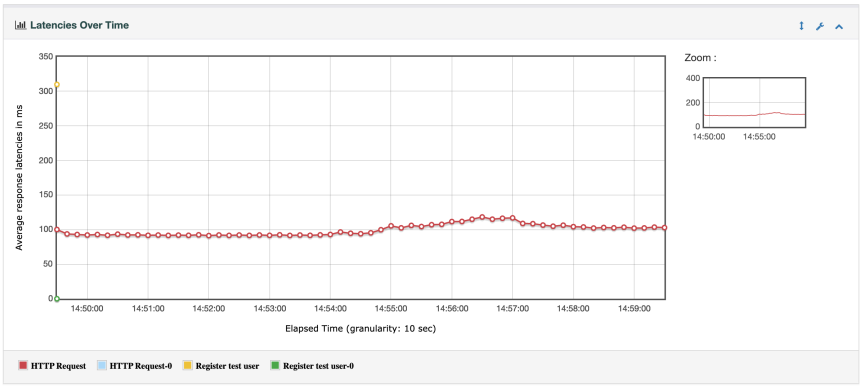
Дата проведення 28.052023

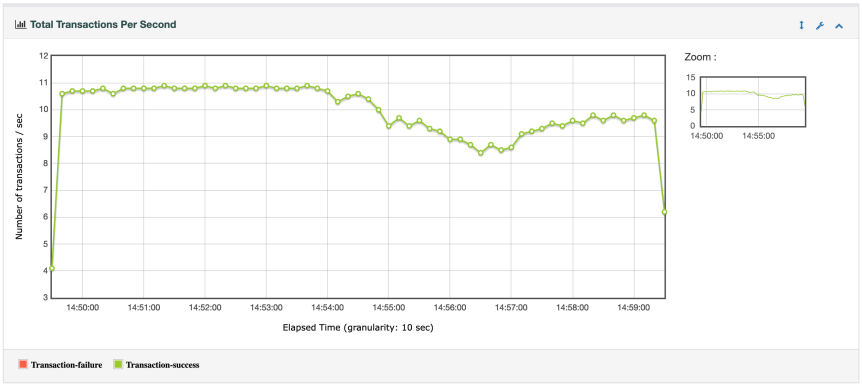
Сценарии

1. Создание нового пользователя: в один поток отправляется запрос на сервер для регистрации нового пользователя.
2. Авторизация, получение токена: обращение к серверу по полученному в п. 1 идентификатору.
3. Поиск случайного пользователя в течении 10 минут:
    1. 1 пользователь.
    2. 10 пользователей параллельно. Старт с 0 до 10 в течении минуты.
    3. Аналогичные сценарии для 100 и 1000 пользователей.

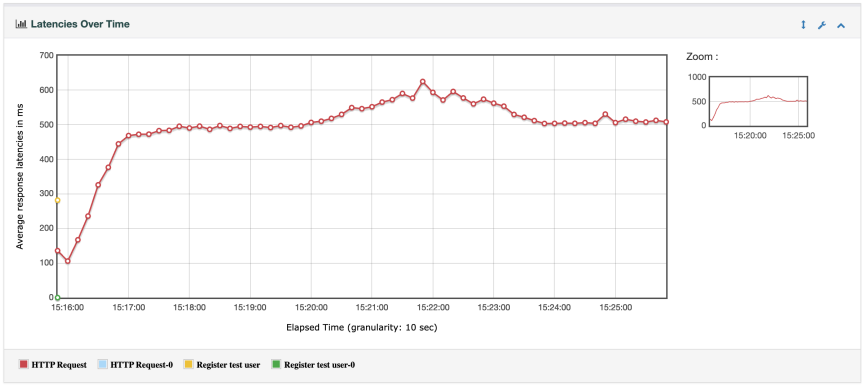
Результаты (до создания индекса)

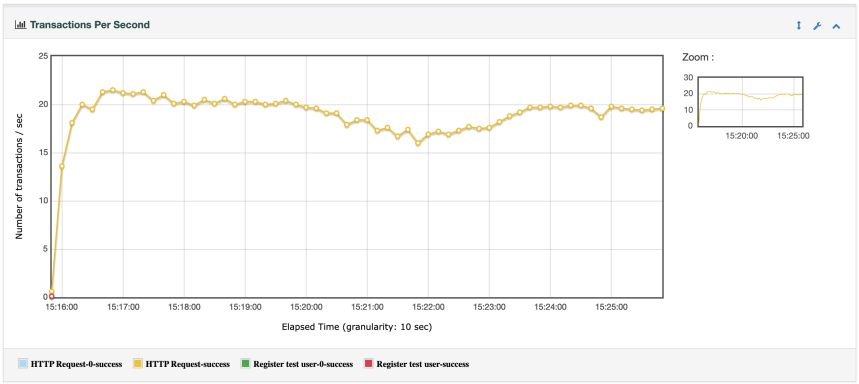
Latency и throughput для при нагрузке 1 пользователь



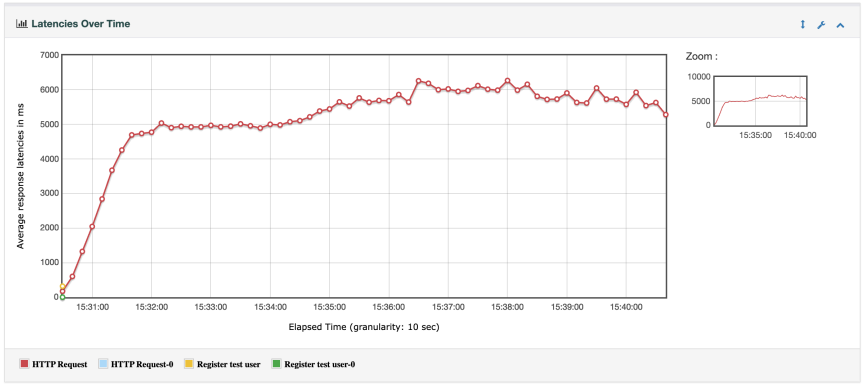


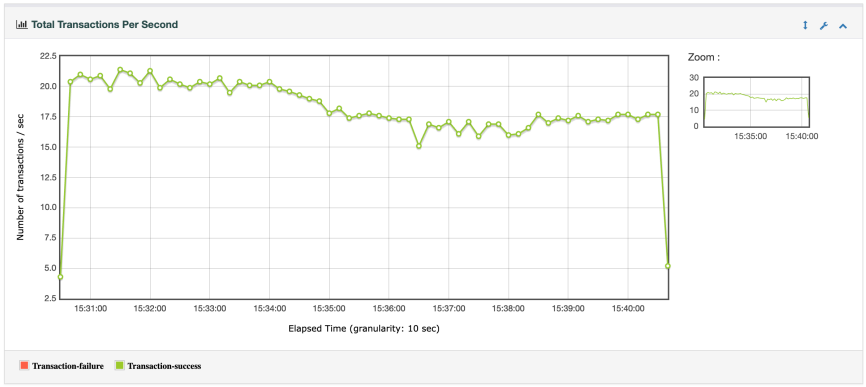
Latency и throughput для при нагрузке 10 пользователей



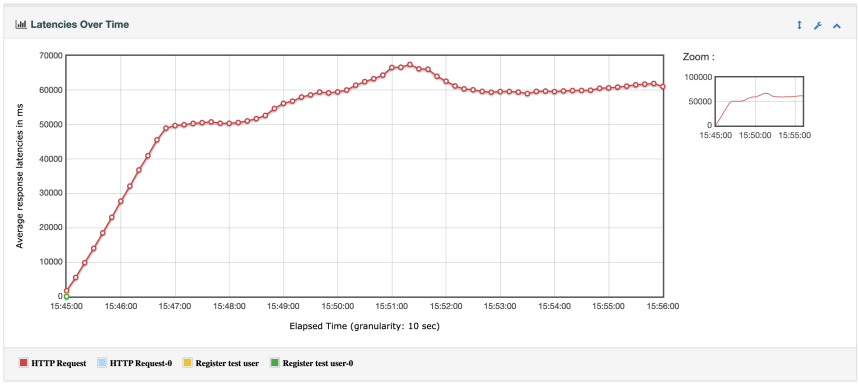


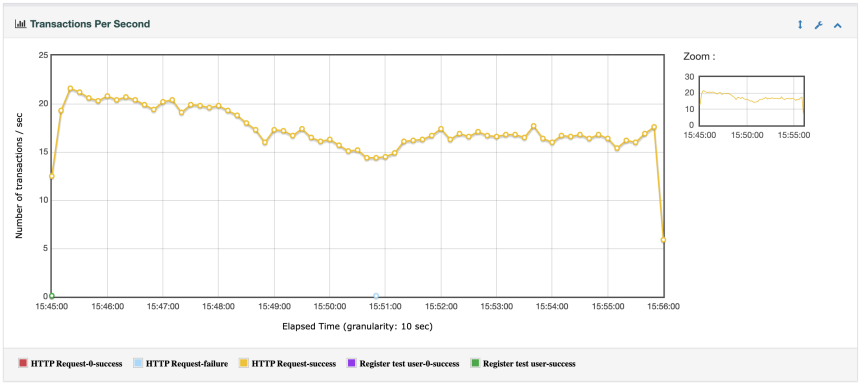
Latency и throughput для при нагрузке 100 пользователей





Latency и throughput для при нагрузке 1000 пользователей





Создание индекса

План запроса до создания индекса:

```
"Gather  (cost=1000.00..28877.89 rows=26 width=197) (actual time=116.771..120.483 rows=23 loops=1)"
"  Output: id, user_id, password, first_name, second_name, age, sex, interests, city"
"  Workers Planned: 2"
"  Workers Launched: 2"
"  ->  Parallel Seq Scan on public.users  (cost=0.00..27875.29 rows=11 width=197) (actual time=93.953..104.222 rows=8 loops=3)"
"        Output: id, user_id, password, first_name, second_name, age, sex, interests, city"
"        Filter: (((users.first_name)::text ~~ 'София'::text) AND ((users.second_name)::text ~~ 'Фетисова'::text))"
"        Rows Removed by Filter: 333332"
"        Worker 0:  actual time=98.956..98.957 rows=0 loops=1"
"        Worker 1:  actual time=68.142..98.951 rows=23 loops=1"
"Planning Time: 0.416 ms"
"Execution Time: 120.593 ms"
```

Поскольку поиск идет по двум полям, целесообразно создать индекс на эти поля в той последовательности, в которой они заданы в запросе на поиск:

```
CREATE INDEX first_name_last_name_idx ON users USING btree (first_name, second_name);
```

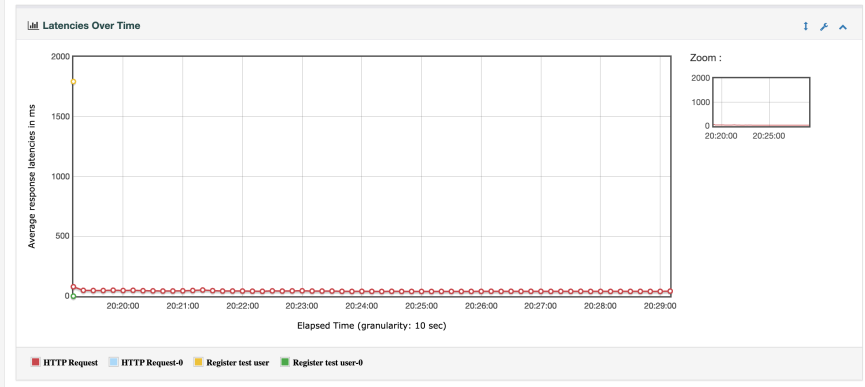
В результате - план запроса изменился следующим образом:

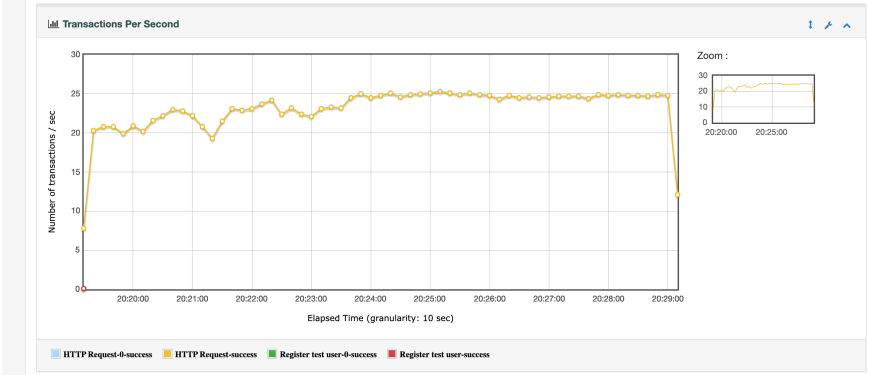
```
"Bitmap Heap Scan on public.users  (cost=4.69..106.38
rows=26 width=197) (actual time=0.086..0.126 rows=23
loops=1)"

"  Output: id, user_id, password, first_name, second_name,
age, sex, interests, city"

"  Filter: (((users.first_name)::text ~~ 'София'::text) AND
((users.second_name)::text ~~ 'Фетисова'::text))"

"  Heap Blocks: exact=8"

"  ->  Bitmap Index Scan on first_second_name_idx
(cost=0.00..4.68 rows=26 width=0) (actual time=0.066..0.066
rows=23 loops=1)"

"        Index Cond: (((users.first_name)::text =
'София'::text) AND ((users.second_name)::text =
'Фетисова'::text))"

"Planning Time: 0.297 ms"

"Execution Time: 0.289 ms»
```

При этом нужно учесть, что запрос не должен включать ‘%’ в префиксе, иначе индекс не будет использоваться.

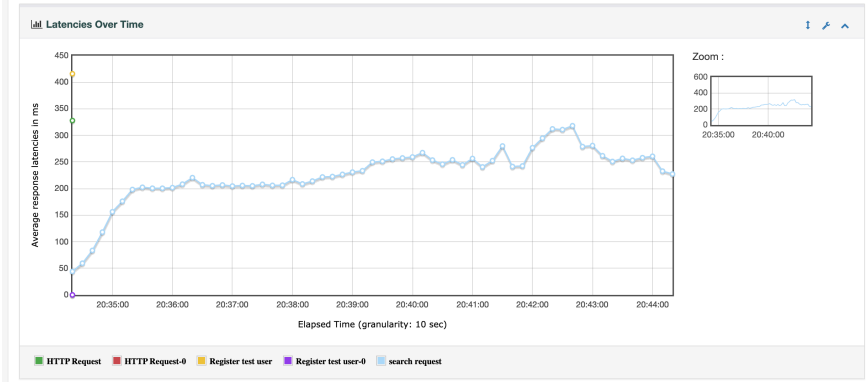
Результаты (после создания индекса)

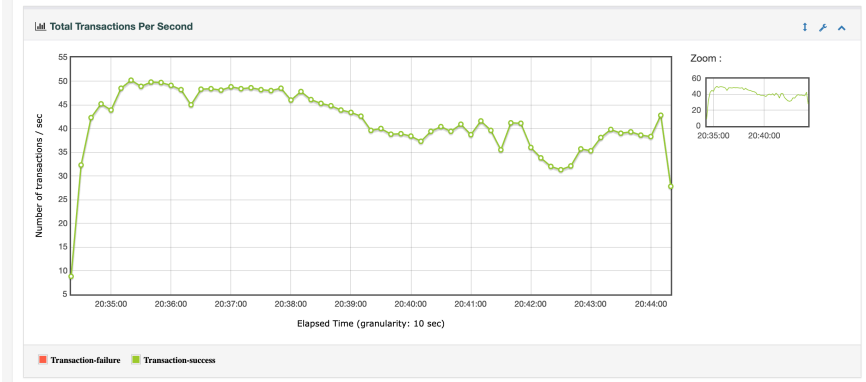
Latency и throughput для при нагрузке 1 пользователь



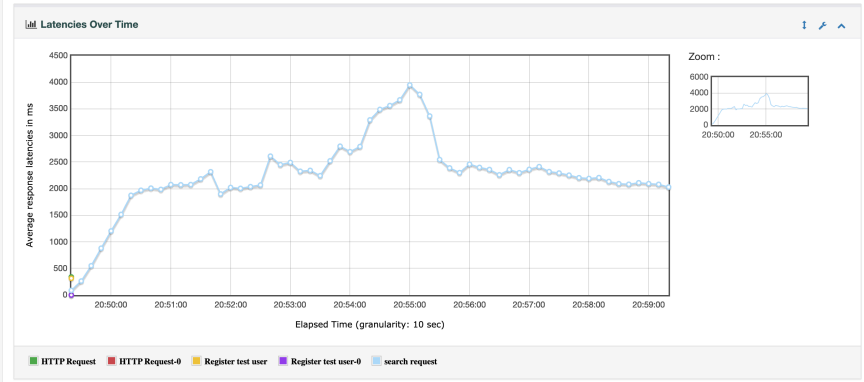


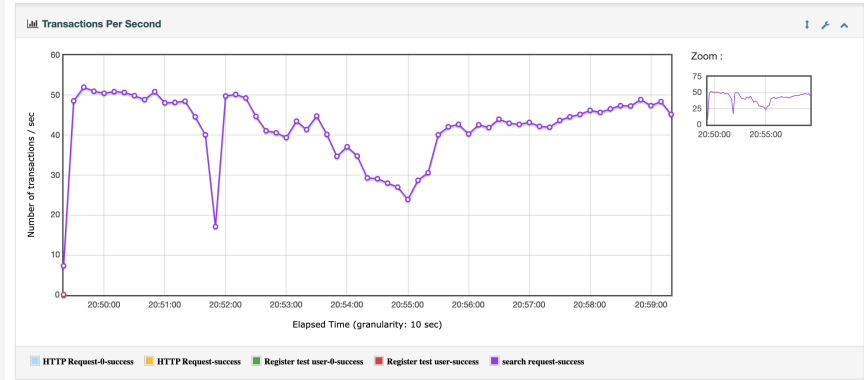
Latency и throughput для при нагрузке 10 пользователей



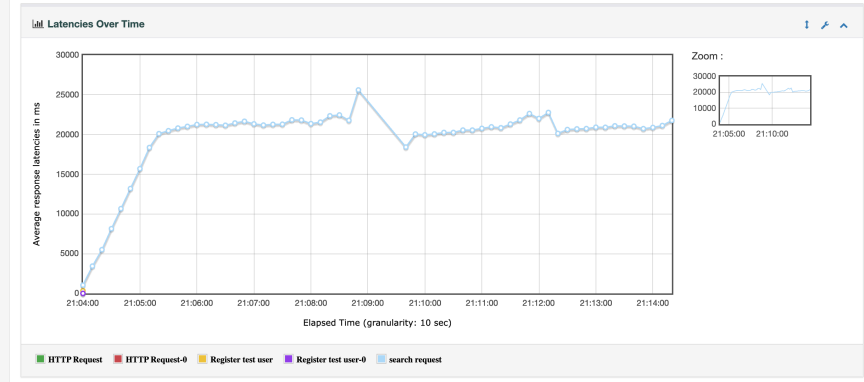


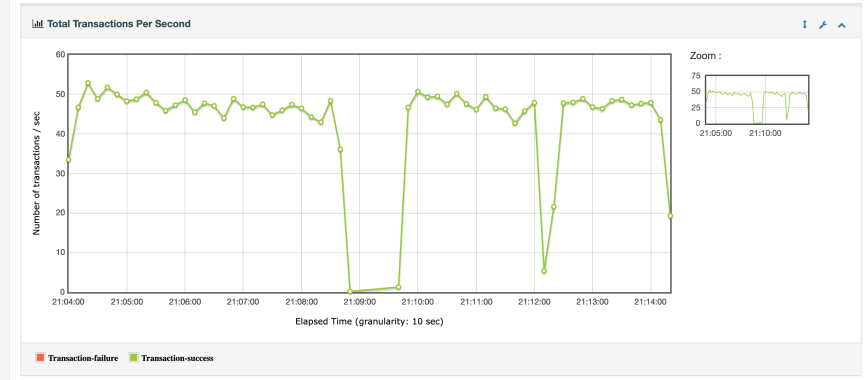
Latency и throughput для при нагрузке 100 пользователей





Latency и throughput для при нагрузке 1000 пользователей





Основные выводы

- *Основной вывод – тест успешен по основному сценарию. Добавление индекса помогло увеличить производительность.*

Рекомендации

a. *Возможно следует создать новый эндпойнт, который будет отдавать только фамилию, имя, айди пользователя. Это позволит создать покрывающий индекс и еще больше ускорить обработку таких запросов.*