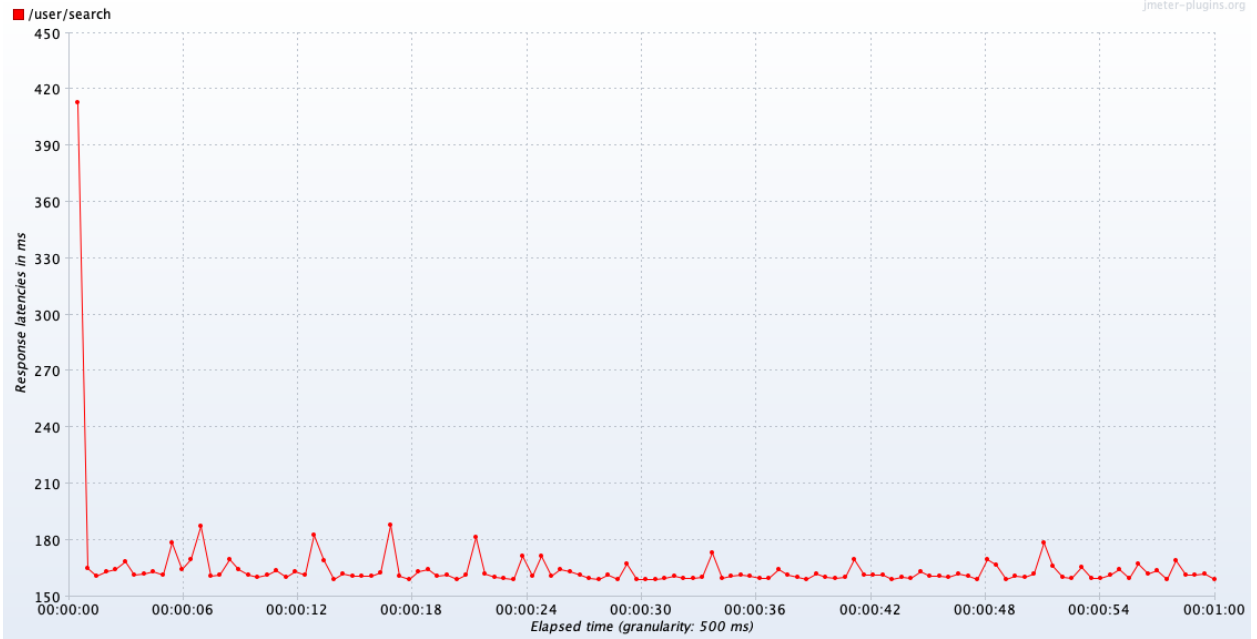
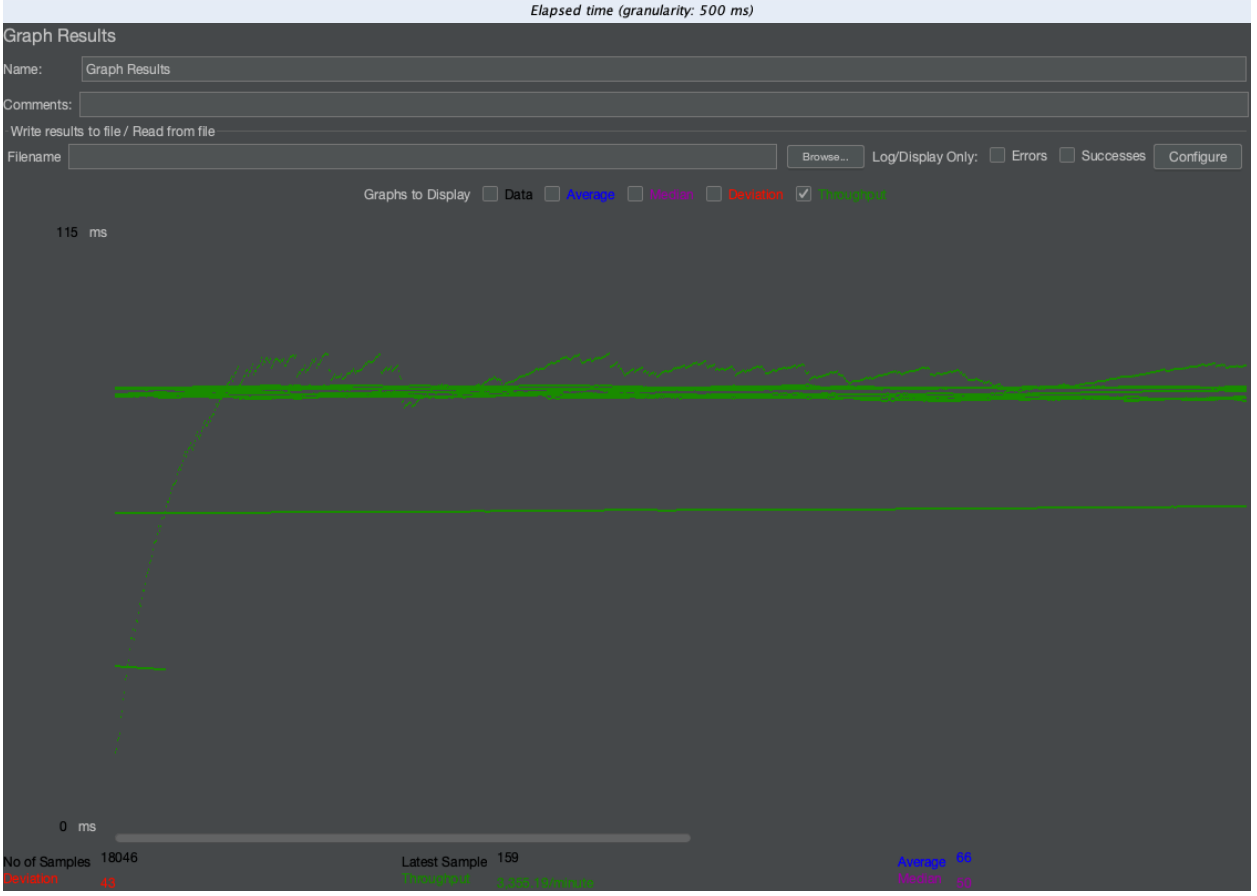
ОТЧЕТ по ДЗ№2

Проводим НТ до создания индекса. Длительность теста 60 секунд.

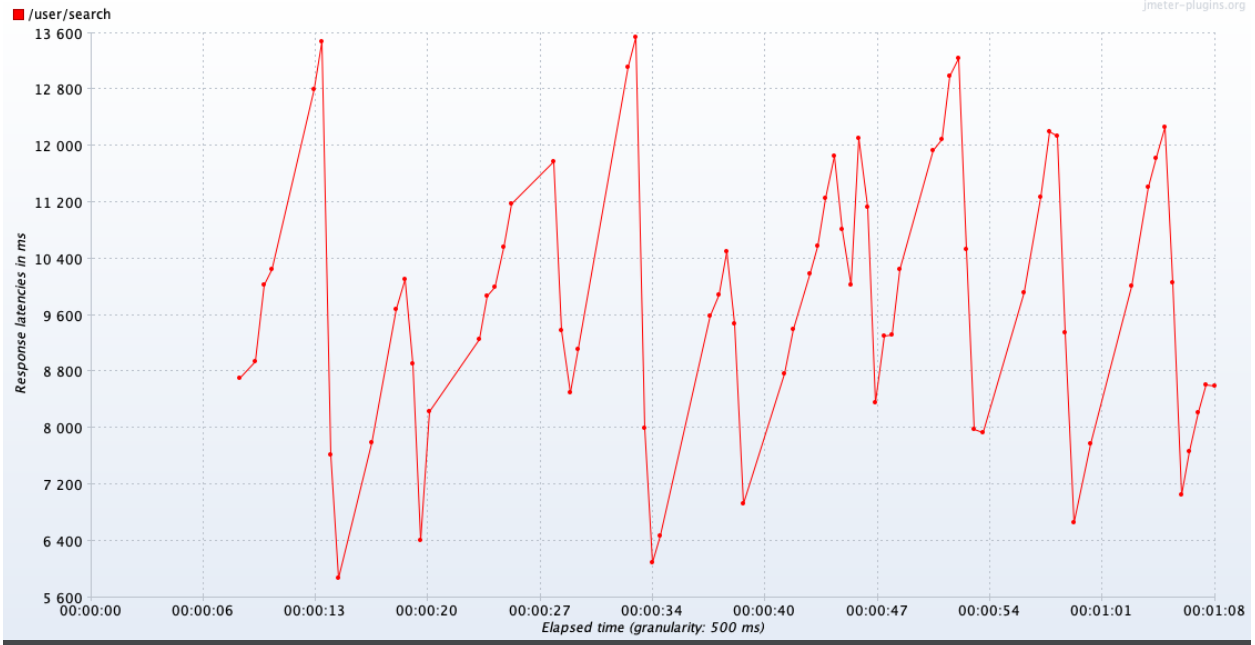
1. 1 запрос в секунду

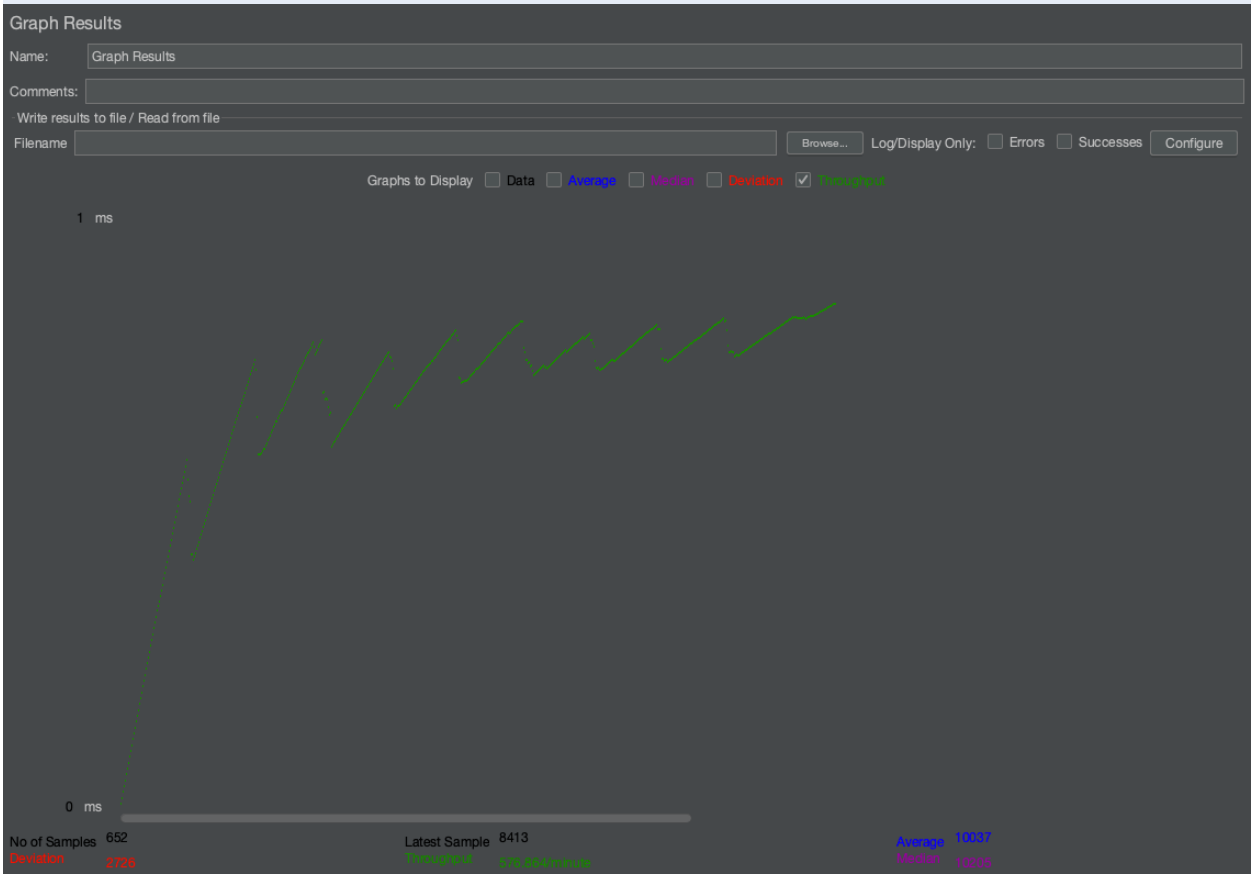




2. 10 запросов в секунду

3. 100 запросов в секунду





4. 1000 запросов секунду
   Сервер падает нагрузку не выдерживает

Проводим НТ после создания индекса. Длительность теста 60 секунд.

1. 1 запрос в секунду

2. 10 запросов в секунду

3. 100 запросов в секунду

4. 1000 запросов в секунду

Запрос добавления индекса

```
alter table users add column first_name_tsvector tsvector
alter table users add column second_name_tsvector tsvector

update users set first_name_tsvector=to_tsvector('russian', first_name)
update users set second_name_tsvector=to_tsvector('russian', second_name)
create index fs_idx on users using gin(first_name_tsvector,second_name_tsvector)
```

Explain до добавления индекса и после

До

```
explain analyze SELECT user_id, first_name, second_name, birthdate,
biography, city FROM users WHERE first_name_tsvector @@ to_tsquery('russian',
'ан:*') and second_name_tsvector @@ to_tsquery('russian', 'бе:*') order by
user_id asc
```

```
QUERY PLAN
Gather Merge  (cost=79866.80..79978.34 rows=956 width=77) (actual time=191.486..193.556 rows=1122 loops=1)
  Workers Planned: 2
  Workers Launched: 2
  ->  Sort  (cost=78866.77..78867.97 rows=478 width=77) (actual time=172.768..172.786 rows=374 loops=3)
        Sort Key: user_id
        Sort Method: quicksort  Memory: 65kB
        Worker 0:  Sort Method: quicksort  Memory: 65kB
        Worker 1:  Sort Method: quicksort  Memory: 78kB
        ->  Parallel Seq Scan on users  (cost=0.00..78845.50 rows=478 width=77) (actual time=140.461..172.625 rows=374 loops=3)
              Filter: (((first_name)::text ~~* 'ан%'::text) AND ((second_name)::text ~~* 'бе%'::text))
              Rows Removed by Filter: 329626
Planning Time: 0.425 ms
Execution Time: 193.609 ms
```

После

```
explain analyze SELECT user_id, first_name, second_name, birthdate,
biography, city FROM users WHERE first_name ilike 'ан%' and second_name ilike
'бе%' order by user_id asc
```

```
QUERY PLAN
Sort  (cost=7625.69..7630.75 rows=2023 width=77) (actual time=18.251..18.341 rows=1122 loops=1)
  Sort Key: user_id
  Sort Method: quicksort  Memory: 183kB
  ->  Bitmap Heap Scan on users  (cost=492.74..7514.61 rows=2023 width=77) (actual time=12.822..17.566 rows=1122 loops=1)
        Recheck Cond: ((first_name_tsvector @@ '''ан'':*'::tsquery) AND (second_name_tsvector @@ '''бе'':*'::tsquery))
        Heap Blocks: exact=373
        ->  Bitmap Index Scan on fs_idx  (cost=0.00..492.23 rows=2023 width=0) (actual time=12.684..12.685 rows=1122 loops=1)
              Index Cond: ((first_name_tsvector @@ '''ан'':*'::tsquery) AND (second_name_tsvector @@ '''бе'':*'::tsquery))
Planning Time: 1.127 ms
Execution Time: 18.494 ms
```

Объяснение индекса

Так как нужен поиск по тексту и одновременно по имени и фамилии, причем еще используется и постфикс, то хорошим вариантом в postgresql будет индекс gin. Так как согласно документации gin одна из рекомендаций для полнотекстового поиска. А так как нужен одновременно поиск по двум полям, то я создал составной индекс на основе двух атрибутов.