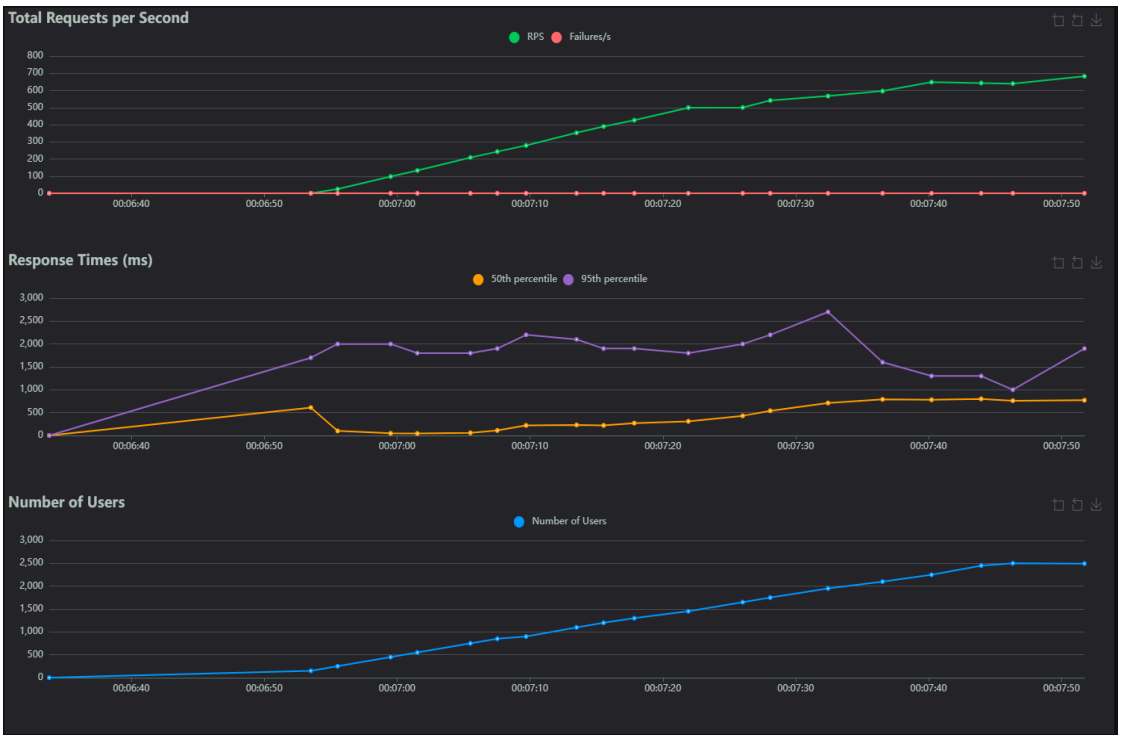
Testovi opterećenja
Student 3 Vladimir Čornenki SV53-2021

Svi testovi su pokretani na istom računaru kao i cela platforma i simulatori. Specifikacija računara: Ryzen 5 5600x sa 16GB RAM-a. Svi requestovi su išli preko nginx-a a ne direktno na go gin server. Svi testovi su rađeni sa Locustom.

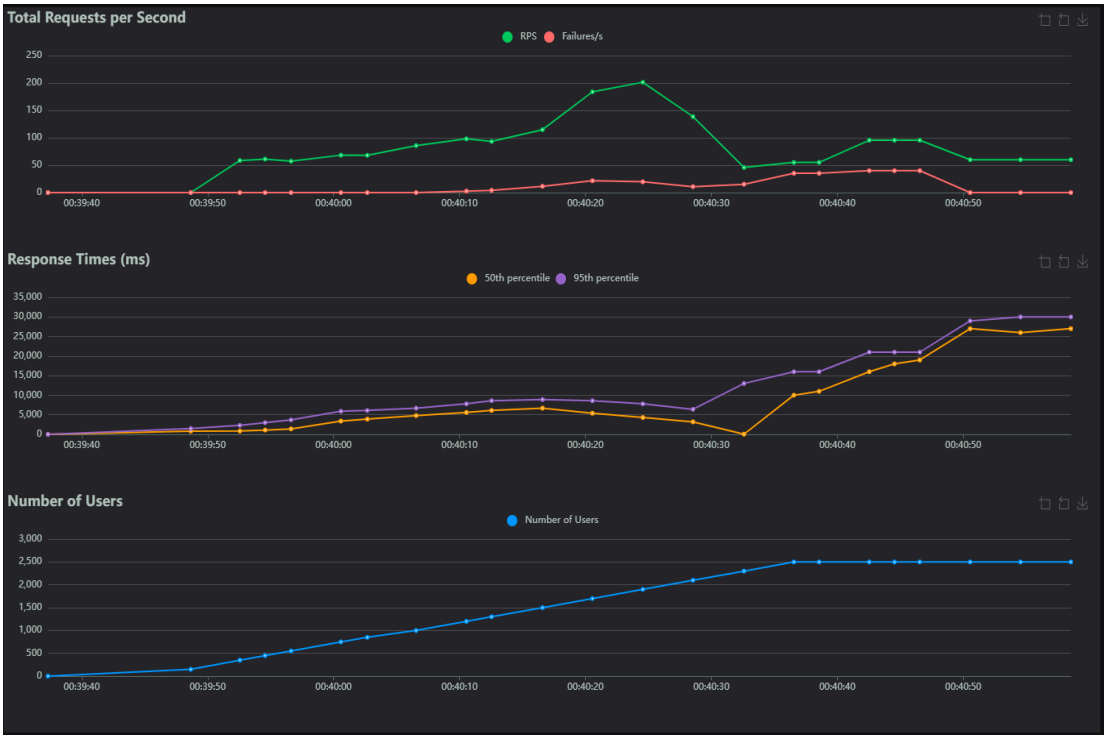
1. Pretraga property-a. Ovaj scenario predstavlja funkcionalnost obicnog usera, u našem sistemu Regular, koji se uloguje i zatim izvršava pretrage. U sklopu ovog scenarija testirao sam i filter i kompleksniji search sa više parametara.



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/api/login	2000	7	2200	12000	16000	3315.52	208	17837	230.19	13.3	0
GET	/api/property/query	26333	0	620	1300	1600	592.83	2	2246	27	668.8	0
	Aggregated	28333	7	660	1800	5400	785.03	2	17837	41.34	682.1	0

Performanse pod opterećenjem su zadovoljavajuće. Sistem uspeva da odgovori i do preko 600 yahteva u sekundi sa zanemarljivim brojem grešaka. Vreme odziva na login raste porastom broja korisnika (što će se videti i u narednim testovima). Endpoint koji je bio cilj ovog testa pokazuje odlične performanse sa prosečnim trajanjem od 592,83.

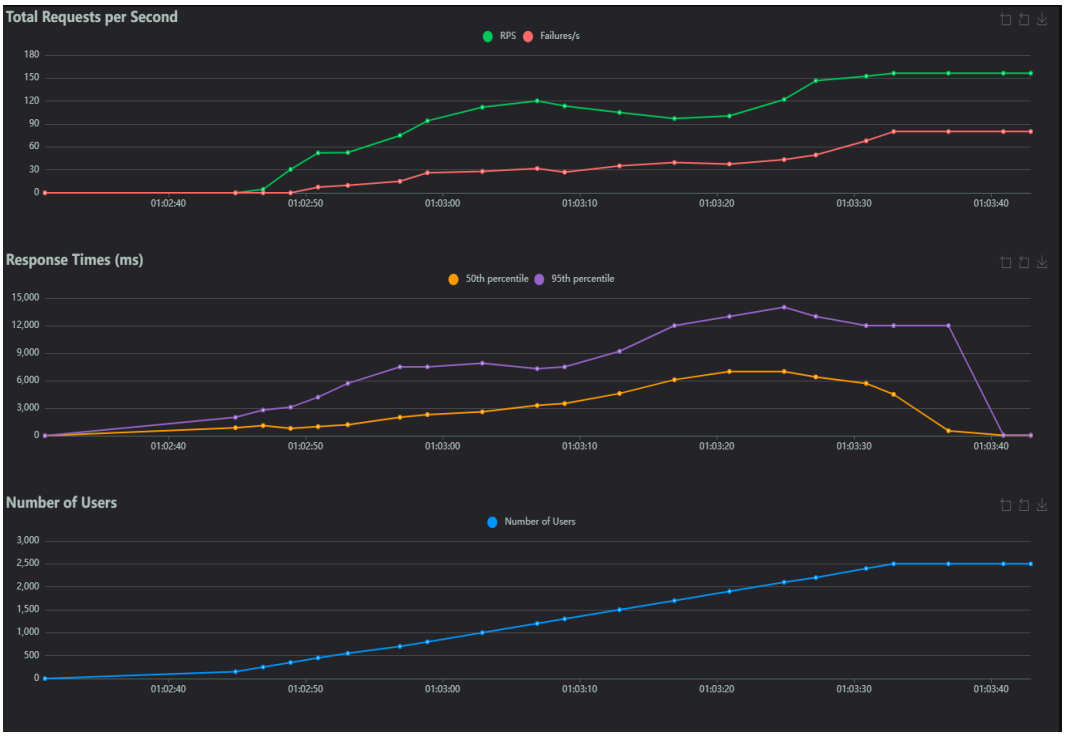
2. Kreiranje novog zahteva za property. Ovo je scenario u kojem korisnik kreira zahtev za property. Korisnik mora prvo uspešno da se uloguje pa da pošalje zahtev za request. Ovaj zahtev je poprilično skup jer obuhvata kreiranje domaćinstava i njihov zapis u bazi podataka kao i čuvanje slika i dokumenata domaćinstva.



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/api/login	2498	745	3600	15000	25000	4525.55	23	44259	165.12	0	0
POST	Create Property	4362	0	7500	35000	47000	13343.25	36	48556	1617.52	0	0
	Aggregated	6860	745	6000	31000	46000	10132.37	23	48556	1088.64	0	0

Prilikom porasta opterećenja zbog skupoće ovog zahteva primećujem skokove u vremenu potrebnom za odgovor i pad broja zahteva u sekundi. Takođe primećujem veći broj neuspelih login zahteva.

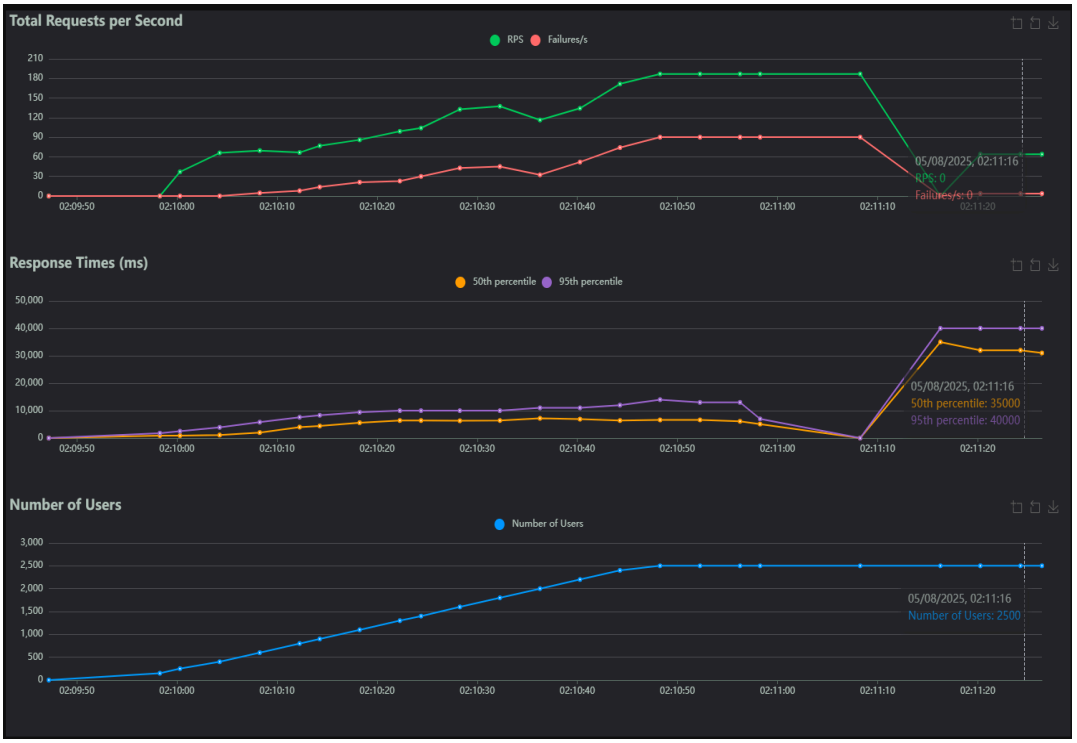
3. Admin prihvata property zahtev. Scenario u kom sam testirao end to end kreiranje i prihvatanje propertya. Korisnik se uloguje i podnosi zahtev za novi property. Zatim admin koji je već ulogovan potvrđuje taj zahtev.



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/api/login	4117	1951	3600	9200	12000	3929.17	5	15947	121.83	0	0
PUT	Accept Property as Admin	475	2	5800	16000	19000	7372.94	52	60003	50.49	0	0
POST	Create Property as User	735	0	6000	13000	15000	6135.92	34	15728	991.32	0	0
	Aggregated	5327	1953	4000	11000	15000	4540.73	5	60003	235.44	0	0

U ovom testu vidimo masovan broj neuspelih login zahteva. Ovo je poprilično skup test sa kompleksnim zahtevima. Primećujem i visoko vreme odziva. Pokušao sam da analiziram i dođem do rešenja za ovaj problem. Pokušao sam da optimizujem gin framework postavljanjem u release mod i korišćenjem alata pprofile, ali nisam uočio ništa sporno kod gin-a. Zatim sam pokušao da eksperimentišem sa connection pool-om baze podataka i to mi je malo popravilo rezultate, ali ne dovoljno da budem zadovoljan. Na internetu sam pročitao nekoliko komentara ljudi koji se žale na sporost GORM-a. Spor orm može značajno da ugrozi performanse, iako ne sumnjam da bi se problem rešio horizontalnim skaliranjem, a verovatno i vertikalnim.

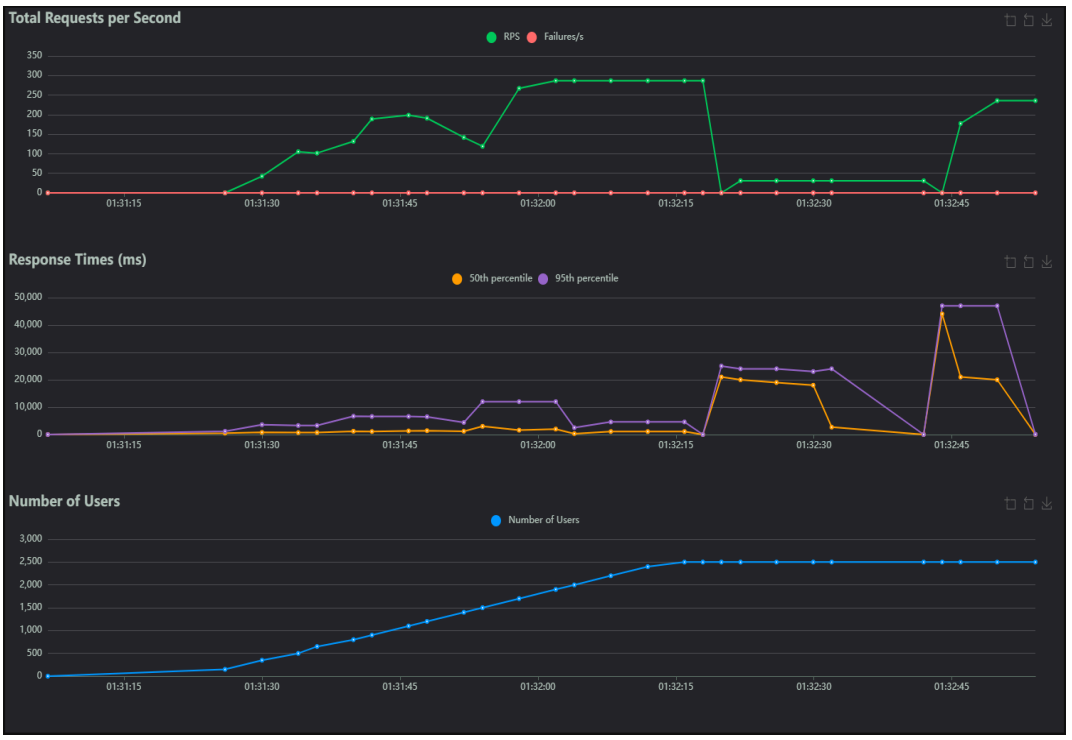
4. Admin odbija zahtev za property. Poput prethodnog zahteva, korisnik kreira zahtev za property, ali ovog puta admin odbija zahtev.



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/api/login	5098	1948	5500	29000	34000	7155.19	40	37688	141.85	19.7	3.7
POST	Create Property as User	1252	0	9700	37000	40000	14474.79	91	42906	994.28	22.8	0
PUT	Decline Property as Admin	887	0	9600	40000	44000	14113.6	42	46119	50	21.5	0
	Aggregated	7237	1948	6400	34000	39000	9274.33	40	46119	278.06	64	3.7

Ista situacija kao kod prethodnog testa. Velika degradacija performansi prilikom većeg opterećenja. Veći broj neuspelih login zahteva. Greška je uglavnom: “ConnectionAbortedError(10053, 'An established connection was aborted by the software in your host machine', None, 10053, None)”, nginx šalje status 499.

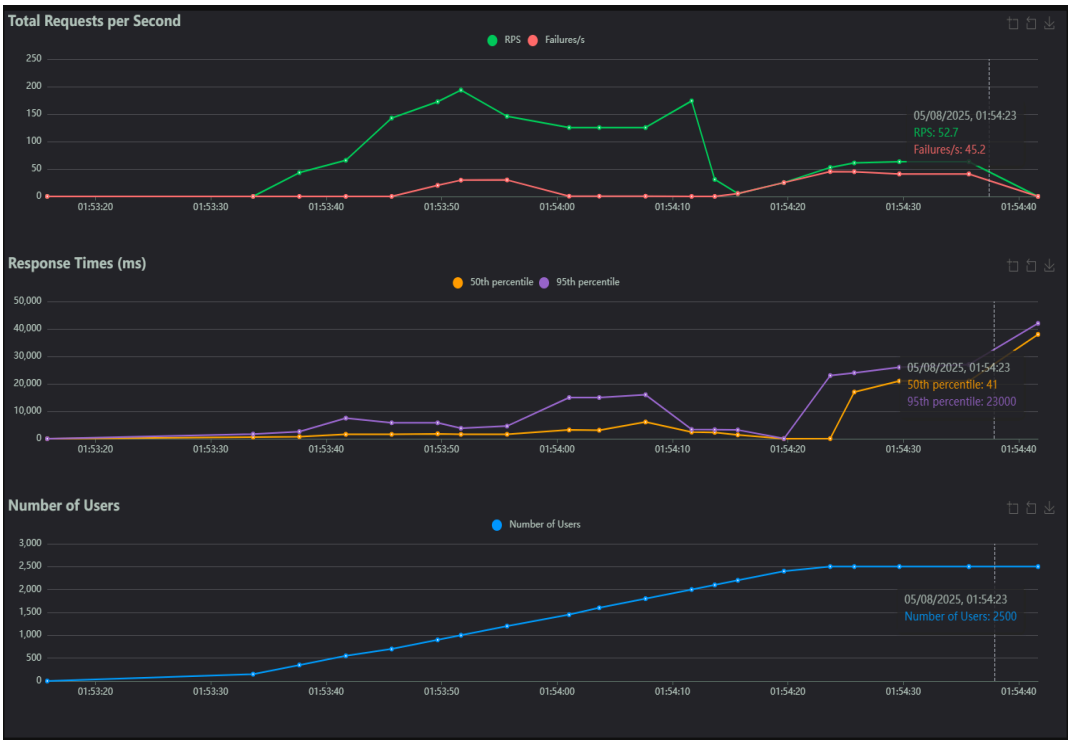
5. Davanje pristupa domaćinstvu. Ovaj scenario testira funkcionalnost vlasnika domaćinstva da da pristup tom domaćinstvu drugim korisnicima sistema. Korisnik se uloguje i pretraži domaćinstva kojih je on vlasnik. Taskovi su da vlasnik pretražuje korisnike i da im daje pristup jednom od njegovih domaćinstava.



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/api/household/query	1998	0	780	22000	23000	3657.13	21	44976	932	21.1	0
POST	/api/households/[id]/access	1268	0	230	22000	43000	3164.1	3	44422	52.72	11.2	0
POST	/api/login	1998	0	2100	22000	25000	5734.28	138	26383	231	0	0
POST	/api/user/query (Search and Prepare)	5857	0	5400	47000	47000	14424.92	2	47555	1592	203.3	0
	Aggregated	11121	0	1500	46000	47000	9645.07	2	47555	1053.4	235.6	0

Primećujem solidne performanse do određenog broja konkurentnih korisnika, recimo 1500-2000. Nakon toga vidimo skokove u vremenu izvršavanja zahteva, a posle nekog vremena i ogroman pad u broju zahteva po sekundi i još većim oscilacijama u vremenu izvršavanja.

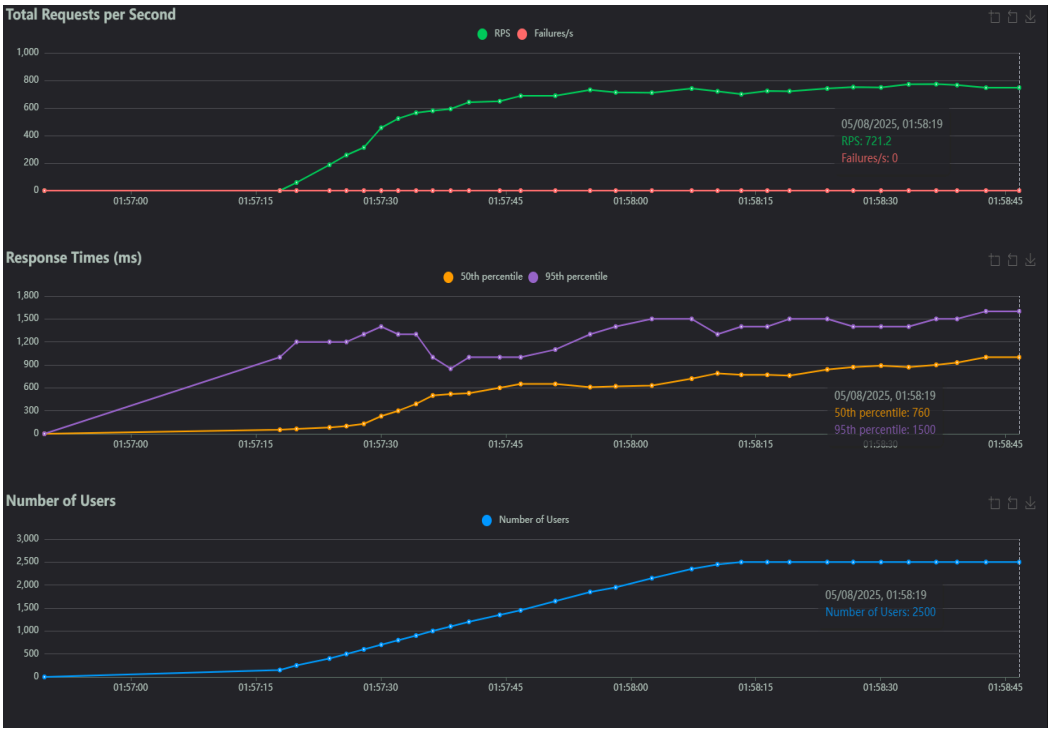
6. Oduzimanje pristupa domaćinstvu. Testiram scenario u kojem vlasnik domaćinstva oduzima vlasništvo korisniku koji ima pristup njegovom domaćinstvu. U ovom scenariju ulogovani vlasnik prvo dodeljuje pristup a zatim ga oduzima.



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/api/household/[id]/access (Grant)	1558	196	2400	60000	60000	19049.51	182	60324	69.24	13.2	0.3
DELETE	/api/household/[id]/access/revoke/[userId]	921	769	38000	56000	56000	30681.97	8	56544	39.24	34.1	28
POST	/api/household/query	1351	52	810	40000	60000	4134.1	23	60119	902.56	1.6	0
POST	/api/login	2500	502	2300	24000	27000	6530.8	21	27690	184.62	0	0
POST	/api/user/query (Search)	1629	0	8900	18000	40000	9422.4	3	40363	1592	0	0
	Aggregated	7959	1519	2700	56000	60000	11961.11	3	60324	555.13	48.9	28.3

Ovo je dosta složen scenario u kojem se sistem baš opterećuje što možemo primetiti i po performansama. Veliki broj fail-ova revoke zahteva je slučaj u kom taj pristup ne postoji. Primećujemo da je pri maksimalnom broju korisnika vreme izvršavanja zahteva ogromno i da su većinom greške.

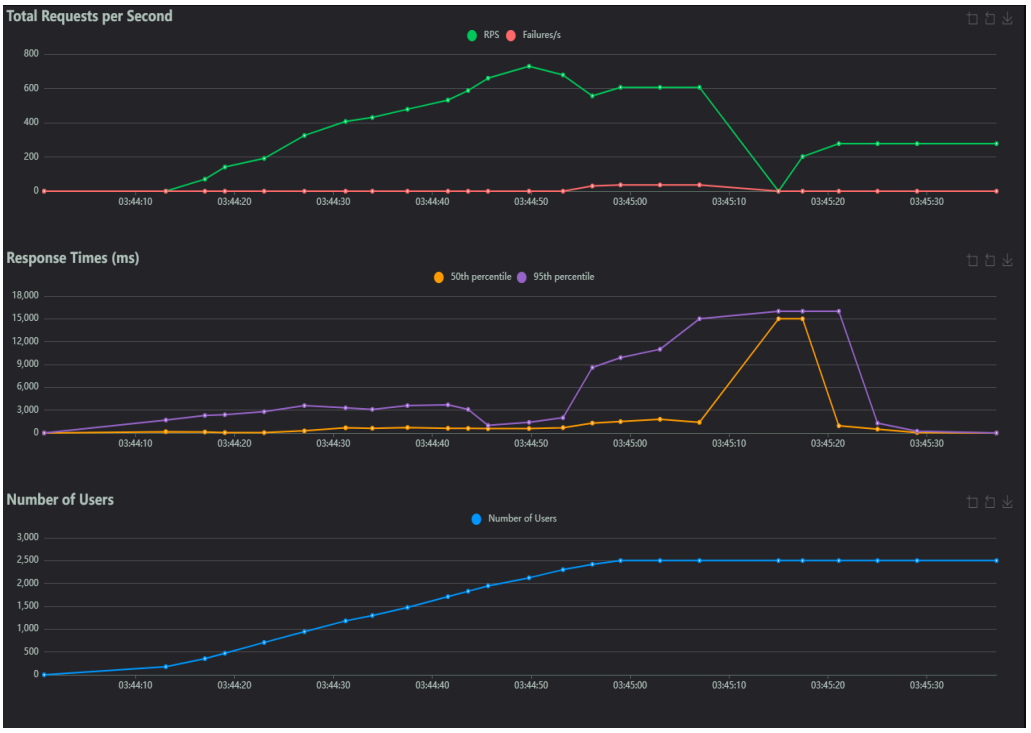
7. Potrošnja struje po gradu. U ovom scenariju admin dobija uvid u ukupnu potrošnju u izabranom gradu. Testirao sam 3 različita zahteva, potrošnju za poslednjih sat vremena, za poslednjih 12 sati i za prethodnu godinu.



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
POST	/api/consumption?period=12h	19046	0	700	1400	1700	731.52	2	2206	661.07	234.9	0
POST	/api/consumption?period=1h	37589	0	700	1400	1600	725.47	2	2214	661.41	459.1	0
POST	/api/consumption?period=1y	3774	0	710	1400	1600	736.85	2	2103	660.82	46.7	0
POST	/api/login	1370	0	2900	52000	58000	12262.57	165	60261	193	6.4	0
	Aggregated	61779	0	710	1400	3800	983.87	2	60261	650.88	747.1	0

Ovaj scenario je najuspešniji pod velikim opterećenjem. Može da odgovori na preko 700 zahteva u sekundi, a ima i malo vreme izvršavanja. Zašto ovaj zahtev radi mnogo bolje od ostalih? Ovaj zahtev ne pristupa PostreSQL bazi! Ovaj zahtev samo šalje influx query. Vidimo i ovde da login zahtev ima poveće vreme izvršavanja, ali ne koči previše sistem. Iz ovog testa možemo da zaključimo da je gotovo sigurno problem do baze podataka ili gorma. Mogući nedostatak je keširanje na nivou zahteva baze podataka. Mi smo implementirali keširanje samo na nivou API ruta. Ali svakako bi to keširanje doprinelo poboljšanju samo GET zahteva, a većina zahteva koja koči naš sistem je POST ili PUT. Zaključak je da se mora baza optimizovati ili skalirati i da se otkrije zašto GORM ne zadovoljava potrebe.

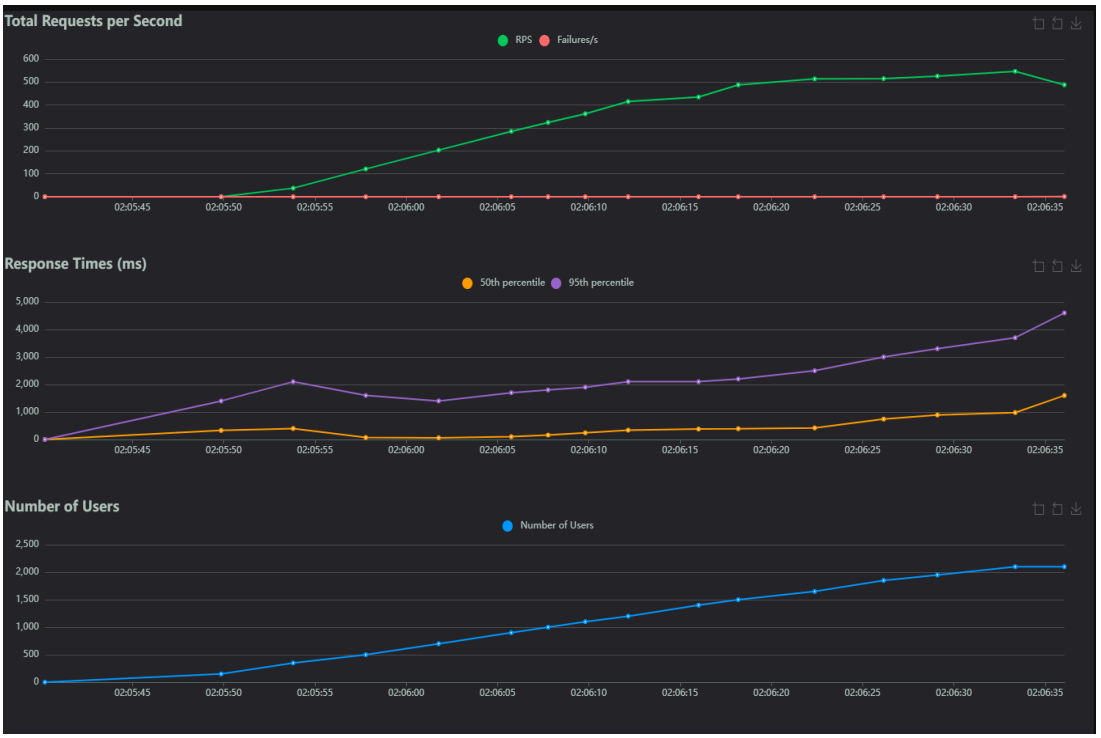
8. Pregled računa. U ovom scenariju vlasnik domaćinstva može da vidi jedan svoj račun. Na početku se vlasnik uloguje i pošalje zahtev da dobavi listu svih svojih računa. A zatim se testira dobavljanje jednog specifičnog računa iz te liste.



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/bills/search	1907	0	440	16000	16000	1518.84	2	16285	5671	11.2	0
GET	/api/bills?id=[billId]	25163	0	680	15000	16000	1828.54	2	16190	582.78	266.3	0
POST	/api/login	2500	593	3800	14000	17000	5040.4	179	18417	177.34	0	0
	Aggregated	29570	593	760	15000	16000	2080.11	2	18417	876.64	277.5	0

Na ovim graficima uočavamo dobre performanse zahteva za korisnikov račun. Ali opet primećujemo da login zahtev pravi probleme. Mogu da zaključim da GET zahtevi dobro funkcionišu zbog efikasnog mehanizma keširanja ruta. I opet mogu da zaključim da je problem u bazi podataka i ili GORM-u.

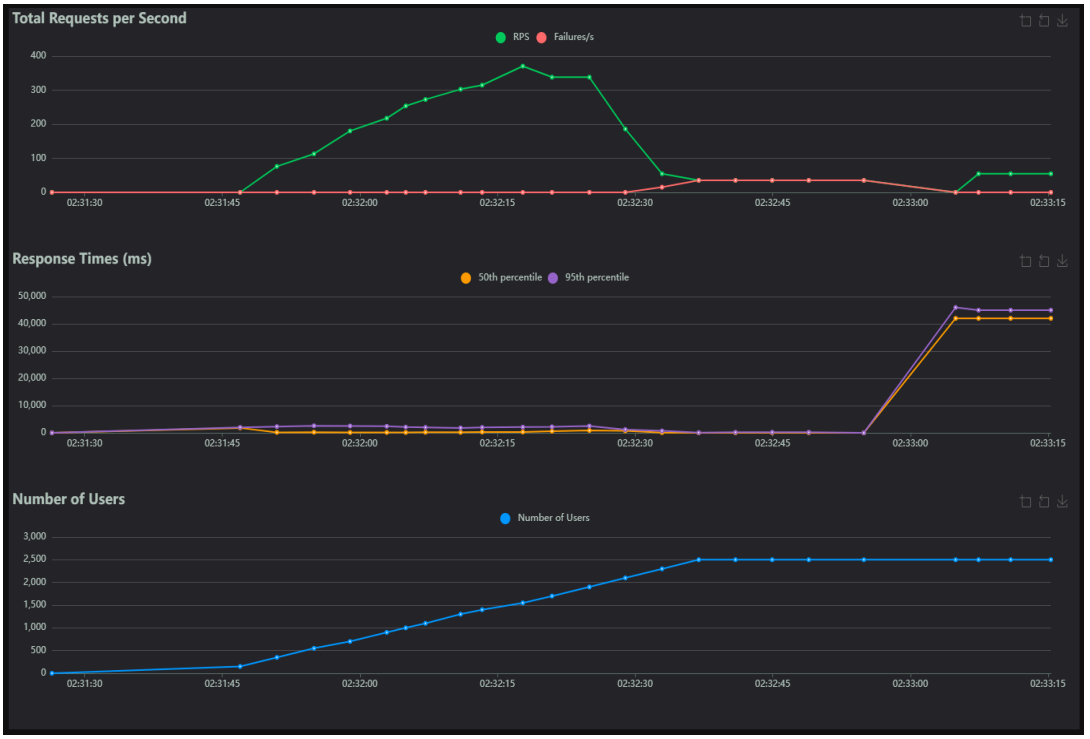
9. Pretraga računa. U ovom scenariju vlasnik domaćinstva može da pretražuje i sortira svoje račune. U okviru ovog scenarija testiram dobavljanje samo neplaćenih računa kao i pretragu računa po ceni i datumu.



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/api/bills/search	16822	0	390	1900	3800	645.55	2	4994	2513.69	453.1	0
POST	/api/login	1953	11	2200	5900	7600	2577.95	151	9603	229.7	35	1.1
	Aggregated	18775	11	470	3200	4900	846.56	2	9603	2276.1	488.1	1.1

Na ovom grafiku opet vidimo solidne performanse jer je ovo GET zahtev koji se kešira. Vidimo opet da login pravi probleme sa velikim vremenom izvršavanja i čak nekim neuspelim zahtevima.

10. Plaćanje računa. Vlasnik domaćinstva može da plati svoje račune. U ovom scenariju korisnik dobavlja svoje neplaćene račune i plaća jedan od njih.



Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
PUT	/api/bills/pay/[billId]	1065	0	460	42000	46000	4680.28	6	46553	42.85	5.1	0
GET	/api/bills/search	9938	0	330	43000	44000	6528.83	2	46496	93.88	20.5	0
POST	/api/login	2500	502	1800	44000	46000	9029.17	22	48054	184.62	28.9	0
	Aggregated	13503	502	410	43000	45000	6845.96	2	48054	106.65	54.5	0

Na ovim graficima se može videti nagli pad zahteva po sekundi pri većem broju korisnika kao i početak fail-ova, a nakon nekog vremena i ogroman skok u vremenu izvršavanja. U tabeli vidimo da nisu problem endpointi za dobavljanje računa i plaćanje već login zahtev. Moguće da je login zahtev skup zbog hesiranja lozinke, ali iz svih priloženih testova se može zaključiti da je potrebno poraditi na bazi podataka i načinu rukovanja sa modelim u bazi podataka, GORM-om.