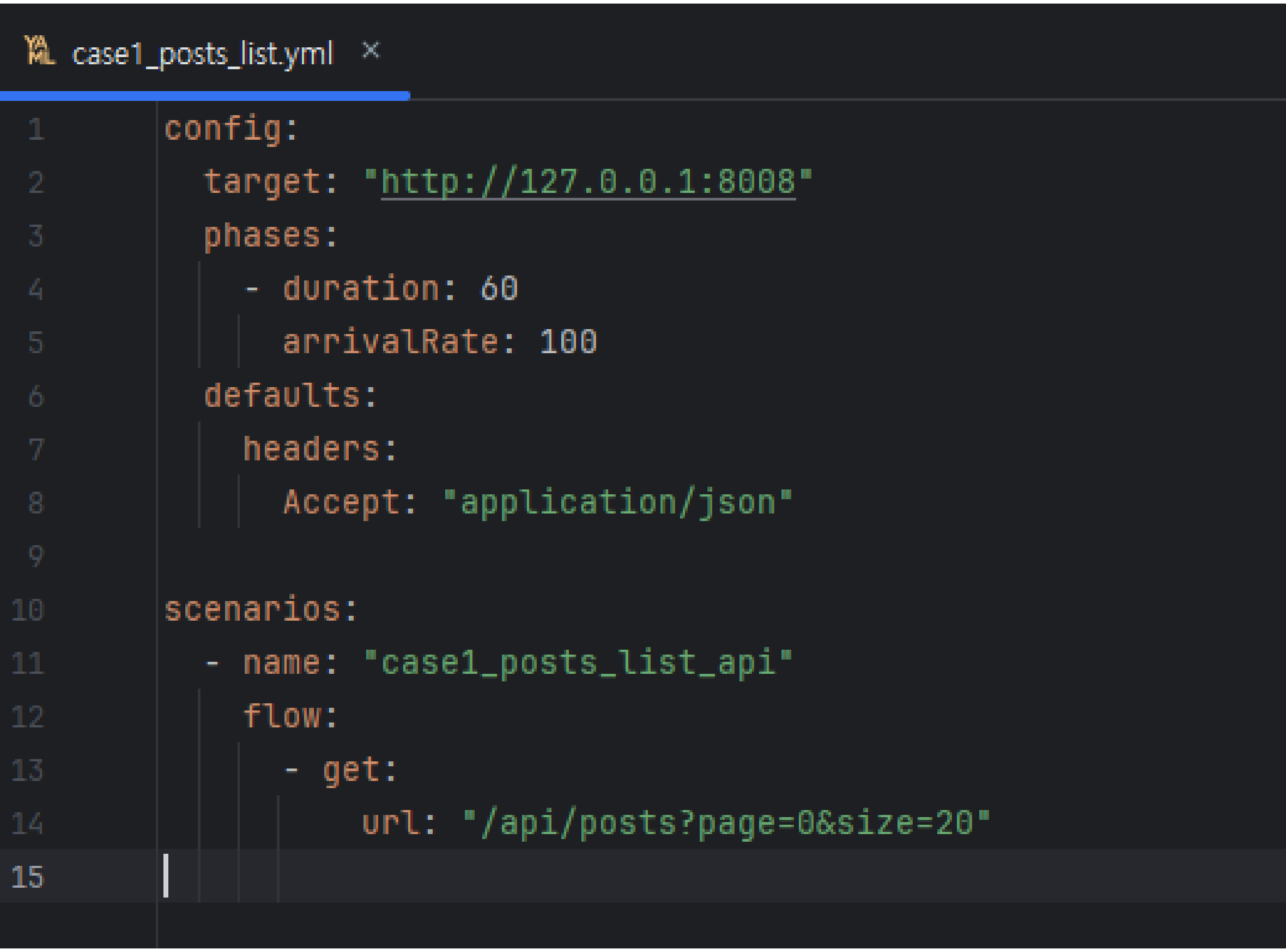
Case1 – N+1 Query Performance Evidence

Verification Evidence PDF

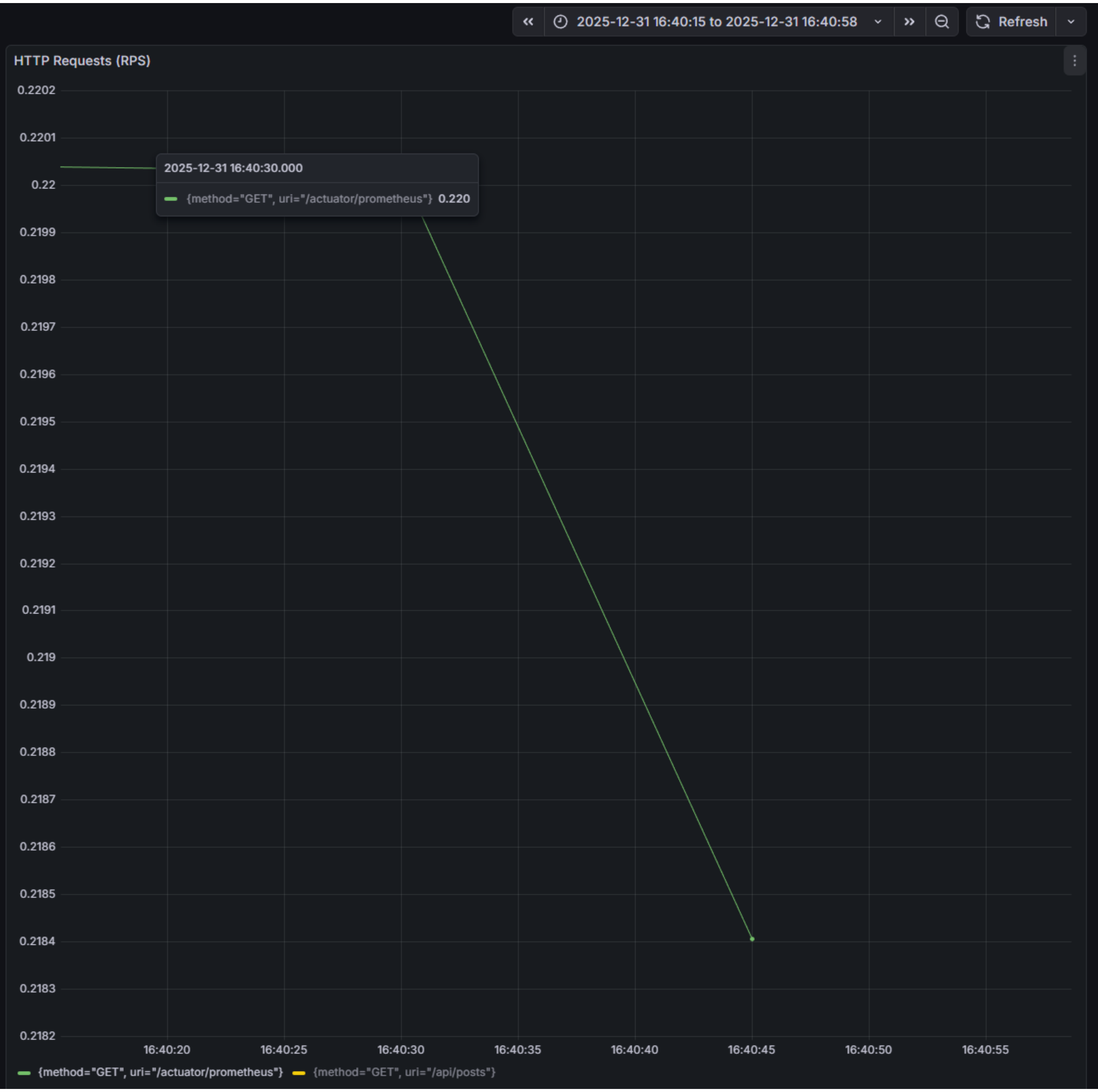
Test Conditions (Fixed)



```yaml
config:
  target: "http://127.0.0.1:8008"
  phases:
    - duration: 60
      arrivalRate: 100
  defaults:
    headers:
      Accept: "application/json"

scenarios:
  - name: "case1_posts_list_api"
    flow:
      - get:
          url: "/api/posts?page=0&size=20"
```

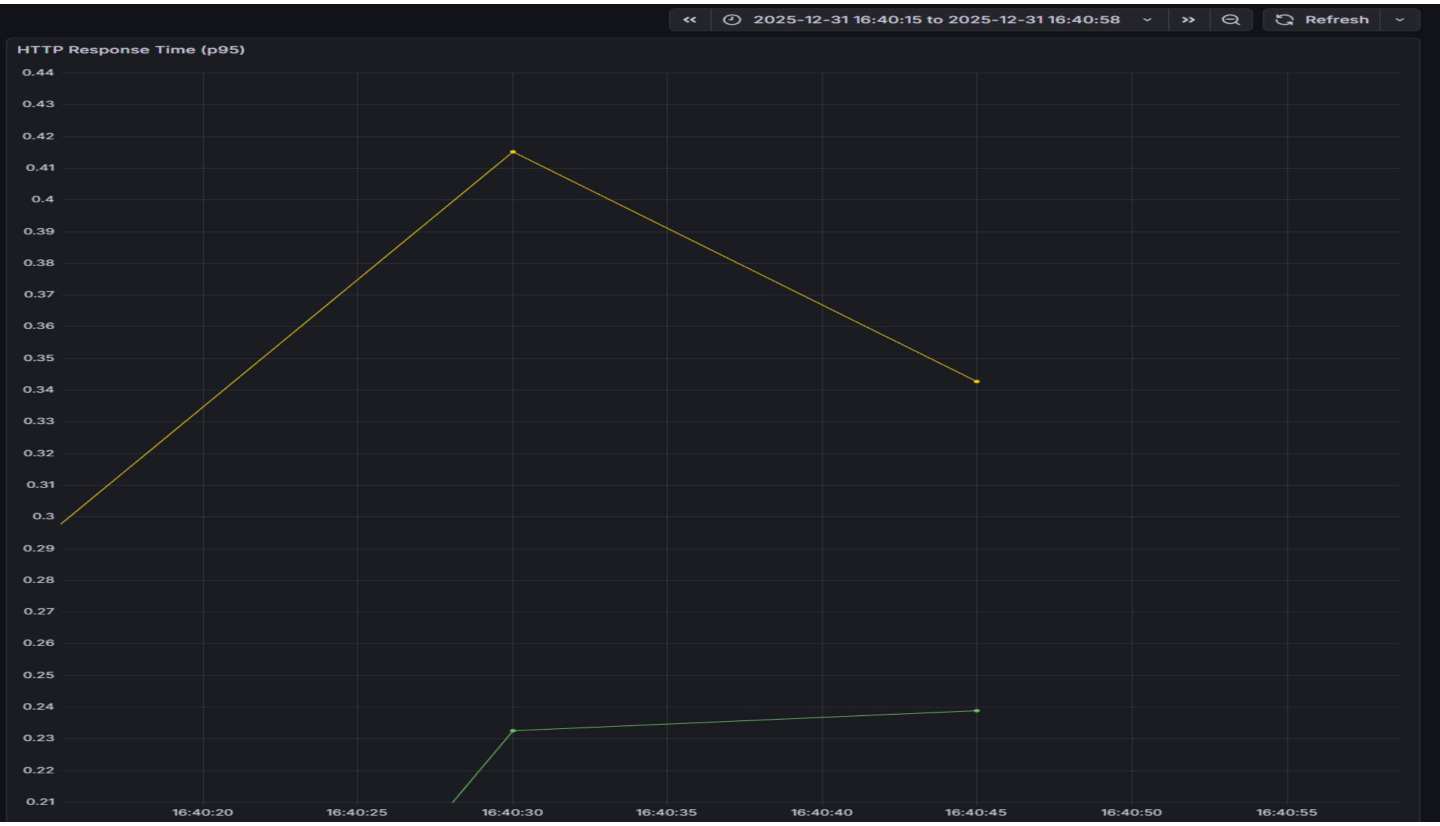
Artillery Test Scenario: case1_posts_list.yml

Grafana – Before (1/2)

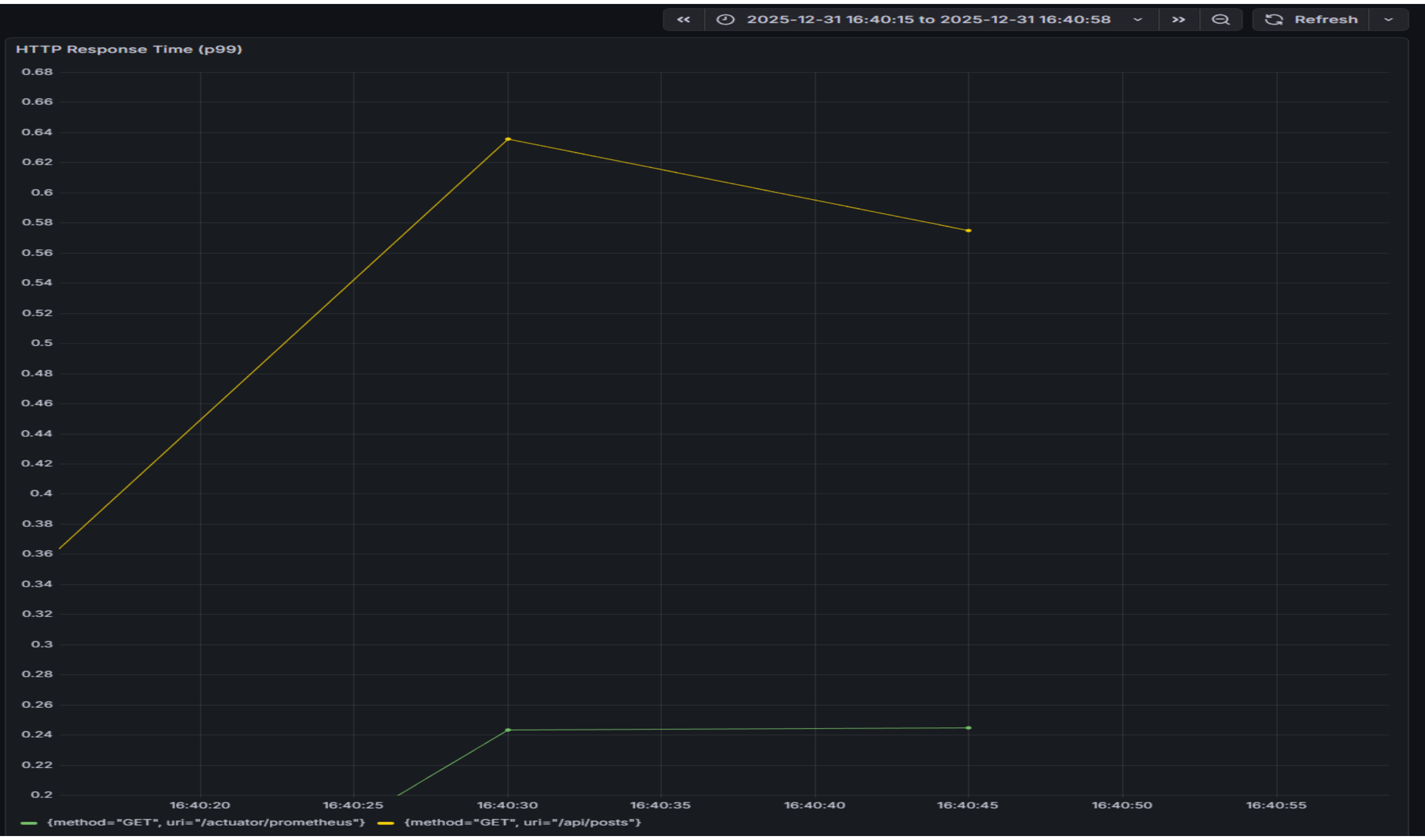


Before – HTTP Requests (RPS), Artillery 100 rps

Grafana – Before (2/2)

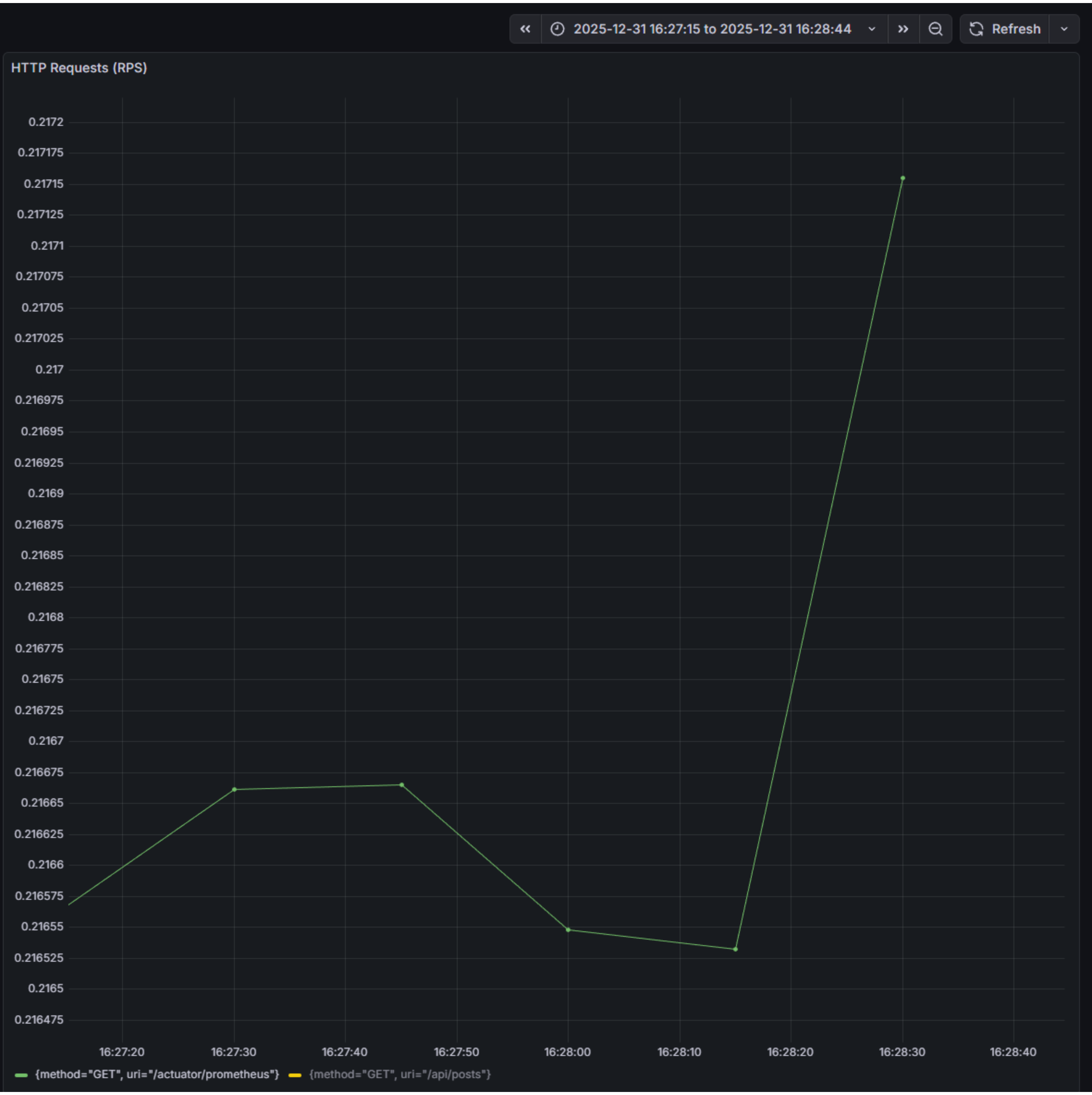


Before – HTTP Response Time p95



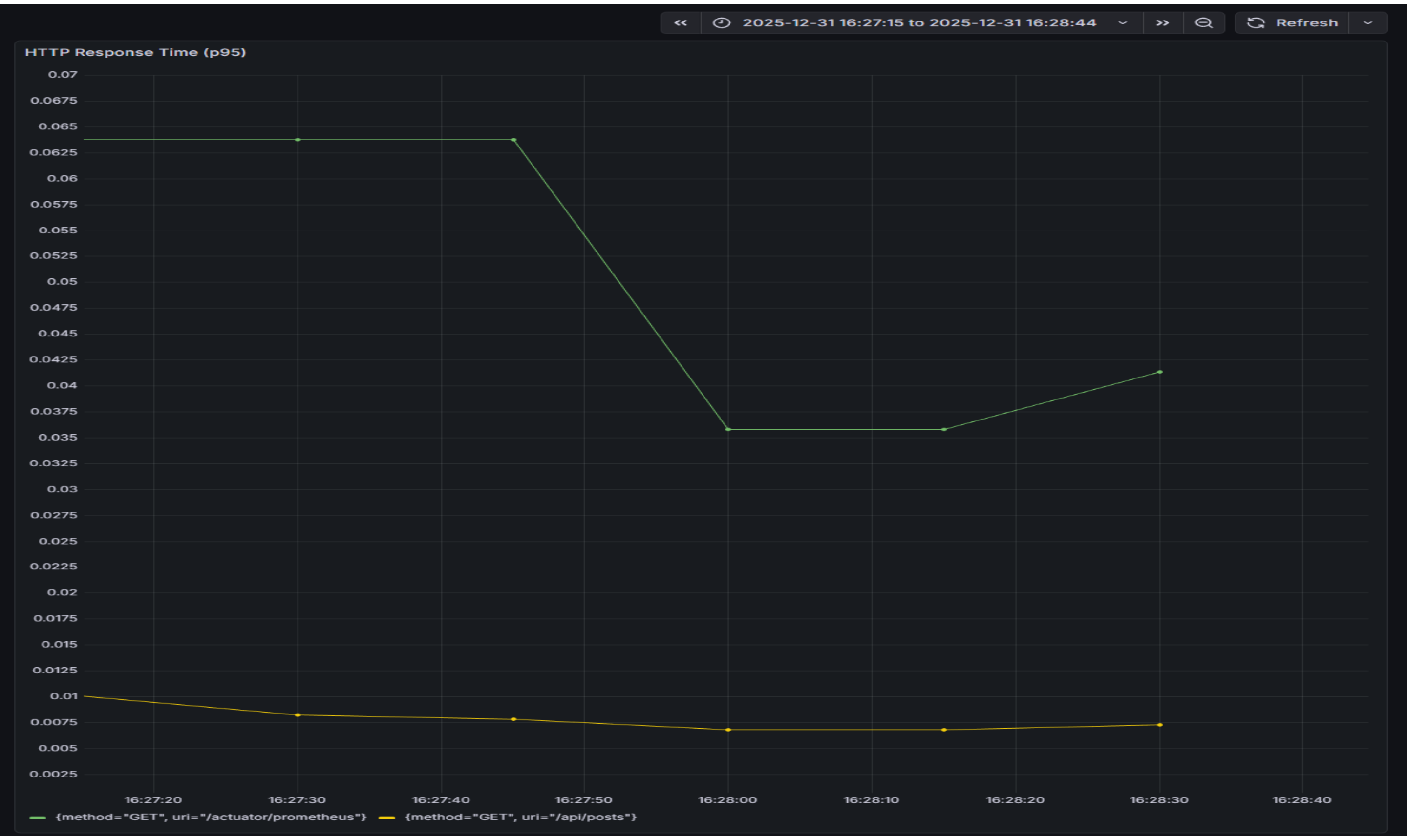
Before – HTTP Response Time p95

Grafana – After (1/2)

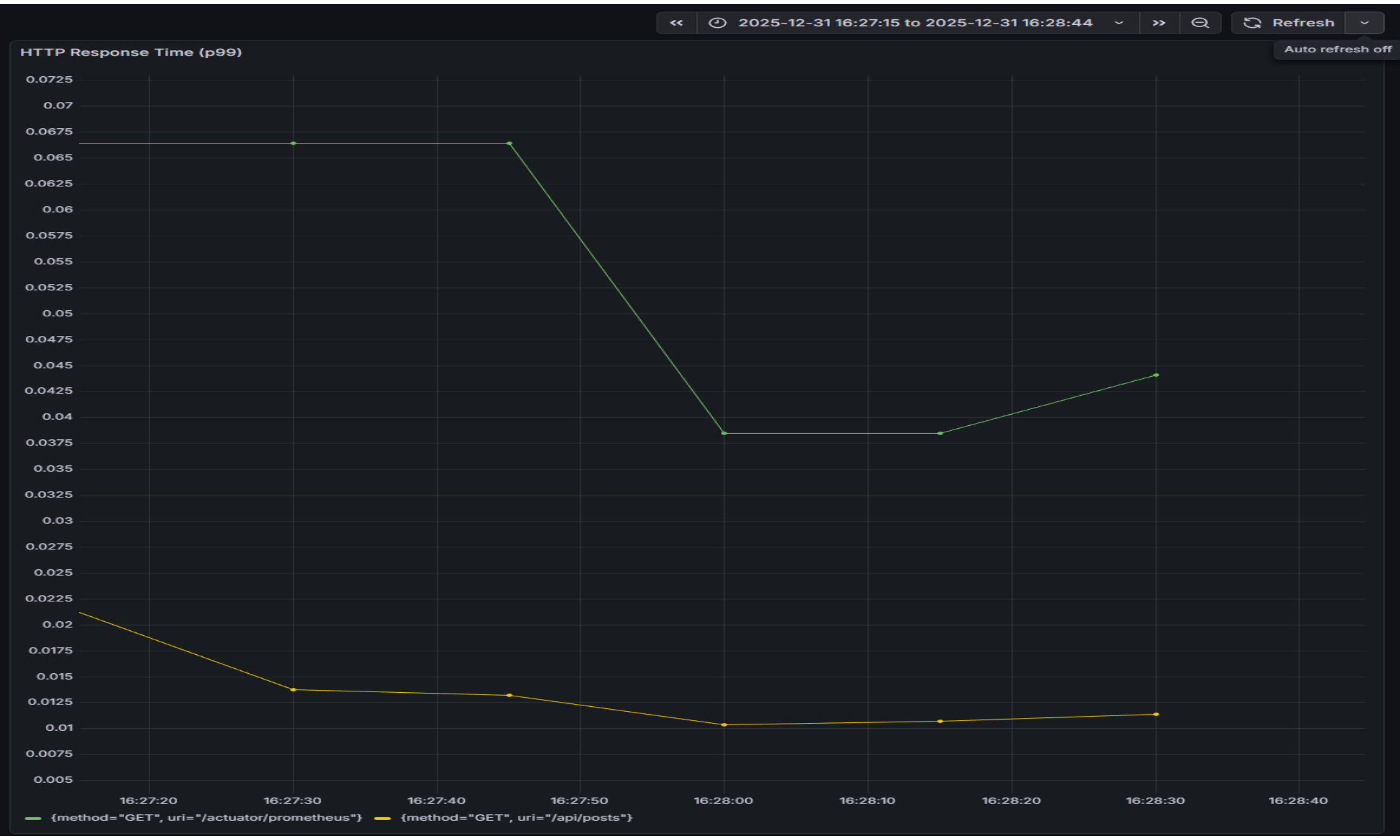


After – HTTP – HTTP Requests (RPS), Artillery 100 rps

Grafana – After (2/2)

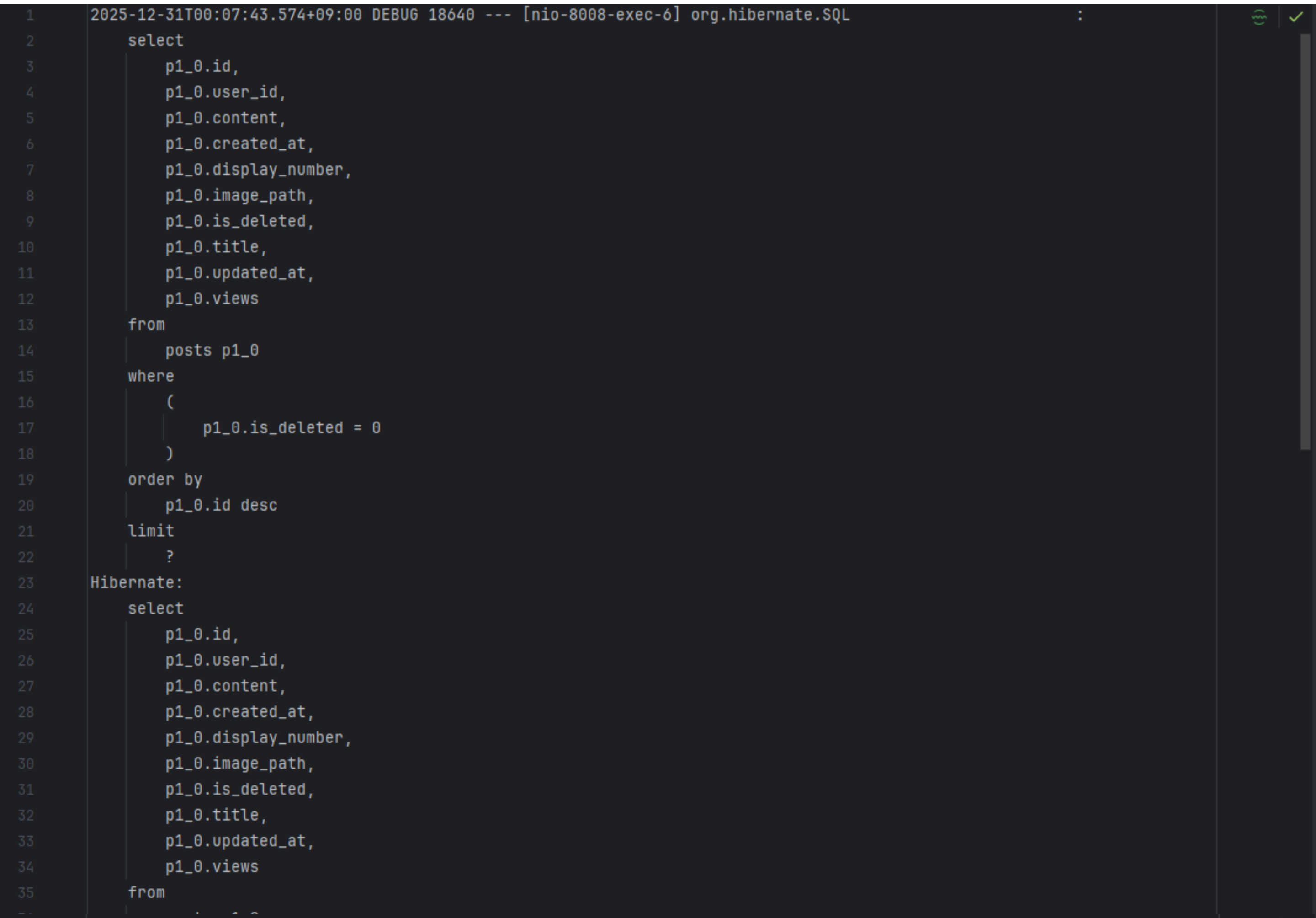


After – HTTP Response Time p95

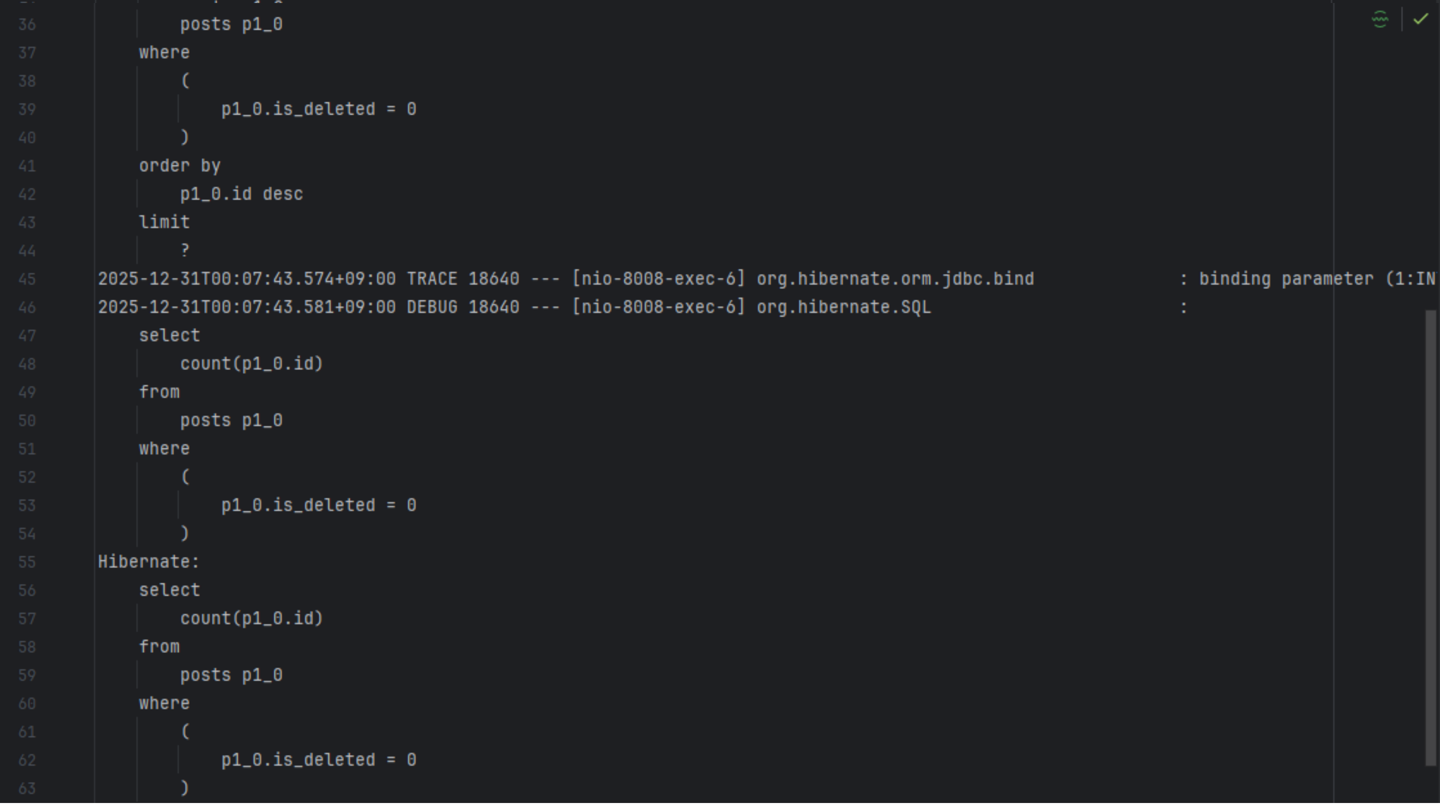


After – HTTP Response Time p99

SQL Log (Before)



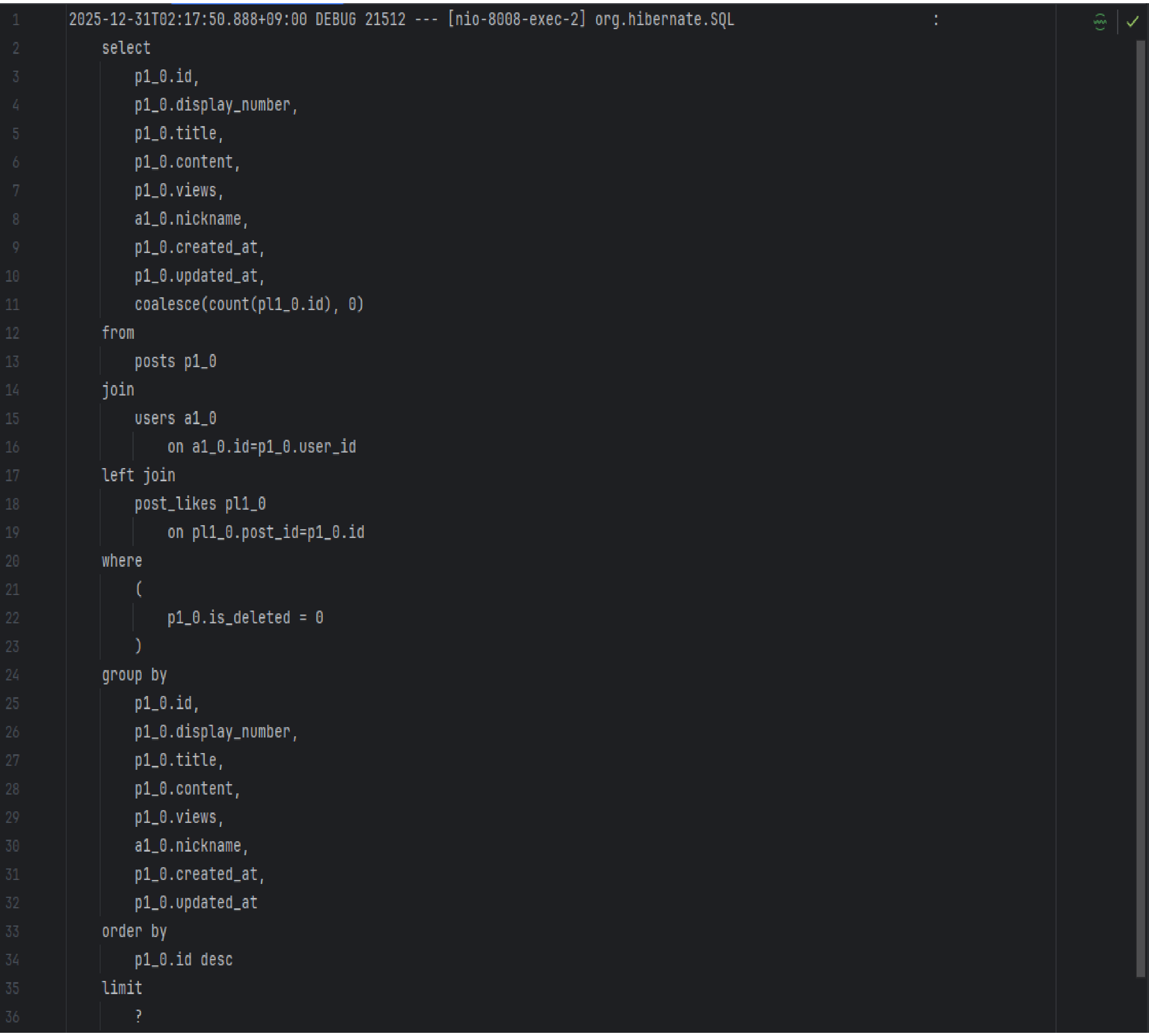
```
2025-12-31T00:07:43.574+09:00 DEBUG 18640 --- [nio-8008-exec-6] org.hibernate.SQL                        :
    select
        p1_0.id,
        p1_0.user_id,
        p1_0.content,
        p1_0.created_at,
        p1_0.display_number,
        p1_0.image_path,
        p1_0.is_deleted,
        p1_0.title,
        p1_0.updated_at,
        p1_0.views
    from
        posts p1_0
    where
        (
            p1_0.is_deleted = 0
        )
    order by
        p1_0.id desc
    limit
        ?
Hibernate:
    select
        p1_0.id,
        p1_0.user_id,
        p1_0.content,
        p1_0.created_at,
        p1_0.display_number,
        p1_0.image_path,
        p1_0.is_deleted,
        p1_0.title,
        p1_0.updated_at,
        p1_0.views
    from
```



```
        posts p1_0
    where
        (
            p1_0.is_deleted = 0
        )
    order by
        p1_0.id desc
    limit
        ?
2025-12-31T00:07:43.574+09:00 TRACE 18640 --- [nio-8008-exec-6] org.hibernate.orm.jdbc.bind              : binding parameter (1:IN
2025-12-31T00:07:43.581+09:00 DEBUG 18640 --- [nio-8008-exec-6] org.hibernate.SQL                        :
    select
        count(p1_0.id)
    from
        posts p1_0
    where
        (
            p1_0.is_deleted = 0
        )
Hibernate:
    select
        count(p1_0.id)
    from
        posts p1_0
    where
        (
            p1_0.is_deleted = 0
        )
```

Before: Evidence: SQL Log - N+1 Query Pattern

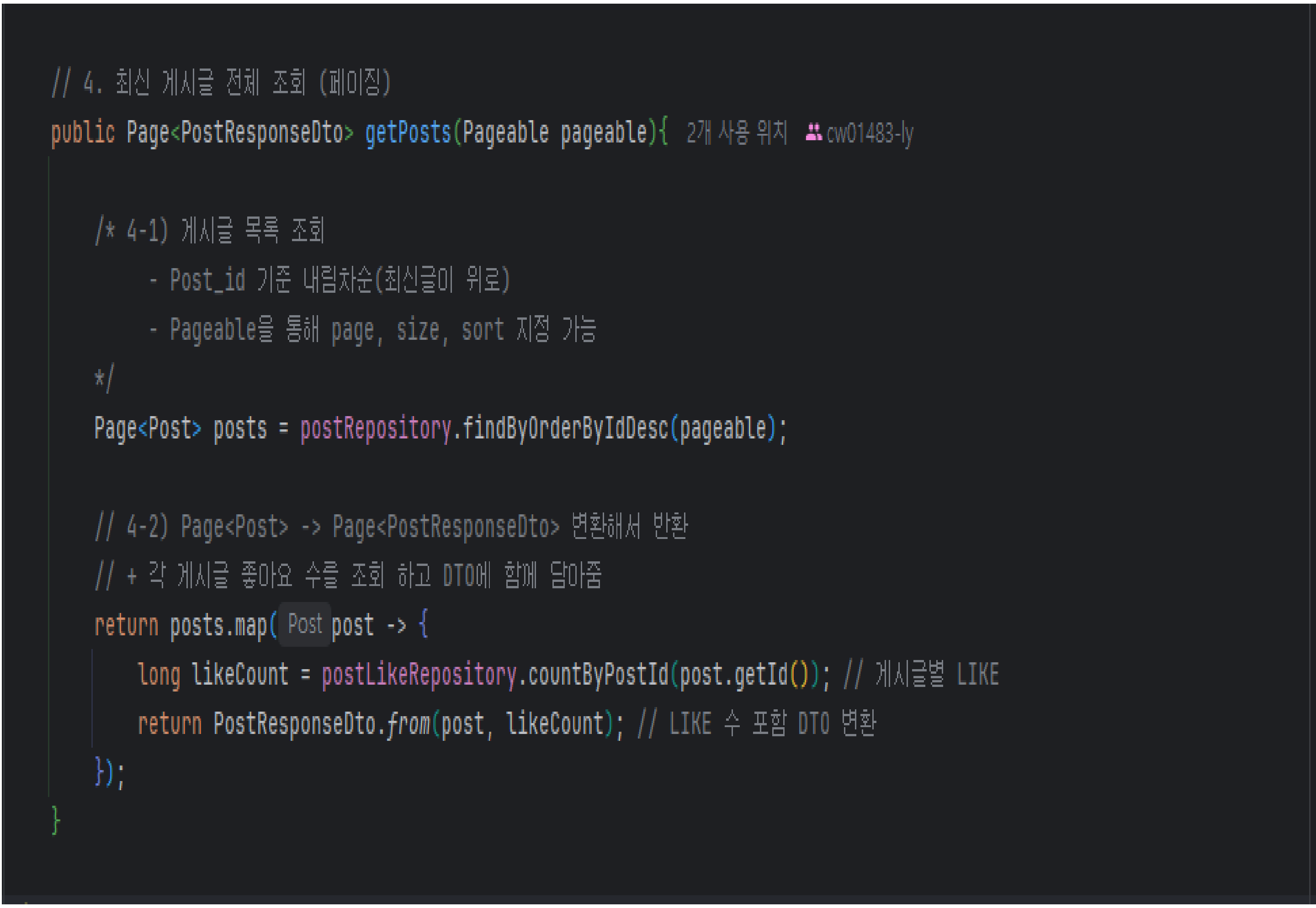
SQL Log (After)



```
2025-12-31T02:17:50.888+09:00 DEBUG 21512 --- [nio-8008-exec-2] org.hibernate.SQL                        :
    select
        p1_0.id,
        p1_0.display_number,
        p1_0.title,
        p1_0.content,
        p1_0.views,
        a1_0.nickname,
        p1_0.created_at,
        p1_0.updated_at,
        coalesce(count(pl1_0.id), 0)
    from
        posts p1_0
    join
        users a1_0
            on a1_0.id=p1_0.user_id
    left join
        post_likes pl1_0
            on pl1_0.post_id=p1_0.id
    where
        (
            p1_0.is_deleted = 0
        )
    group by
        p1_0.id,
        p1_0.display_number,
        p1_0.title,
        p1_0.content,
        p1_0.views,
        a1_0.nickname,
        p1_0.created_at,
        p1_0.updated_at
    order by
        p1_0.id desc
    limit
        ?
```

After: Evidence: SQL Log - Optimized Single Query

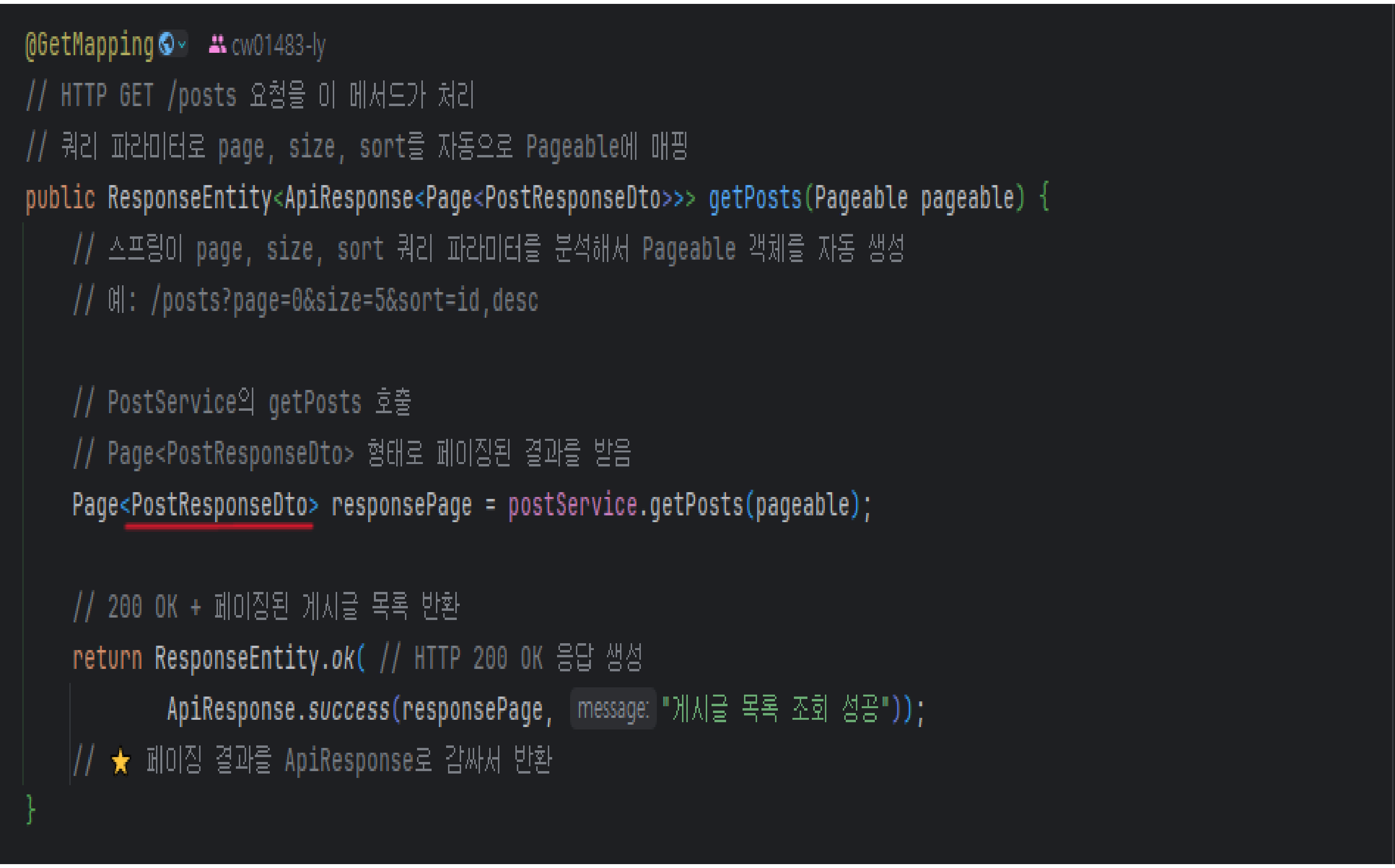
Code Change (Before) (1/2)



```
// 4. 최신 게시글 전체 조회 (페이징)
public Page<PostResponseDto> getPosts(Pageable pageable){  2개 사용 위치  cw01483-ly

    /* 4-1) 게시글 목록 조회
          - Post_id 기준 내림차순(최신글이 위로)
          - Pageable을 통해 page, size, sort 지정 가능
    */
    Page<Post> posts = postRepository.findByOrderByIdDesc(pageable);

    // 4-2) Page<Post> -> Page<PostResponseDto> 변환해서 반환
    // + 각 게시글 좋아요 수를 조회 하고 DTO에 함께 담아줌
    return posts.map( Post post -> {
        long likeCount = postLikeRepository.countByPostId(post.getId()); // 게시글별 LIKE
        return PostResponseDto.from(post, likeCount); // LIKE 수 포함 DTO 변환
    });
}
```

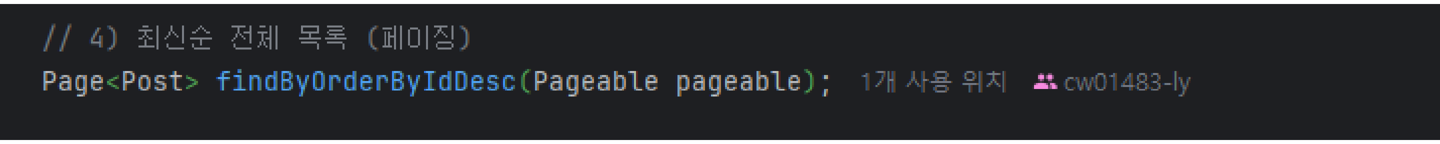
Before (Controller): Post list API calling legacy service

Code Change (Before) (2/2)



```
@GetMapping  cw01483-ly
// HTTP GET /posts 요청을 이 메서드가 처리
// 쿼리 파라미터로 page, size, sort를 자동으로 Pageable에 매핑
public ResponseEntity<ApiResponse<Page<PostResponseDto>>> getPosts(Pageable pageable) {
    // 스프링이 page, size, sort 쿼리 파라미터를 분석해서 Pageable 객체를 자동 생성
    // 예: /posts?page=0&size=5&sort=id,desc

    // PostService의 getPosts 호출
    // Page<PostResponseDto> 형태로 페이징된 결과를 받음
    Page<PostResponseDto> responsePage = postService.getPosts(pageable);

    // 200 OK + 페이징된 게시글 목록 반환
    return ResponseEntity.ok( // HTTP 200 OK 응답 생성
            ApiResponse.success(responsePage, message: "게시글 목록 조회 성공"));
    // ⭐ 페이징 결과를 ApiResponse로 감싸서 반환
}
```

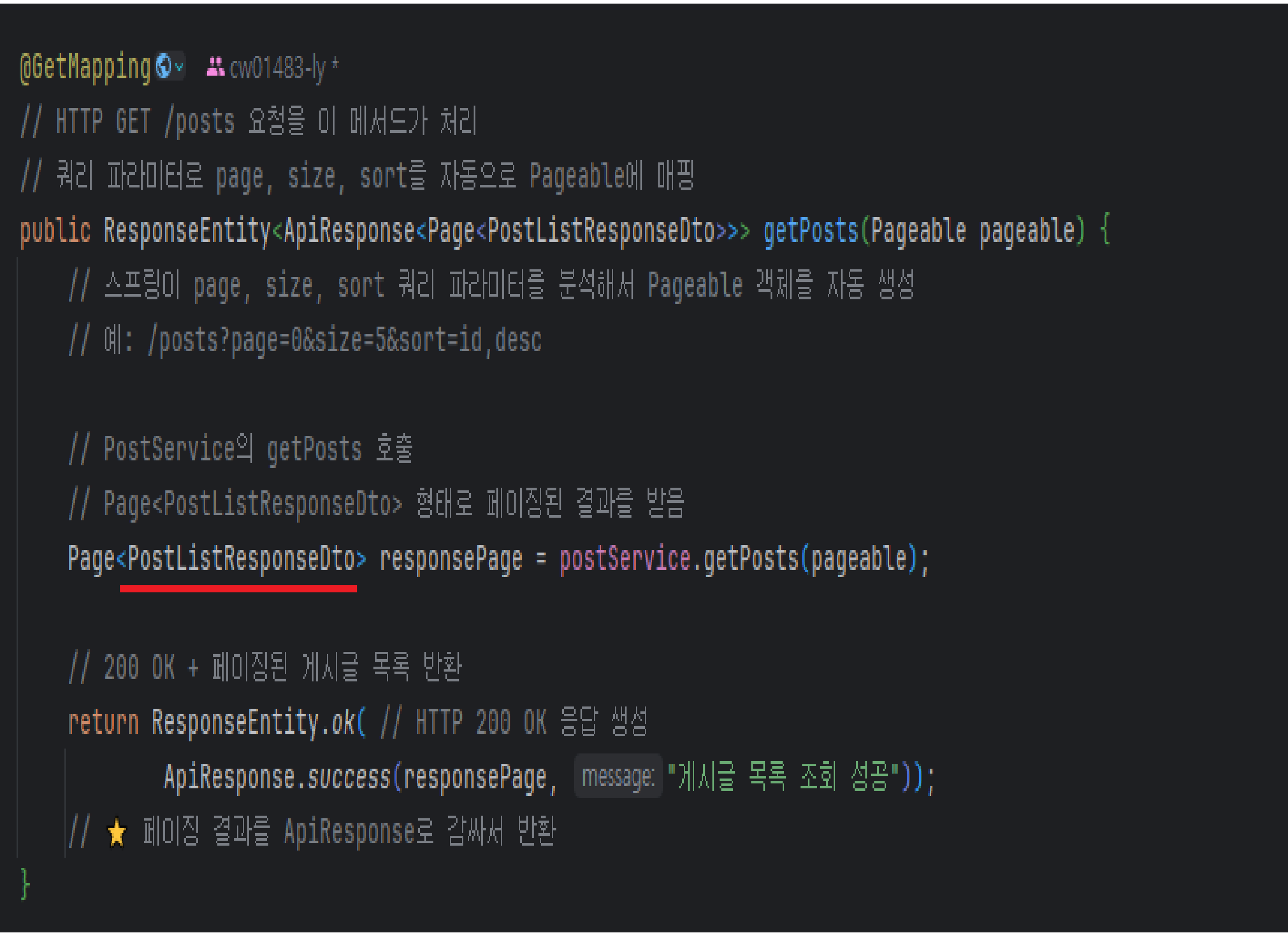
Before (Service): N+1 issue caused by per-post count query



```
// 4) 최신순 전체 목록 (페이징)
Page<Post> findByOrderByIdDesc(Pageable pageable); 1개 사용 위치  cw01483-ly
```

Before (Repository): Simple paging query (No aggregation support)

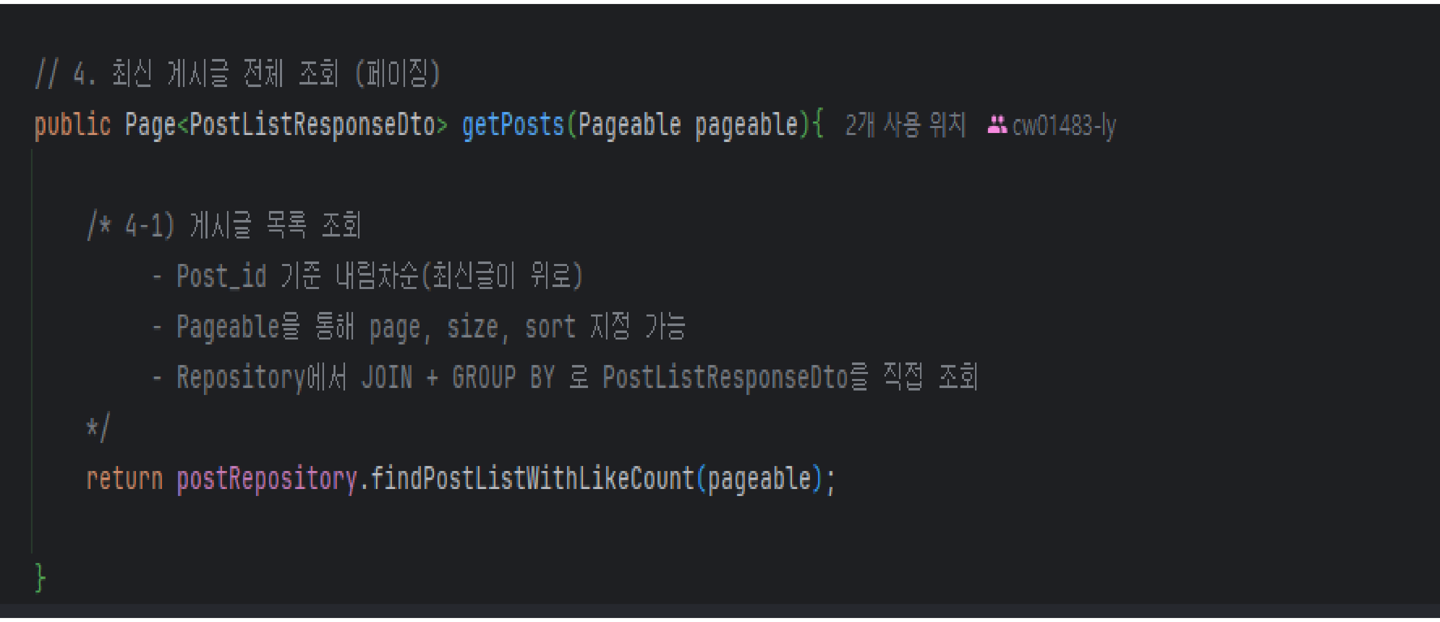
Code Change (After) (1/2)



```java
@GetMapping  cw01483-ly *
// HTTP GET /posts 요청을 이 메서드가 처리
// 쿼리 파라미터로 page, size, sort를 자동으로 Pageable에 매핑
public ResponseEntity<ApiResponse<Page<PostListResponseDto>>> getPosts(Pageable pageable) {
    // 스프링이 page, size, sort 쿼리 파라미터를 분석해서 Pageable 객체를 자동 생성
    // 예: /posts?page=0&size=5&sort=id,desc

    // PostService의 getPosts 호출
    // Page<PostListResponseDto> 형태로 페이징된 결과를 받음
    Page<PostListResponseDto> responsePage = postService.getPosts(pageable);

    // 200 OK + 페이징된 게시글 목록 반환
    return ResponseEntity.ok( // HTTP 200 OK 응답 생성
            ApiResponse.success(responsePage, message: "게시글 목록 조회 성공"));
    // ⭐ 페이징 결과를 ApiResponse로 감싸서 반환
}
```

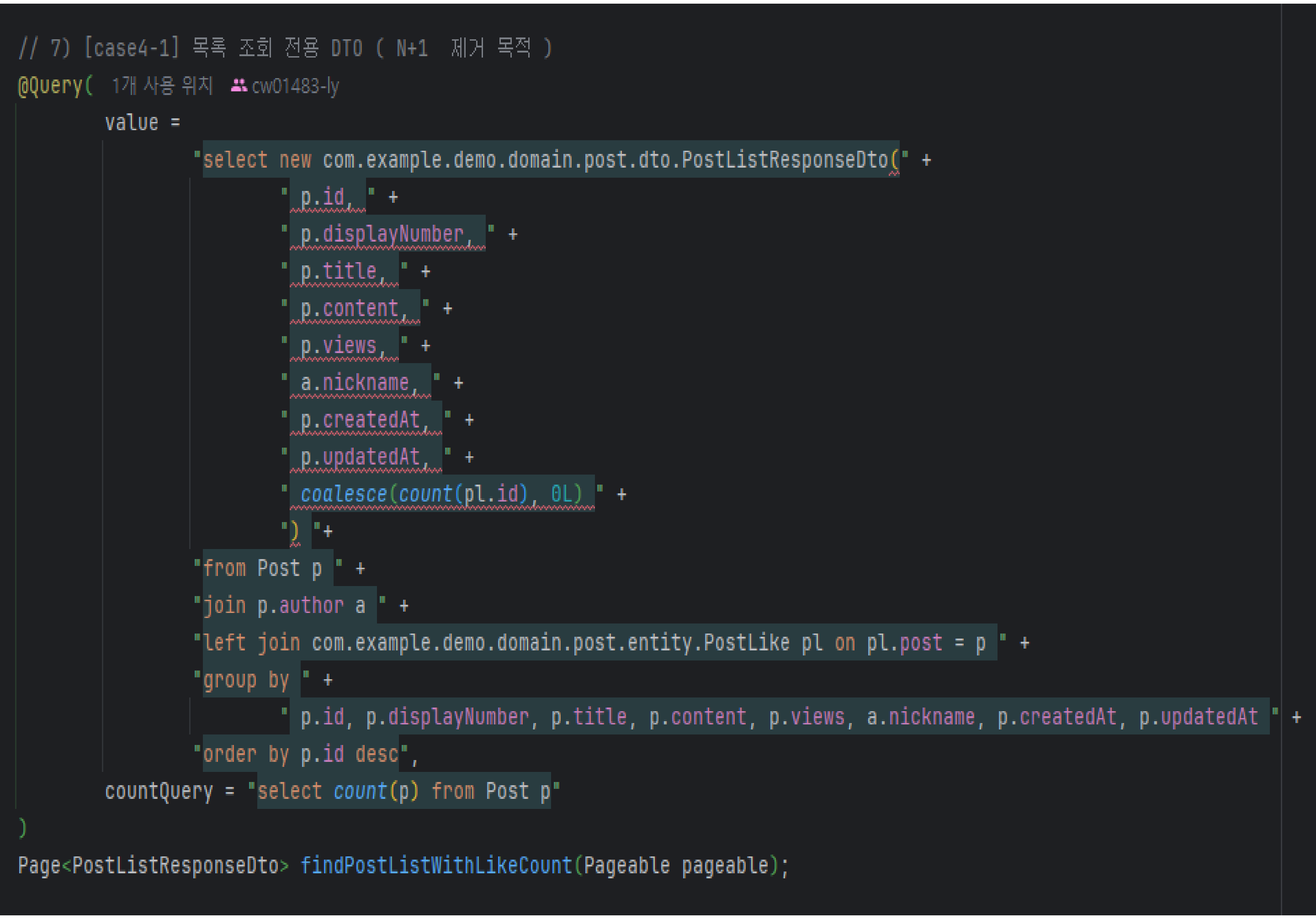
After (Controller): Refactored API using optimized DTO

Code Change (After) (2/2)



```
// 4. 최신 게시글 전체 조회 (페이징)
public Page<PostListResponseDto> getPosts(Pageable pageable){  2개 사용 위치  cw01483-ly

    /* 4-1) 게시글 목록 조회
         - Post_id 기준 내림차순(최신글이 위로)
         - Pageable을 통해 page, size, sort 지정 가능
         - Repository에서 JOIN + GROUP BY 로 PostListResponseDto를 직접 조회
    */
    return postRepository.findPostListWithLikeCount(pageable);

}
```

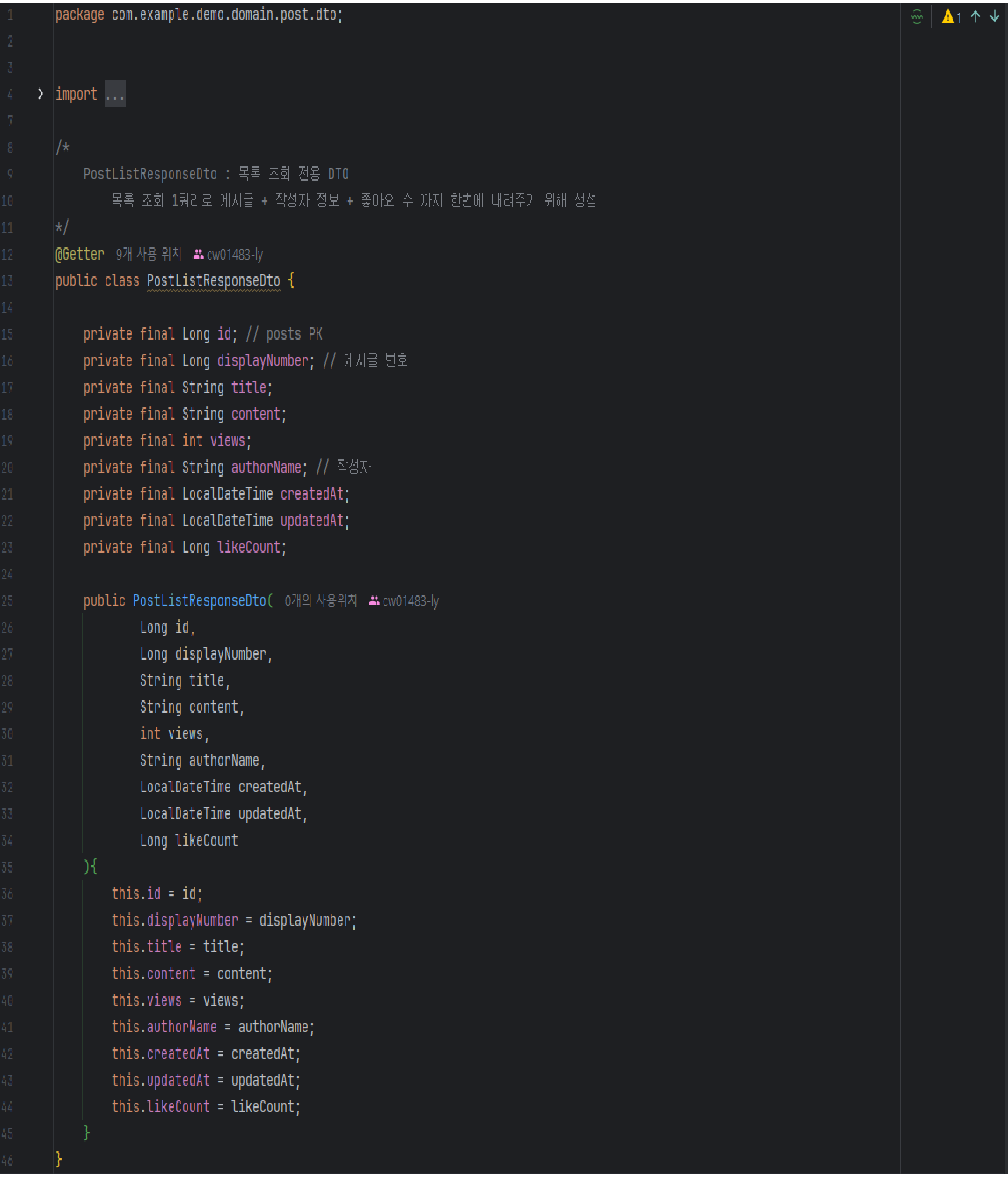
After (Service): Performance-optimized single service call



```
// 7) [case4-1] 목록 조회 전용 DTO ( N+1  제거 목적 )
@Query(  1개 사용 위치  cw01483-ly
        value =
                "select new com.example.demo.domain.post.dto.PostListResponseDto(" +
                        " p.id, " +
                        " p.displayNumber, " +
                        " p.title, " +
                        " p.content, " +
                        " p.views, " +
                        " a.nickname, " +
                        " p.createdAt, " +
                        " p.updatedAt, " +
                        " coalesce(count(pl.id), 0L) " +
                        ") "+
                "from Post p " +
                "join p.author a " +
                "left join com.example.demo.domain.post.entity.PostLike pl on pl.post = p " +
                "group by " +
                        " p.id, p.displayNumber, p.title, p.content, p.views, a.nickname, p.createdAt, p.updatedAt " +
                "order by p.id desc",
        countQuery = "select count(p) from Post p"
)
Page<PostListResponseDto> findPostListWithLikeCount(Pageable pageable);
```

After (Repository): Optimized JOIN & GROUP BY query

DTO Evidence (After)



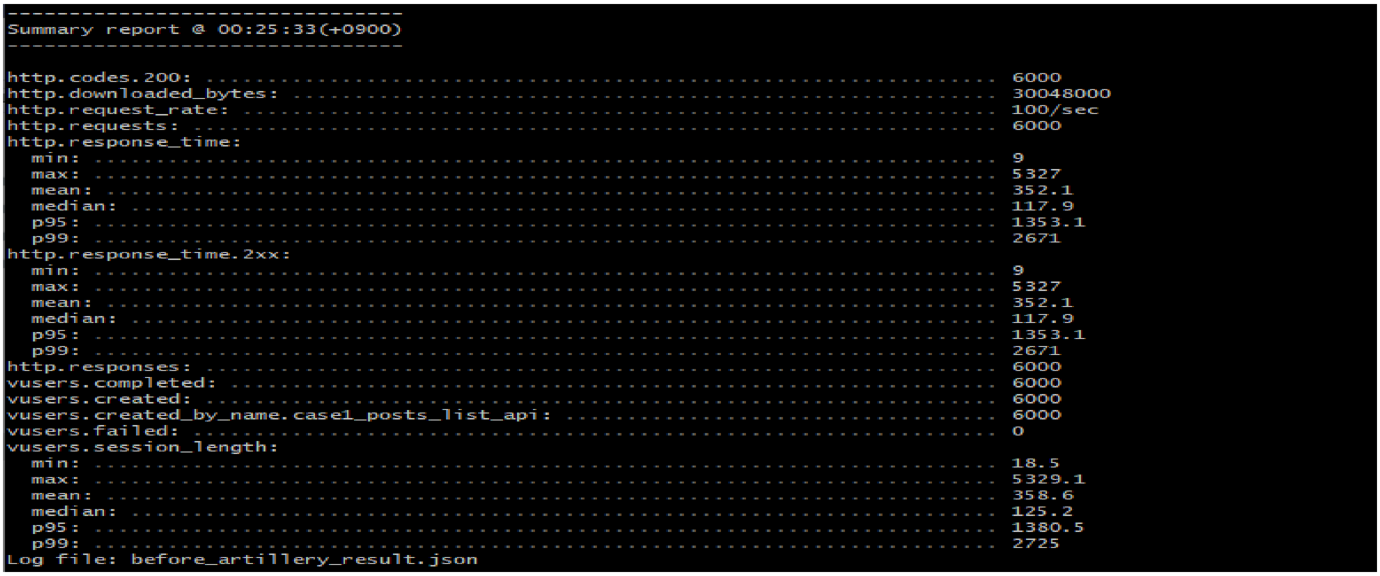
```java
package com.example.demo.domain.post.dto;


import ...

/*
    PostListResponseDto : 목록 조회 전용 DTO
        목록 조회 1쿼리로 게시글 + 작성자 정보 + 좋아요 수 까지 한번에 내려주기 위해 생성
*/
@Getter
public class PostListResponseDto {

    private final Long id; // posts PK
    private final Long displayNumber; // 게시글 번호
    private final String title;
    private final String content;
    private final int views;
    private final String authorName; // 작성자
    private final LocalDateTime createdAt;
    private final LocalDateTime updatedAt;
    private final Long likeCount;

    public PostListResponseDto(
            Long id,
            Long displayNumber,
            String title,
            String content,
            int views,
            String authorName,
            LocalDateTime createdAt,
            LocalDateTime updatedAt,
            Long likeCount
    ){
        this.id = id;
        this.displayNumber = displayNumber;
        this.title = title;
        this.content = content;
        this.views = views;
        this.authorName = authorName;
        this.createdAt = createdAt;
        this.updatedAt = updatedAt;
        this.likeCount = likeCount;
    }
}
```

After (DTO): Specialized DTO for query projection

Artillery Result Comparison

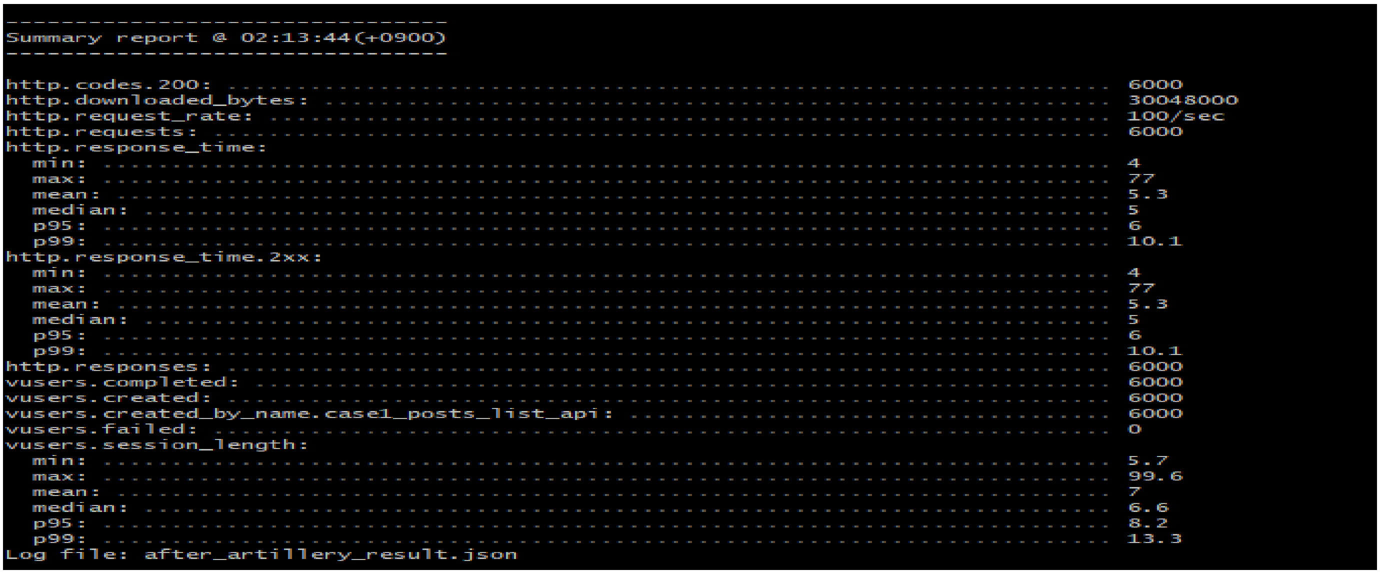
Metric	Before	After	Improvement
p95	1353 ms	6 ms	99.6% ↓
p99	2671 ms	10 ms	99.6% ↓
Mean	352 ms	5 ms	98.5% ↓
Max	5327 ms	77 ms	98.6% ↓

Table - Artillery latency summary (http.response_time)



```
--------------------------------
Summary report @ 00:25:33(+0900)
--------------------------------

http.codes.200: ............................................................... 6000
http.downloaded_bytes: ........................................................ 30048000
http.request_rate: ............................................................ 100/sec
http.requests: ................................................................ 6000
http.response_time:
  min: ........................................................................ 9
  max: ........................................................................ 5327
  mean: ....................................................................... 352.1
  median: ..................................................................... 117.9
  p95: ........................................................................ 1353.1
  p99: ........................................................................ 2671
http.response_time.2xx:
  min: ........................................................................ 9
  max: ........................................................................ 5327
  mean: ....................................................................... 352.1
  median: ..................................................................... 117.9
  p95: ........................................................................ 1353.1
  p99: ........................................................................ 2671
http.responses: ............................................................... 6000
vusers.completed: ............................................................. 6000
vusers.created: ............................................................... 6000
vusers.created_by_name.case1_posts_list_api: .................................. 6000
vusers.failed: ................................................................ 0
vusers.session_length:
  min: ........................................................................ 18.5
  max: ........................................................................ 5329.1
  mean: ....................................................................... 358.6
  median: ..................................................................... 125.2
  p95: ........................................................................ 1380.5
  p99: ........................................................................ 2725
Log file: before_artillery_result.json
```

Before – Artillery summary



```
--------------------------------
Summary report @ 02:13:44(+0900)
--------------------------------

http.codes.200: ............................................................... 6000
http.downloaded_bytes: ........................................................ 30048000
http.request_rate: ............................................................ 100/sec
http.requests: ................................................................ 6000
http.response_time:
  min: ........................................................................ 4
  max: ........................................................................ 77
  mean: ....................................................................... 5.3
  median: ..................................................................... 5
  p95: ........................................................................ 6
  p99: ........................................................................ 10.1
http.response_time.2xx:
  min: ........................................................................ 4
  max: ........................................................................ 77
  mean: ....................................................................... 5.3
  median: ..................................................................... 5
  p95: ........................................................................ 6
  p99: ........................................................................ 10.1
http.responses: ............................................................... 6000
vusers.completed: ............................................................. 6000
vusers.created: ............................................................... 6000
vusers.created_by_name.case1_posts_list_api: .................................. 6000
vusers.failed: ................................................................ 0
vusers.session_length:
  min: ........................................................................ 5.7
  max: ........................................................................ 99.6
  mean: ....................................................................... 7
  median: ..................................................................... 6.6
  p95: ........................................................................ 8.2
  p99: ........................................................................ 13.3
Log file: after_artillery_result.json
```

After – Artillery summary

Conclusion

- **Final Result**: **223x Performance Gain**(1,181ms → 5ms)
- **Core Solution: JOIN & DTO Projection** (Resolved N+1 Issue)
- **Validation**: **SQL Log / Grafana / Artillery**

Final Status: **Optimized & Verified**