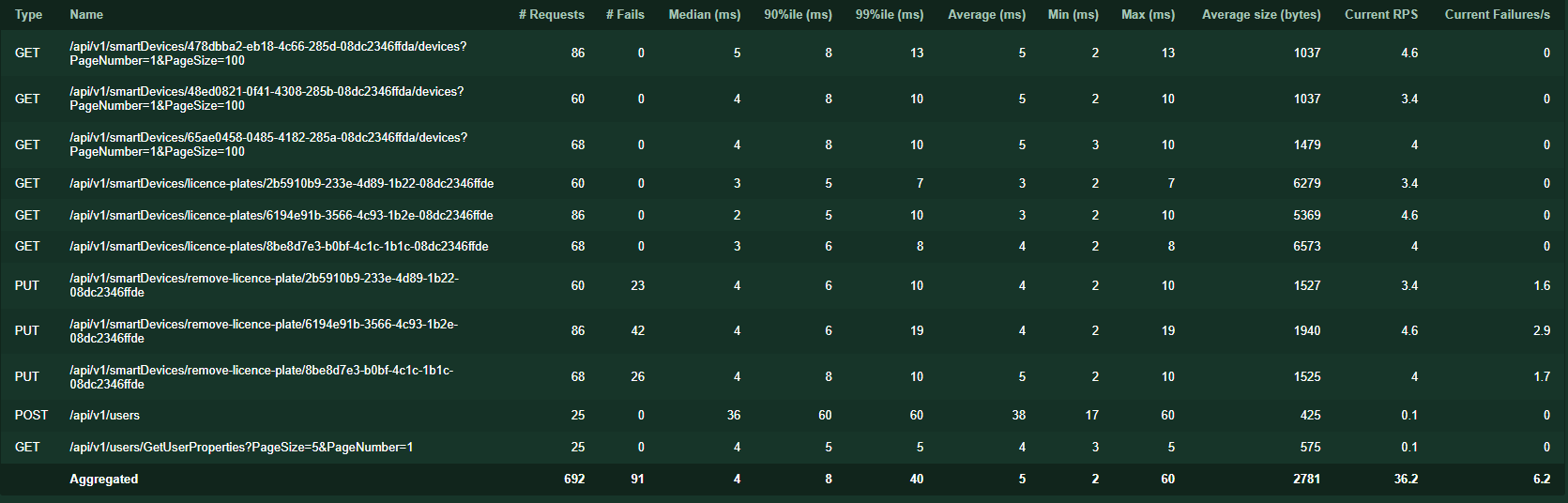
Ovo je najjednostavnija funkcionalnost koja koristi mqtt koju imam tako da je ovakav rezultat očekivan, znatno je bolji nego u prethodnom scenariju.

3.9 Scenario 9: Uklanjanje validne tablice iz kapije

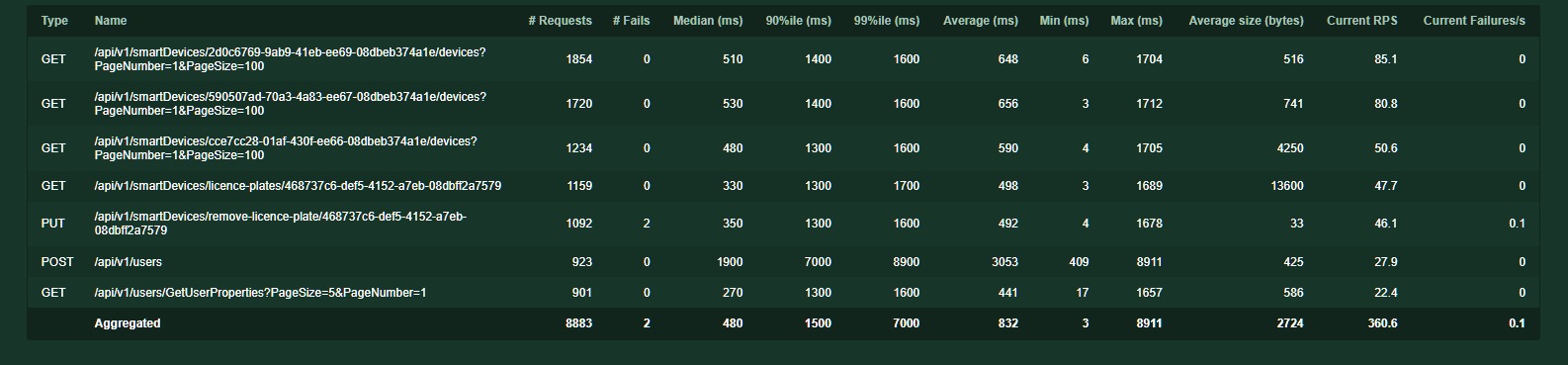
Scenario 9 radi kao i scenario u kojem smo dodavali tablice kapiji. Samo ovaj put uklanjamo tablice iz liste validnih tablica.

Test 1: Users 5, Spawn rate 1

Test 1 je prošao dobro, bez grešaka sa dobrim performansama.

Test 2: Users 25, Spawn rate 5

Test 2 je počeo da izaziva greške. Greške su identične kao u Scenariju 7 (Dodavanje tablica). Greške ne ruše kompletam program. S obzirom da je ponašanje scenarija isto kao i Scenario 7 smatram da nema potrebe za dodavanjem ostatka grafika. Greške su češće sa više User-a. i svaka greška je tipa klijent nije konektovan.

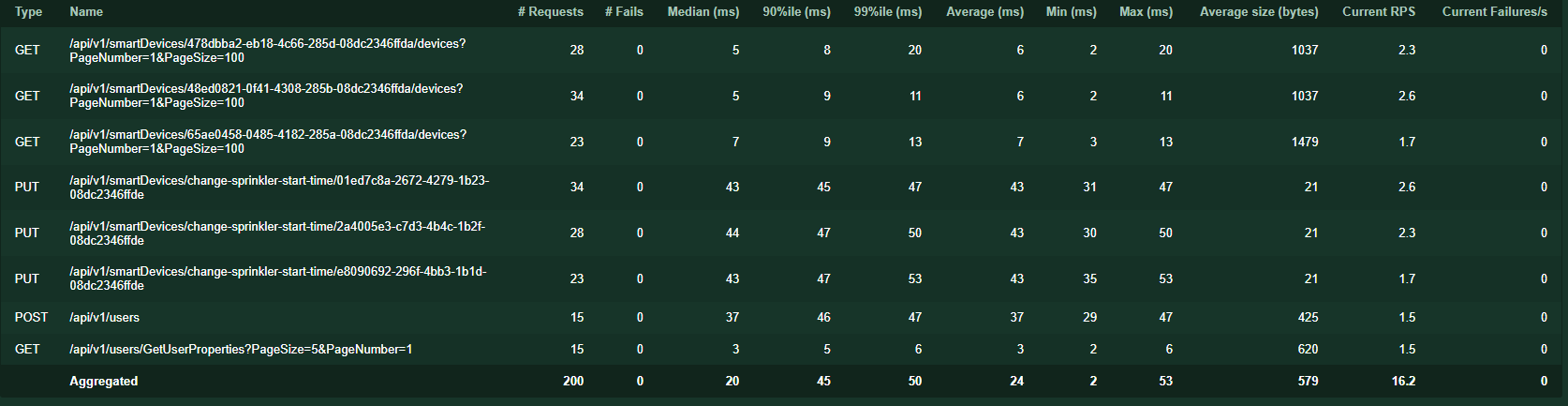
Users 1500, Spawn rate 200

Nakon ispravke navedene u scenariju 7 jedina greška koja se dogodila je bila 400, da nema tablice koju pokušavamo da uklonimo.

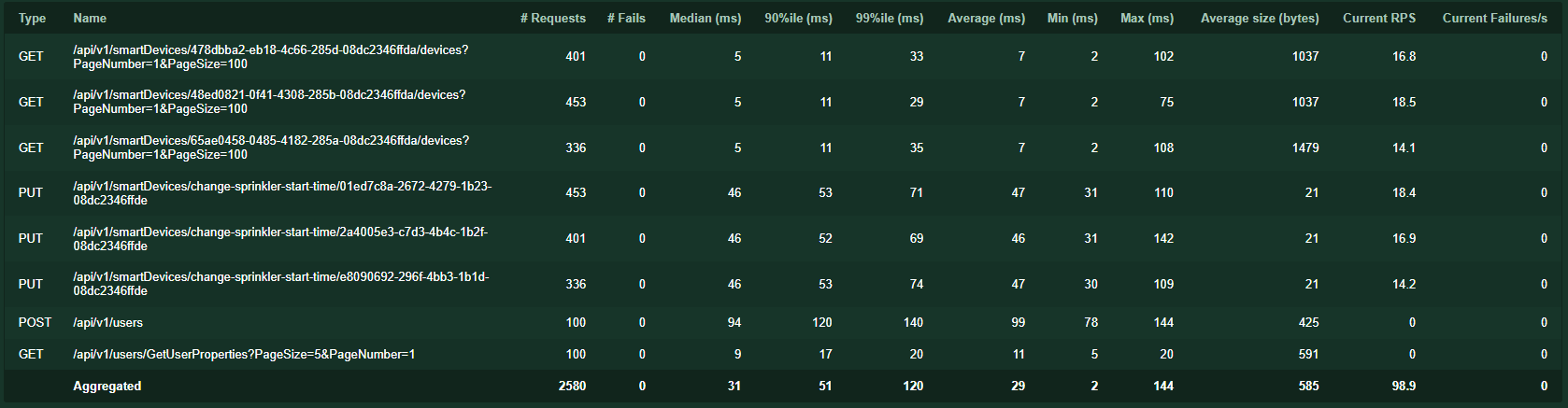
Grafik ispravljenog scenaria 9

3.10 Scenario 10: Izmena vremena aktivacije prskalice

Scenario 10 je poslednji scenario koji ću testirati. Početak scenarija je kao i do sada, login, dobavljanje nekretnina, pa prskalice. Nakon toga korisnik unosi random vreme za početak aktivacije prskalice. Ovo je takođe scenario u kojem imamo mqtt.

Test 1: Users 15, Spawn rate 3

Test 1 je prošao dobro, performanse su dobre i nema grešaka.

Test 2: Users 100, Spawn rate 10

Test 2 je takođe dobro prošao. Razlika u performansama nije velika osim u login-u. Nije bilo grešaka i sistem je bio relativno brz.

Test 3: Users 1000, Spawn rate 100

Test 3 je počeo da izaziva greške. Performanse su drastično lošije nego u testovima 1 i 2. Greške nisu fatalne. Greške su tipa klijent nije konektovan kao i u svakom scenariju koji je koristio mqtt do sada. Greške nisu fatalne, ali kako raste broj korisnika sve su učestalije. Ipak mnogo bolje performanse od Scenarija u kojima se dodaju ili izbacuju tablice iz kapije.

3.11 Utisci o testiranju sa alatom Locust

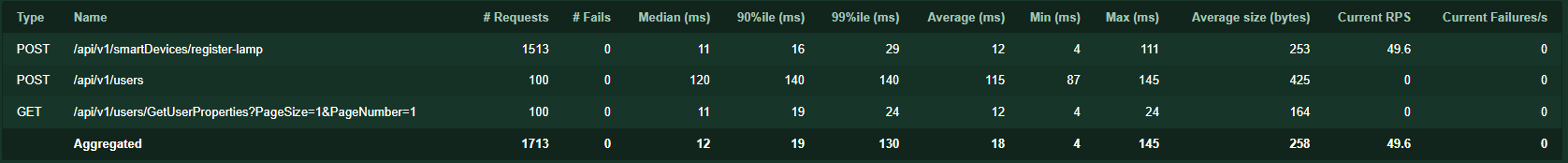
Testiranje alatom locust je bilo poprilično jednostavno. Nije bilo potrebno previše vremena da se otkucaju python skripte za testove. Rezultati su zadovoljavajući. Login je operacija koja je kočila svaki od testova. Sa povećanjem broja korisnika login je bio znatno sporiji. Funkcionalnosti koje su koristile mqtt su bile sklone greškama. U većini slučajeva te greške nisu bile fatalne po sistem. Sistem radi dobro do oko 1000 korisnika. Preko 1000 korisnika počinju da se javljaju razne vrste grešaka. Treba imati na umu da je ovo testiranje sprovođeno na 1 mašini koja je imala pokrenut backend, docker, simulacionu skriptu kao i sam locust test. Pojačanje mašine bi zasigurno pojačalo performanse na testovima. Koristio sam downsampling i paginaciju kako bih poboljšao performanse testova. Kolega je dodao index na login kako bi ubrzao operaciju loginovanja. Koristio sam skripte za punjenje influx baze kako bismo imali uvid u performanse izveštaja kada je količina podataka velika. Smatram da je testiranje dobro prošlo iako naravno ima mesta za napredak kao i uvek.

1. Rezultati testiranja sa simulatorima

U ovom poglavlju cilj mi je da pokažem ponašanje sistema kada je pokrenut različit broj simulatora određenog uređaja. Prikazivaću rezultate testova za uređaje: Lampa, Kapija i prskalice.

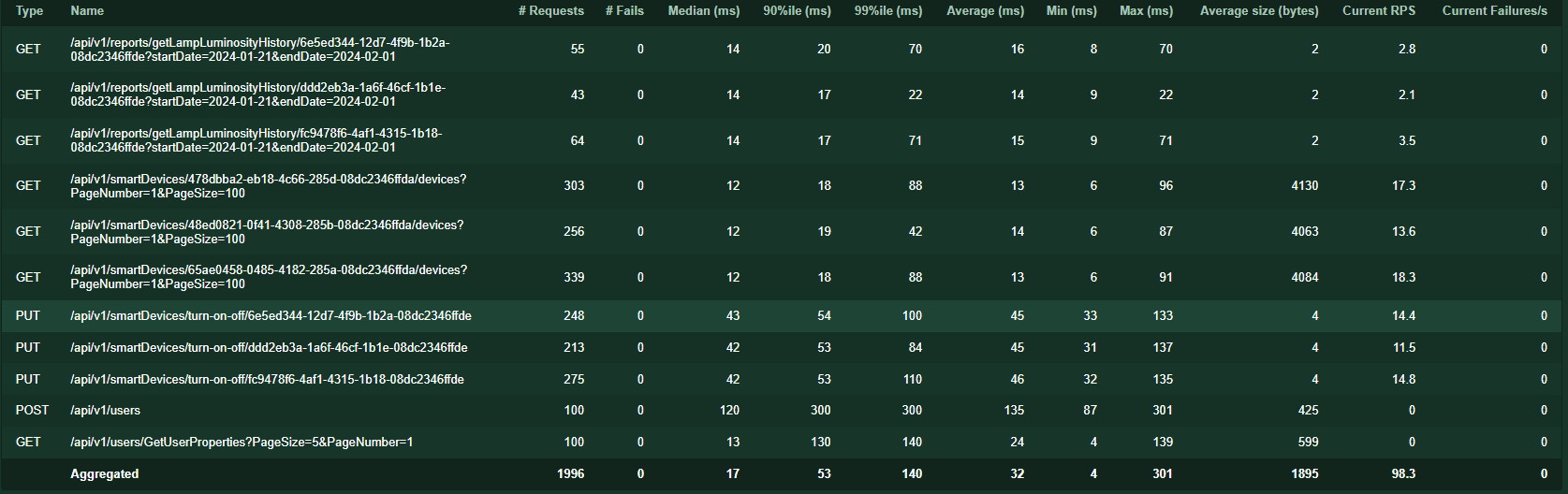
4.1 Simulator: Lampa

Testiranje je rađeno tako što je prvo dodato 1500 lampi u našu bazu podataka.

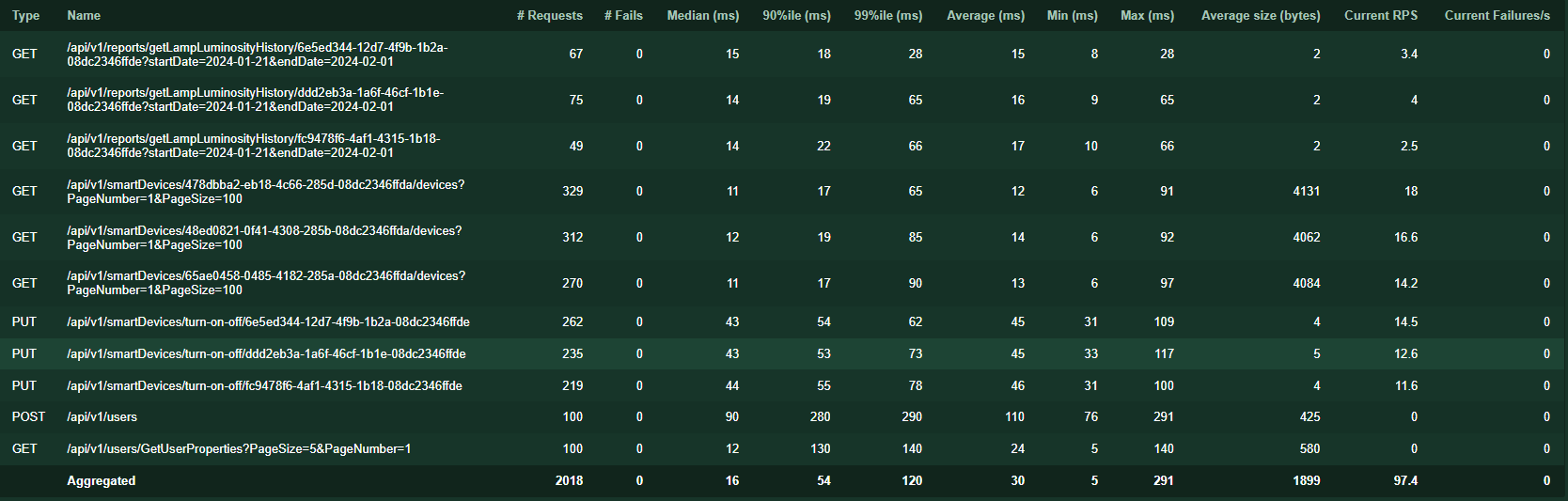
Registracija lampi

Nakon što smo dodali lampe, možemo početi sa testiranjem različitog broja simulatora.

Počeo sam sa aktivacijom simulacije 100 lampi odjednom. Nakon što je simulacija započeta u locustu sam pokrenuo skriptu koja dobavlja izveštaj lampe i koja pali i gasi lampu. Skripta je pokretana sa 100 user-a i 10 spawn rate-om.

100 simulators, 100 users

Sistem je radio glatko sa 100 simulatora uključenih i sa 100 usera. Nije dolazilo do greiaka i vreme potrebno za izvršenje akcije je bilo malo.

500 simulators, 100 users

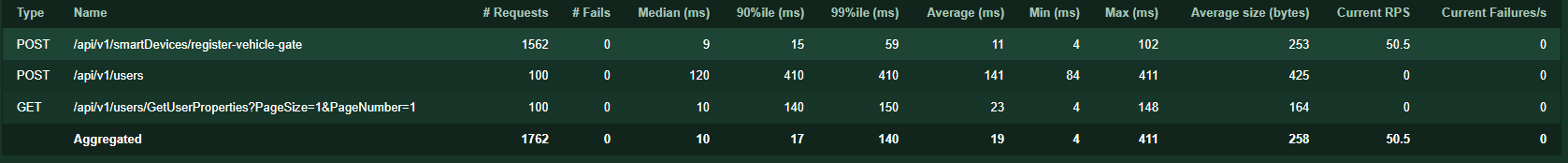
Sistem je radio jednako dobro i sa 500 uključenih simulatora lampe. Nije bio spor i nije bilo grešaka.

Nakon 500, pokušao sam sa 1000 simulatora. Ipak sistem nije uspeo da izdrži toliku količinu simulatora lampe. Baza je pukla jer jer dostignuta maksimalna veličina pool-a.

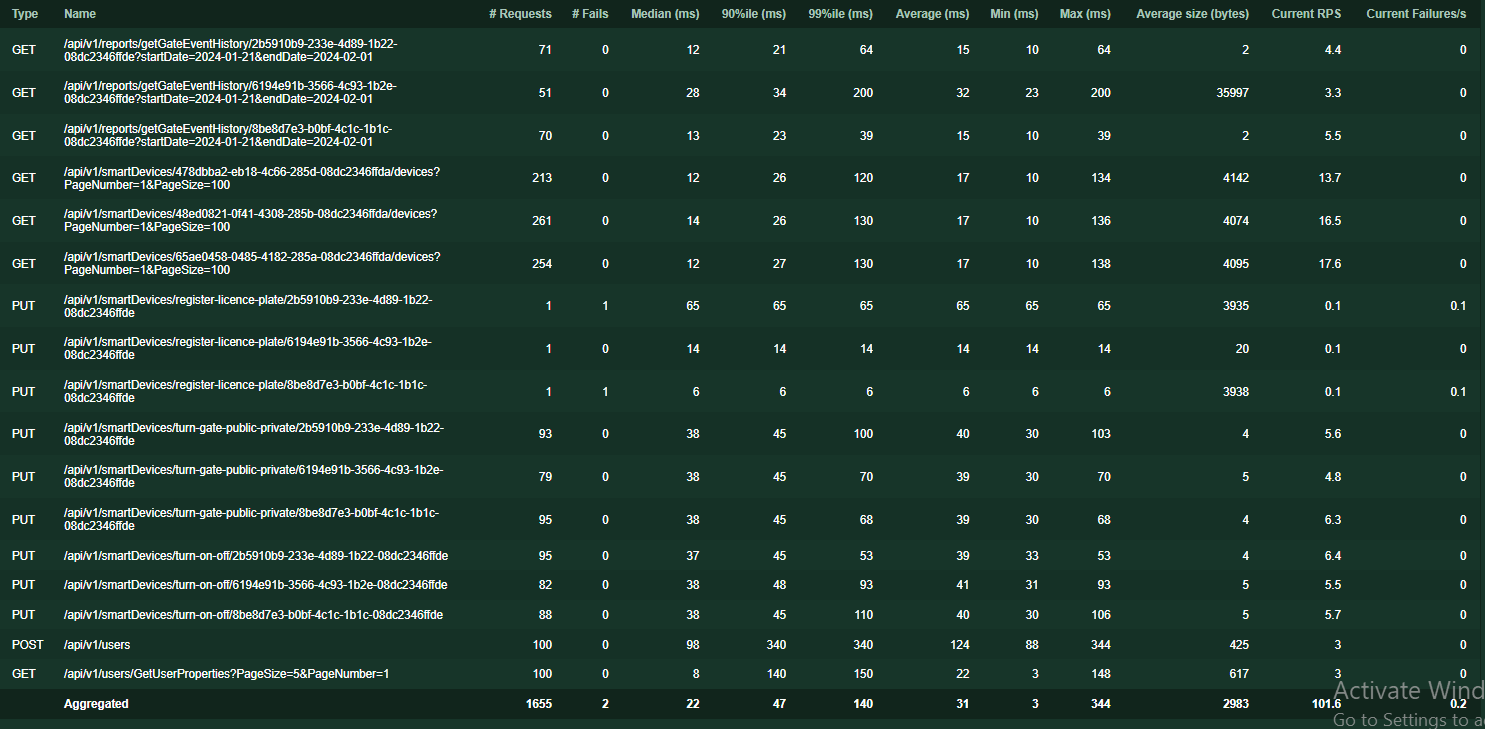
Možemo da vidimo da na 500 simulatora i 100 simulatora vreme je otprilike isto. Tako da ne usporavamo sistem sa pokretanjem veće količine simulatora. Broj user-a sam ostavio da bude isti, zato što bismo imali problema sa mqtt klijentima koji su nevezani za sam broj simulatora. Probleme sa mqtt klijentima smo ustanovili u poglavlju 3 gde su rađeni locust testovi opterećenosti.

4.2 Simulator: Kapija

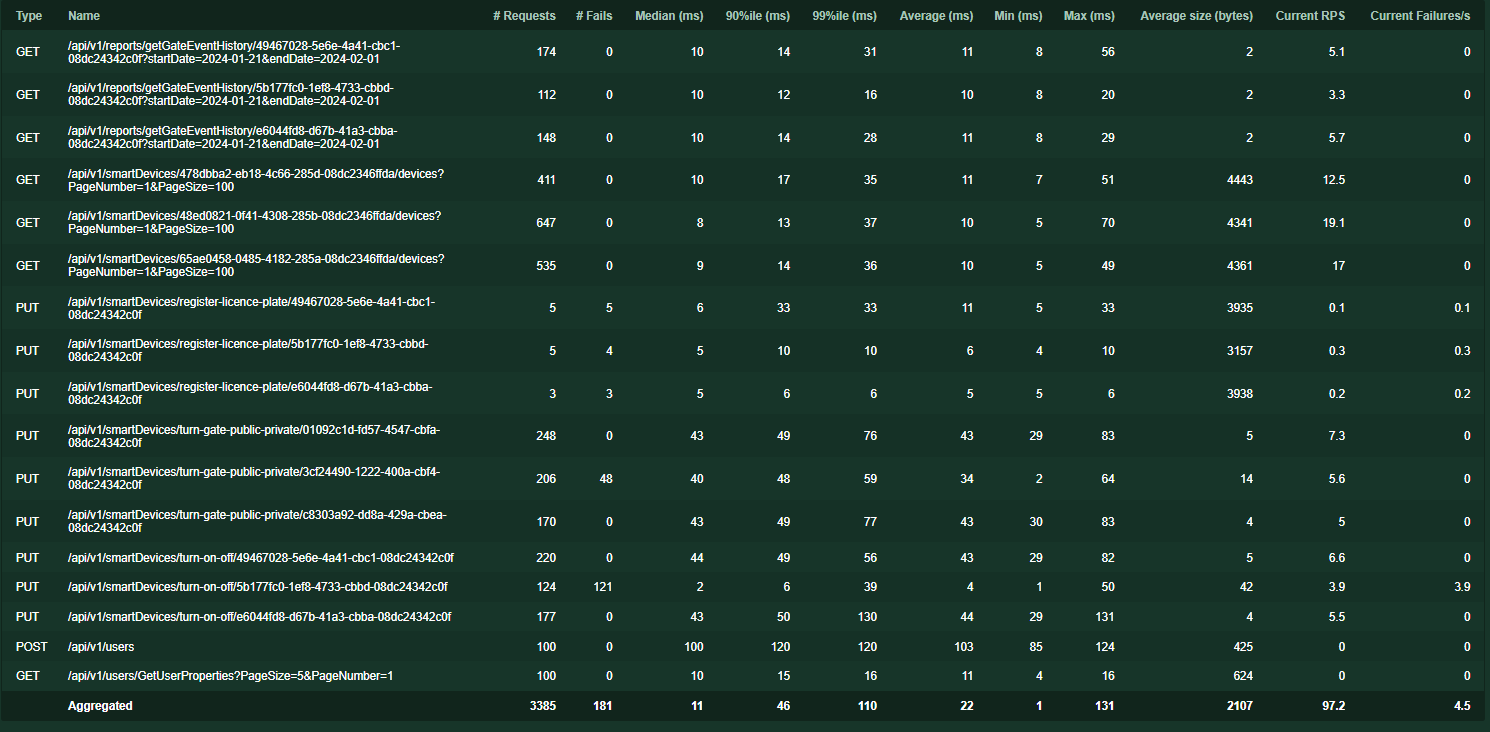
Kao i kod lampe, prvo dodajemo 1500 kapija u sistem.

Registracija kapija

Nakon dodavanja kapija, pokrećemo 100 simulatora i 100 user-a.

100 simulators, 100 users

Sistem radi dobro. Greške su zbog mqtt-a, videli smo te greške sa klijentom u poglavlju 3. Osim toga performanse sistema su dobre i nema grešaka fatalnih po sistem.

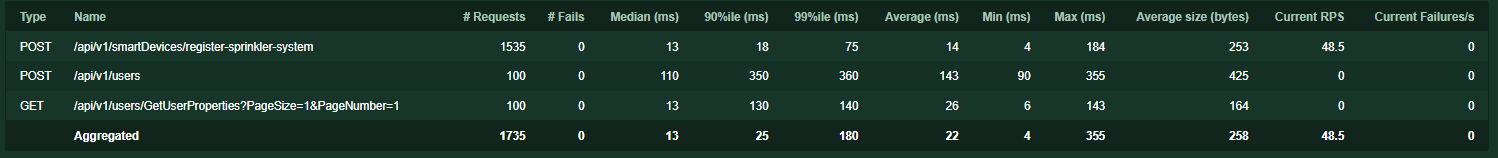
200 simulators, 100 users

U ovom testu ima više grešaka ali to su većinski errori 404, pored njih su mqtt greške, ali idalje ništa fatalno za sistem. Vidimo da je prosečno vreme čak i manje nego u prethodnom testu. Povećanjem za 100 simulatora ne gubimo performanse na vremenu.

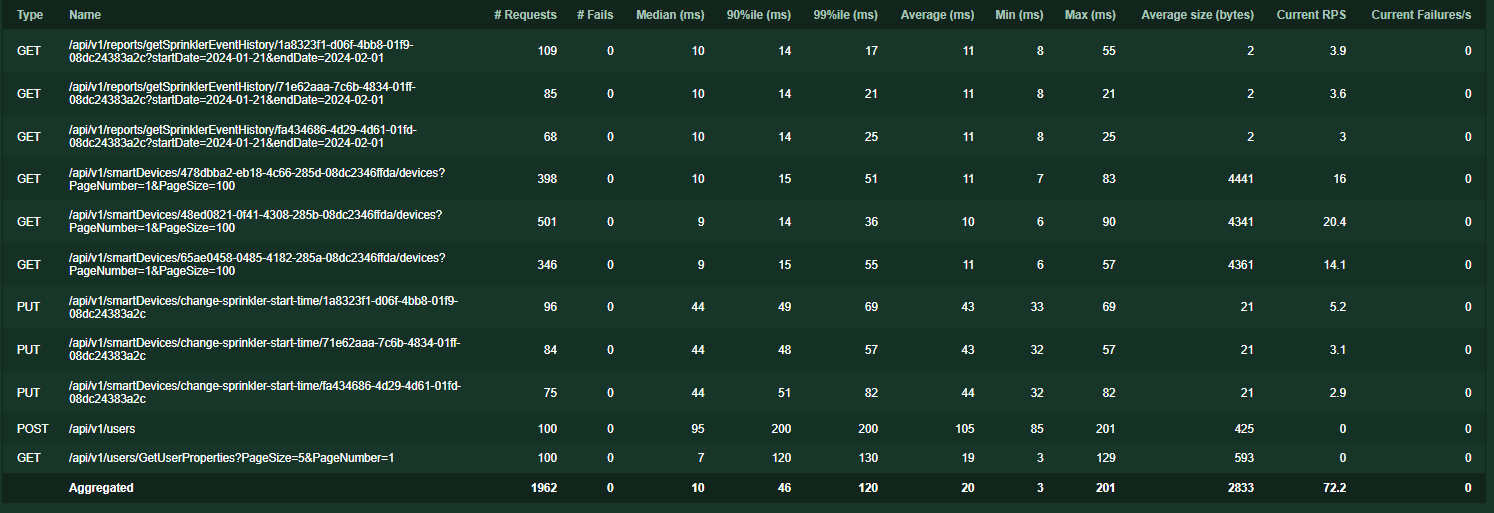
Ipak nakon ovog testa, pokušao sam sa 300 kapija. Sistem nije mogao da izdrži i baza je pukla kao kada smo pokušali sa 1000 lampi. Kapija je kompleksnija simulacija sa više akcija, čekanja i logike od lampe. Tako da je očekivano da baza puca na manjoj količini simulacija kapije.

4.3 Simulator: Prskalice

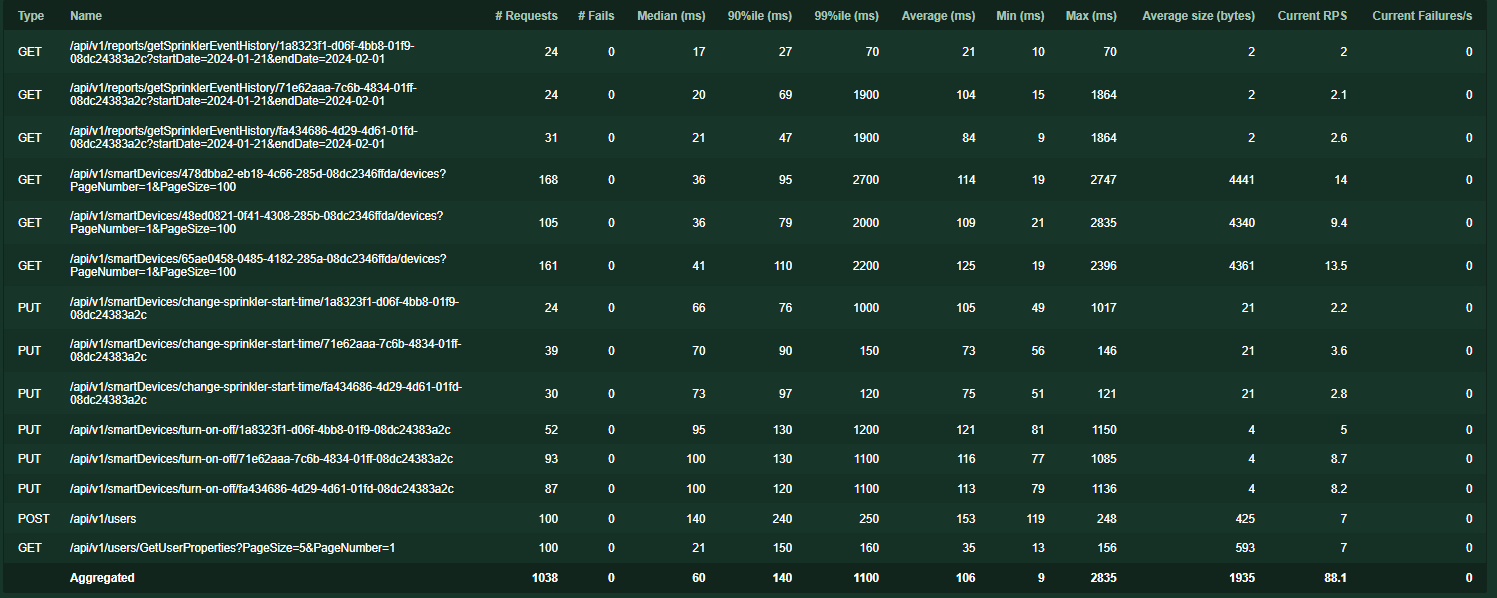
Kao i u prethodnim simulatorima, prvo dodajemo 1500 prskalica u sistem.

Registracija prskalica

Nakon registracije prskalica pokrećemo 500 prskalica.

500 simulators, 100 users

Sistem radi bez grešaka, takođe performanse su dobre.

5000 simulators, 100 users

Ovaj ogromni skok u količini simulatora je napravljen zato što sistem prskalica u svojoj simulaciji mora da prođe kroz puno if-ova kako bi uopšte uradio nešto.Pre ovog testa sam dodao još 3500 prskalica, kako bih mogao da testiram na 5000 simulatora. Samo pokretanje ogromnog broja simulacija ne kvari sistem. U realnom sistemu se ne bi dešavalo da toliki broj prskalica u isto vreme prođe sve provere. Ovde vidimo slabljenje u performansama tako da porast količine simulatora ipak utiče na performanse. Taj uticaj je veoma mali s obzirom na razliku u količini simulatora koja se pokreće.

4.4 Utisci o testiranju simulatora

Testiranje simulatora je bilo zanimljivije i drugačije od simulacije opterećenja. Razlog je taj što su simulacije od 3 tipa uređaja koje sam testirao različite. Kapija kao najkompleksnija simulacija koja konstantno šalje podatke, ima sleep-ove i mnogo upisa ruši bazu i sistem već na 300 simulacija. Lampa koja je najjednostavnija ipak uspeva da izdrži 500 simulacija. A sprinkler koji je složeniji, ali samim time i ređe odrađuje operacije može da podrži najveći broj simulacija. U testovima se videlo da postoji razlika u performansama sistema u zavisnosti od količine pokrenutih simulacija ali nije previše značajna. Baza je usko grlo sistema i ona puca kada preteramo sa simulatorima.

1. Zaključak

Na osnovu rezultata testiranja izvršenih u okviru ovog izveštaja, zaključujemo da softversko rešenje za monitoring pamente kuće pokazuje solidne performanse u većini testiranjih scenarija.

Najveći problemi koje smo imali su problemi sa mqtt-om koji se vide u sekciji 3, problemi sa bazom koji se vide u sekciji 4 kao i problemi sa samom konfiguracijom koje nisam uspeo da rešim. Potrudio sam se da poboljšam performanse koliko god je to moguće. Dodavanjem paginacije, downsamplinga. Kolega je dodao indeksiranje u bazi, takođe smo izmenili na koliko često simulatori čitaju vrednost da ne bude zakucan broj već random, kako ne bismo dobili previše zahteva u istom momentu. Trudili smo se tokom izrade celog projekta da pazimo na performanse sistema i da ih koliko je u našoj mogućnosti poboljšamo. Pri pokretanju svih stvari potrebnih za rad aplikacije, kao i testova, dešavalo se da memorija korišćena na mom računaru dođe i do 99%, kao i da sistem puca zbog prevelikog rada CPU-a. Tako da jedan od načina da se performanse sistema poboljšaju bi bio pojačavanje samog računara na kojem se sistem pokreće, kao i uklanjanje rada nepotrebnih stvari na računaru. Noć pred rok kolega je pronašao grešku koja je prouzrokovala mqtt errore. Ostavljeni su grafici kao I izveštaji performansi u scenarijima 7 i 9 nakon ispravke greške. Sistem je sada u mogućnosti da podnese znatno veći broj korisnika bez pucanja. Zbog manjka vremena i kako ne bismo poremetili sistem koji je već bio spreman za odbranu nisam uradio sve testove nakon ispravljanja greške. Ali rezultati su očevidni u scenarijima 7 i 9.Smatram da svi testovi koji su imali problem sa mqtt-om i sa greškom da klijent nije konektovan više nemaju te probleme.