C6-Go-Blog

Advanced Programming C

Anggota :

1. Diah Afia Safitri - 2016750420
2. Rahfi Alyendra Gibran - 2106705764
3. Maulana Bayu Risma Santoso Sari - 2106750401
4. Bryan Tjandra - 2106706634
5. Alvin Xavier Rakha Wardhana - 2106750300

Link

GitLab Backend Auth & Blog

- ristek.link/c6-go-blog-backend-auth
- ristek.link/c6-go-blog-backend-blog

GitHub Frontend

- ristek.link/c6-go-blog-frontend

SonarQube Blog

- ristek.link/c6-go-blog-sonarqube-blog

SonarQube Auth

- ristek.link/c6-go-blog-sonarqube-auth

Demo

- https://goblog.not4saken.com/home

Pembagian Kerja

Auth : Rahfi
Post : Bayu, Bryan
Comment : Alvin, Diah

Go-Blog

Sebuah platform yang memungkinkan pengguna untuk membuat, mempublikasikan, dan membagikan postingan atau tulisan blog mereka

Posting blog

Comment blog

Notification

Search blog

Design Pattern

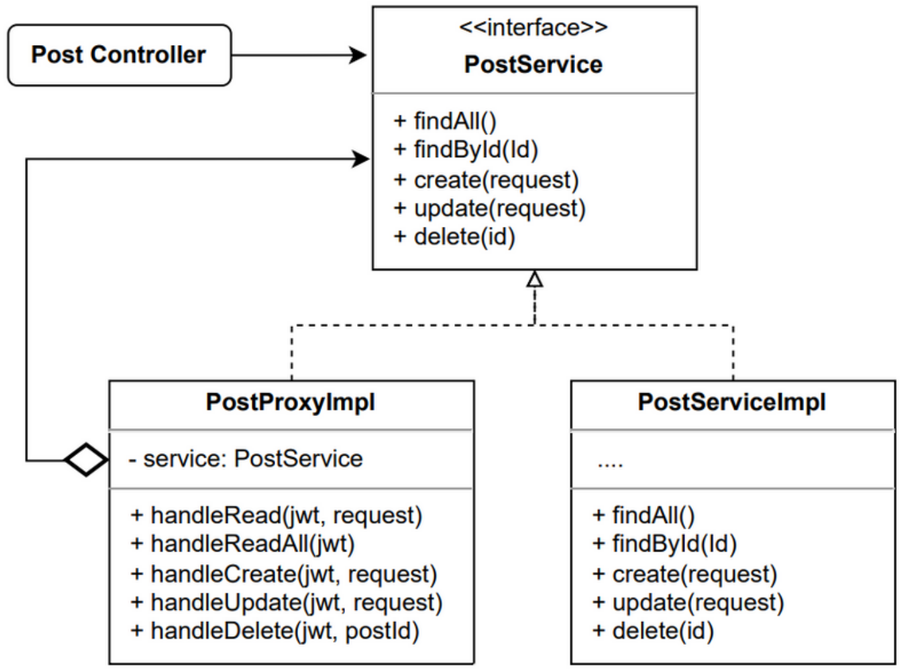
Proxy Pattern

Proxy Pattern memungkinkan Anda melakukan sesuatu sebelum atau setelah permintaan diteruskan ke objek asli.

- Proxy -> PostProxyImpl
- Service -> PostServiceImpl
- Service Interface -> PostService
- Client -> PostController

Tujuan:

- Clear exception handling
- Easier to maintain
- Better readibility
- Separation of concern



```java
public class PostController {

    private final PostProxy postProxyService;

    @Operation(summary = "Get all posts that exist")
    @GetMapping("/get_all_post")
    public ResponseEntity<List<Post>> getAllPost(@Re
        List<Post> response;
```

Client

```java
public interface PostService {
    List<Post> findAll();
    GetPostResponse findById(Integer id);
    CompletableFuture<Post> create(CreatePostRequest request);
    CompletableFuture<Post> update(UpdatePostRequest request);
    void delete(Integer id);
    List<Post> findByTitleIgnoreCase(SearchPostRequest request);
}
```

Service Interface

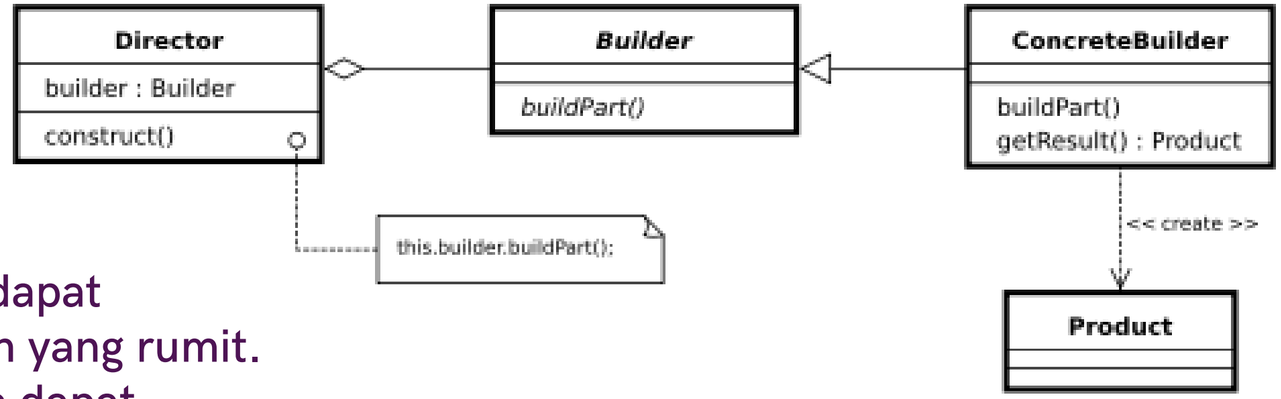
```java
public class PostProxyImpl extends ProxyServiceBase implements PostProxy {

    private final PostRepository postRepository;
    private final PostService postService;

    @Override
    public Post handleCreate(String jwt, CreatePostRequest request) {
        authenticateJWT(jwt);
        Post result = null;
```

Proxy

Service

```java
public class PostServiceImpl implements PostService {
    private final PostRepository postRepository;
    private final CommentRepository commentRepository;
    private final UserRepository userRepository;
    private final ExecutorService executor = Executors.newFixedThreadPool

    @Override
    public List<Post> findAll() {
        var sort = Sort.by(Sort.Direction.DESC, "timeCreated");
```

Builder Pattern



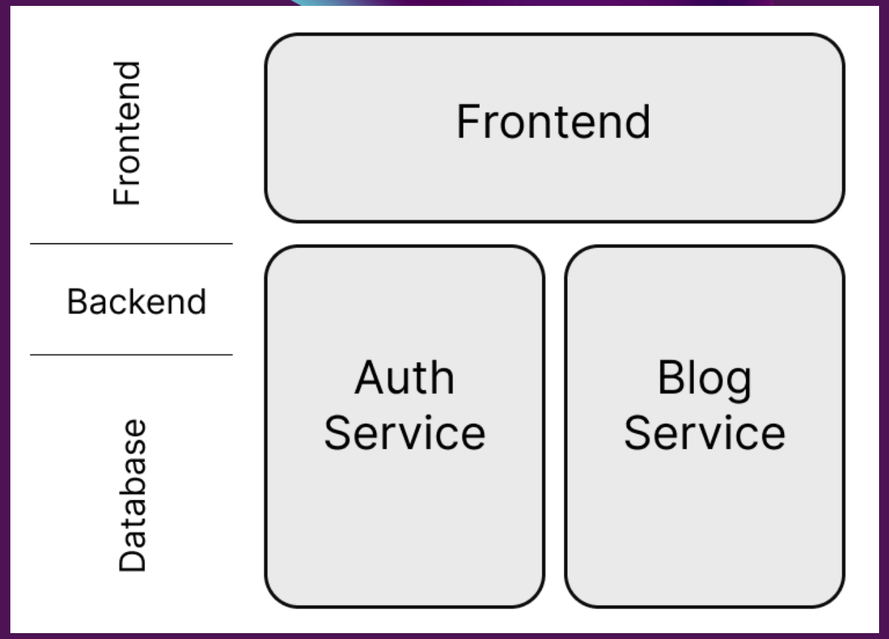
Dalam beberapa kasus, pembuatan objek dapat melibatkan banyak parameter atau langkah yang rumit. Dengan menggunakan Builder pattern, kita dapat memisahkan proses pembuatan objek kompleks dari klien yang menggunakannya.

```
Comment comment = Comment.builder()
        .content(request.getContent())
        .timeCreated(new Date())
        .postId(request.getPostId())
        .creator(user)
        .build();
```

```
var post = Post.builder()
        .creator(user)
        .title(request.getTitle())
        .content(request.getContent())
        .likes(0)
        .timeCreated(new Date())
        .build();
```

```
var user = User.builder()
        .active(true)
        .username(request.getUsername())
        .fullname(request.getFullname())
        .email(request.getEmail())
        .gender(Gender.valueOf(request.getGender().toUpperCase())
        .password(passwordEncoder.encode(request.getPassword()))
        .birthdate(request.getBirthdate())
        .build();
```

```
var notification = Notification.builder()
        .timeCreated(new Date())
        .commentContent(request.getContent())
        .postId(request.getPostId())
        .creator(user)
        .subscriberId(request.getSubscriberId())
        .isNotified(false)
        .build();
```



Microservices Component:

- Auth
- Blogs

Frontend : NextJS
Backend : SpringBoot
Database : PostgreSQL
Reverse Proxy : Caddy

Microservices Components

Auth (Registrasi dan Login)

Microservices auth digunakan untuk menghandle autentikasi dan autorisasi. Hal ini penting karena auth memegang informasi rahasia seperti kredensial pengguna (username, password, token, dll) yang tidak boleh diakses atau dimodifikasi oleh lebih dari satu instance pada saat yang sama.

Microservices Blogs

Post

Create, Read, Update, Delete blog post

Comment

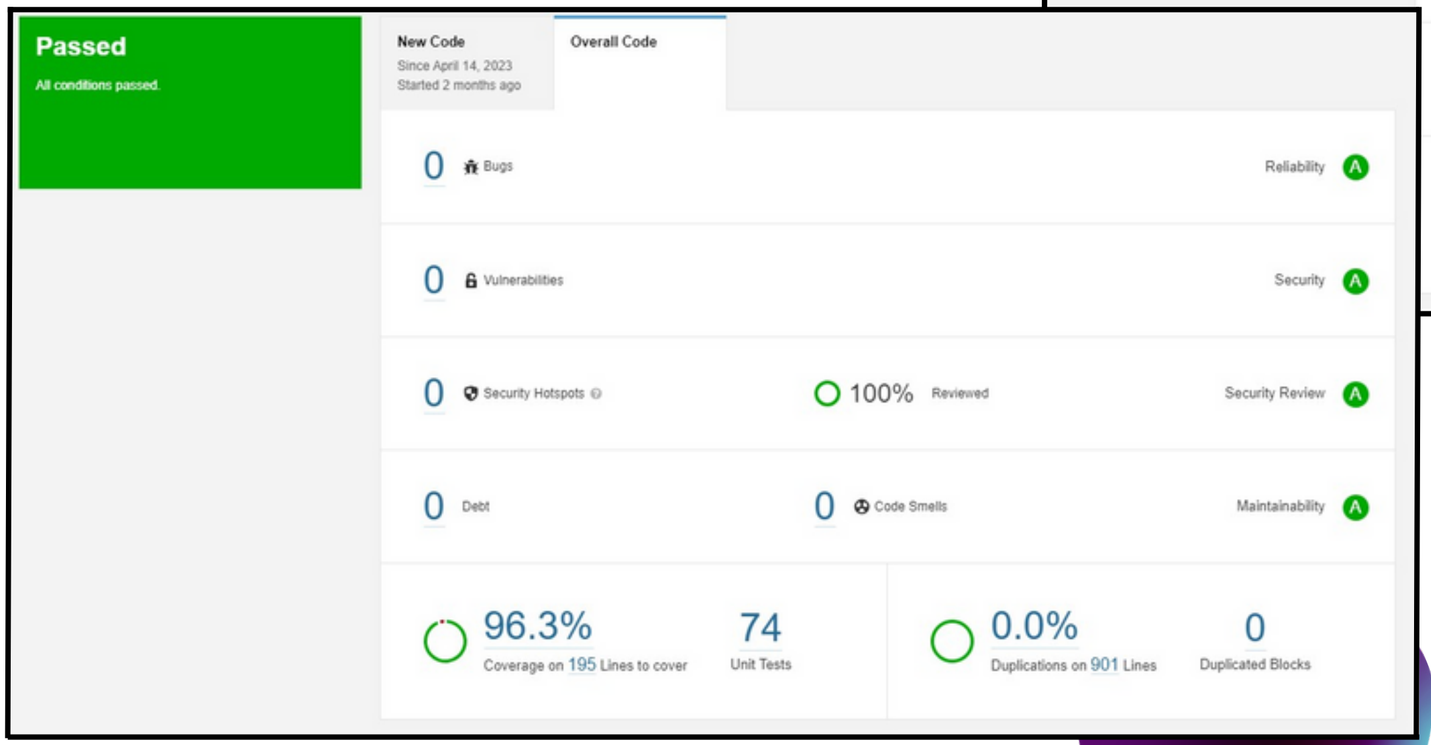
Create, Read, Delete blog comment

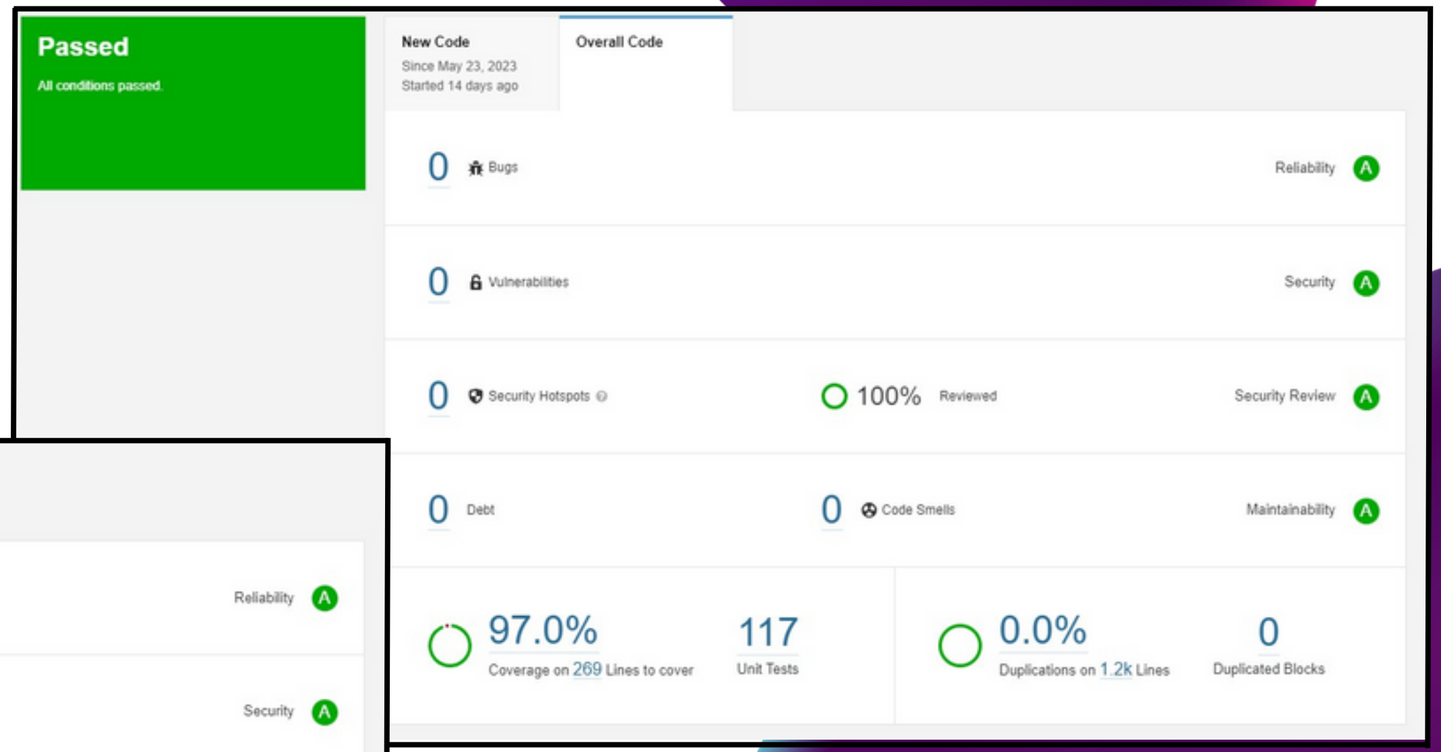
Notifications

Notify pengguna jika ada penambahan komen dengan metode pull HTTP Request setiap sekian detik

Clean Code

Auth

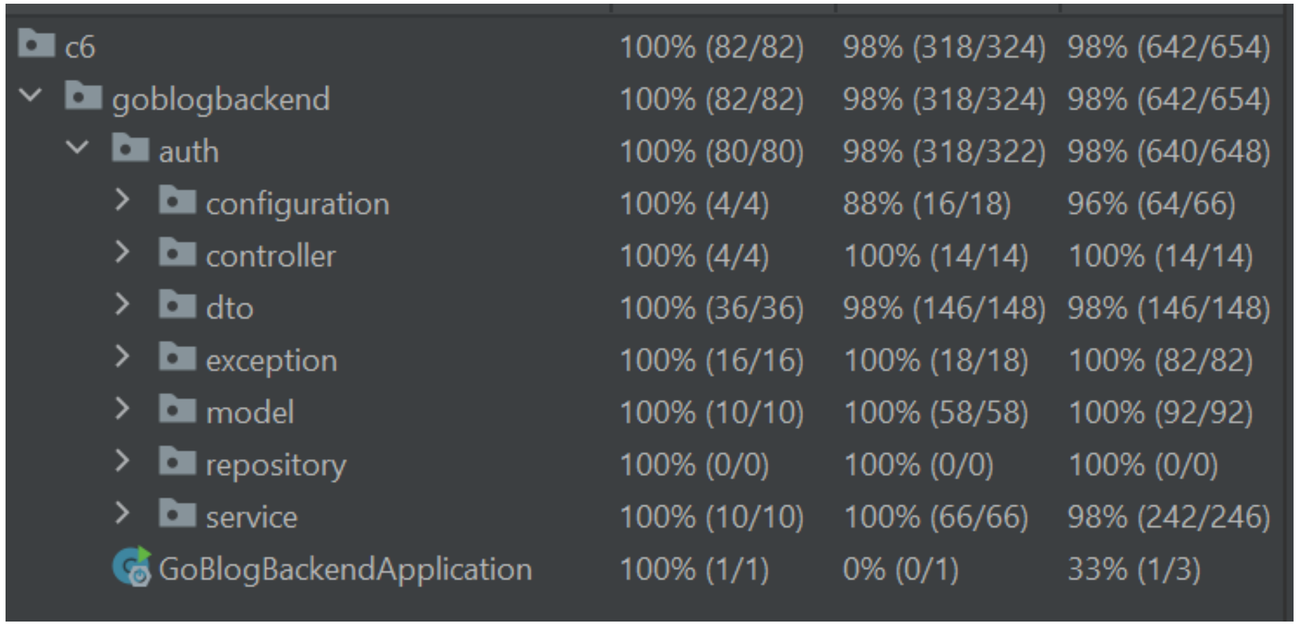




Blog

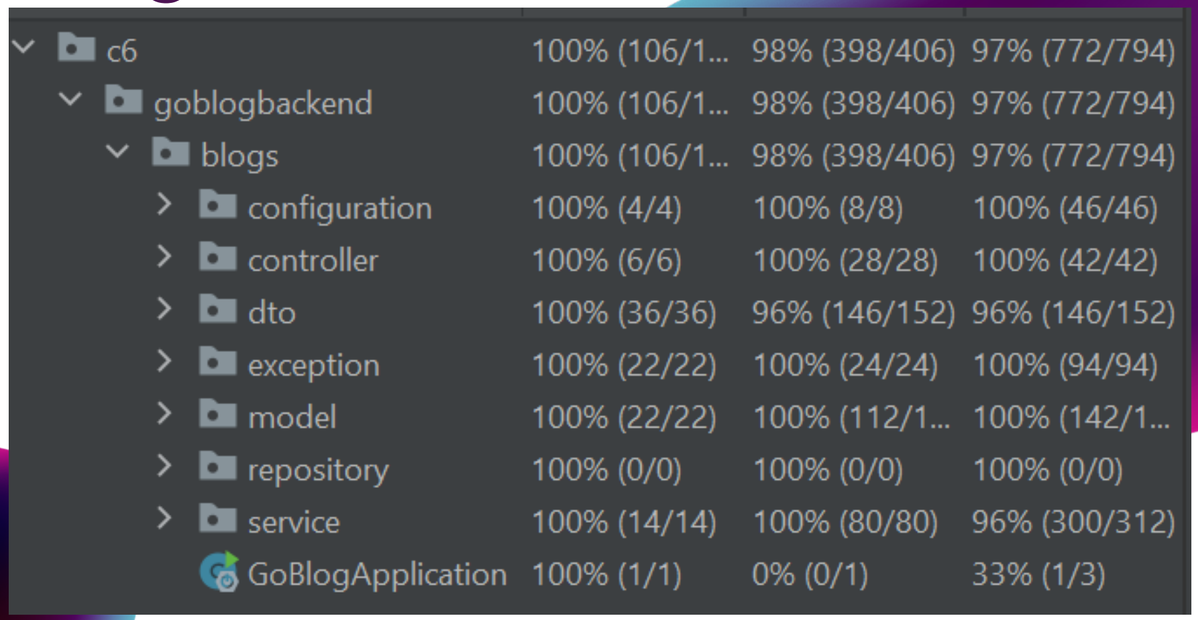
Test Coverage

Auth



c6	100% (82/82)	98% (318/324)	98% (642/654)
goblogbackend	100% (82/82)	98% (318/324)	98% (642/654)
auth	100% (80/80)	98% (318/322)	98% (640/648)
configuration	100% (4/4)	88% (16/18)	96% (64/66)
controller	100% (4/4)	100% (14/14)	100% (14/14)
dto	100% (36/36)	98% (146/148)	98% (146/148)
exception	100% (16/16)	100% (18/18)	100% (82/82)
model	100% (10/10)	100% (58/58)	100% (92/92)
repository	100% (0/0)	100% (0/0)	100% (0/0)
service	100% (10/10)	100% (66/66)	98% (242/246)
GoBlogBackendApplication	100% (1/1)	0% (0/1)	33% (1/3)

Blog

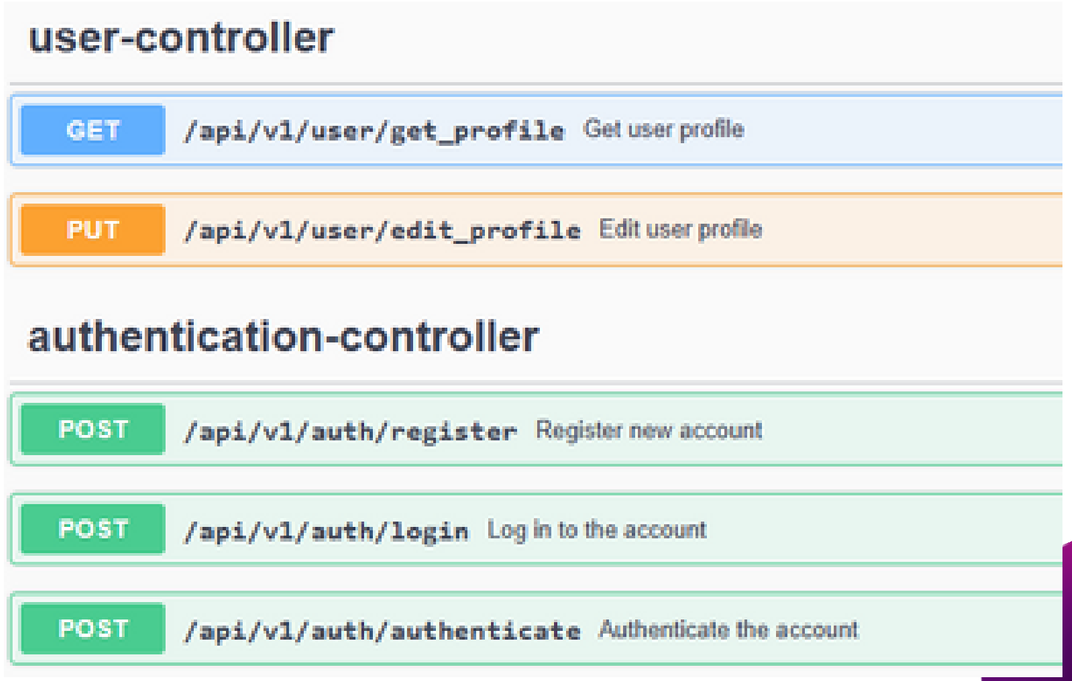


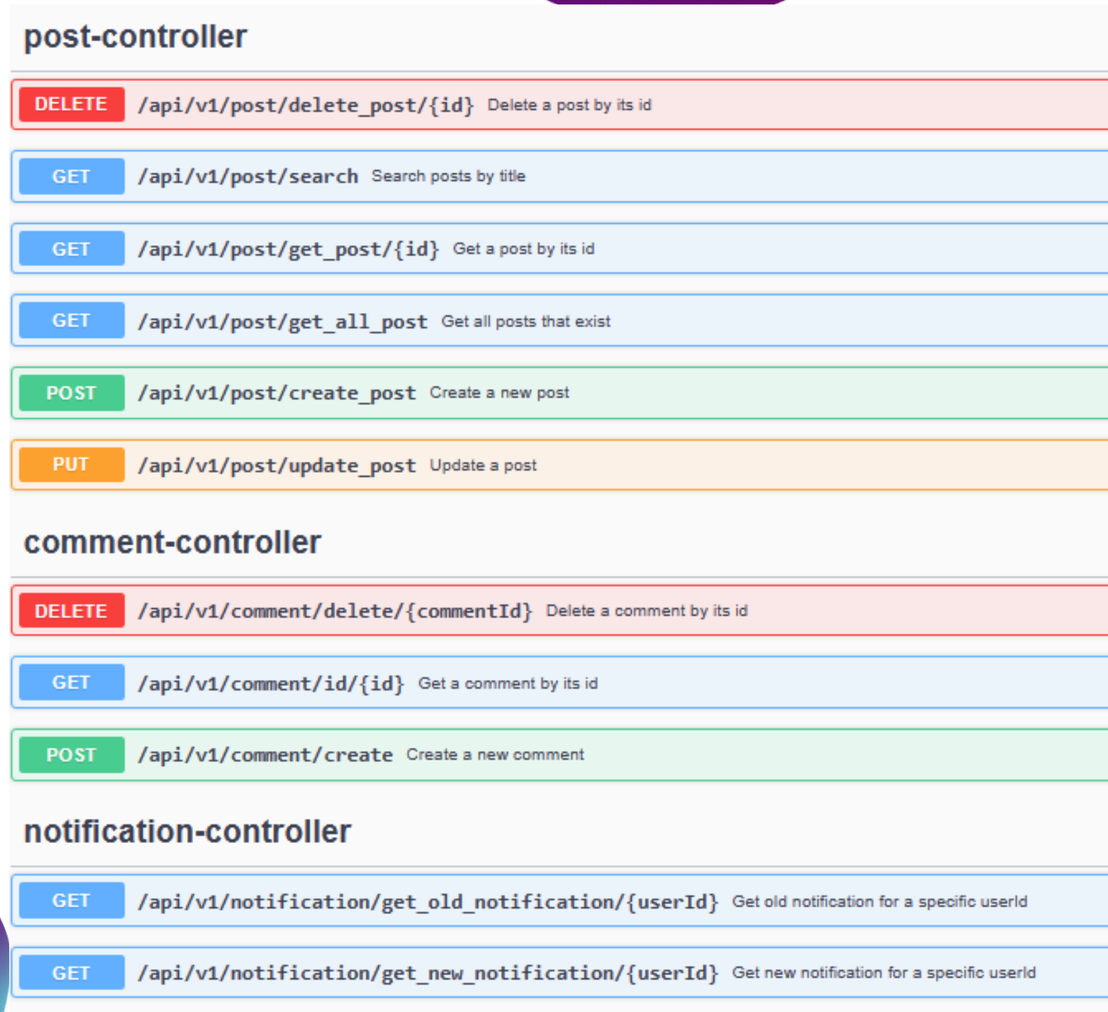
c6	100% (106/1...	98% (398/406)	97% (772/794)
goblogbackend	100% (106/1...	98% (398/406)	97% (772/794)
blogs	100% (106/1...	98% (398/406)	97% (772/794)
configuration	100% (4/4)	100% (8/8)	100% (46/46)
controller	100% (6/6)	100% (28/28)	100% (42/42)
dto	100% (36/36)	96% (146/152)	96% (146/152)
exception	100% (22/22)	100% (24/24)	100% (94/94)
model	100% (22/22)	100% (112/1...	100% (142/1...
repository	100% (0/0)	100% (0/0)	100% (0/0)
service	100% (14/14)	100% (80/80)	96% (300/312)
GoBlogApplication	100% (1/1)	0% (0/1)	33% (1/3)

API Docs

Kami menggunakan Swagger yang sudah disesuaikan dengan spesifikasi OpenApi untuk mendokumentasikan endpoint-endpoint yang telah kami buat untuk setiap servicenya.

Dapat diakses di:
https://goblog-auth.not4saken.com/docs/swagger-ui/index.html
https://goblog-blog.not4saken.com/docs/swagger-ui/index.html





Concurrency & Asynchronous

Kami mengimplementasikan concurency pada saat mengedit profile, membuat post, mengupdate post, dan mengambil notifikasi dari database.

```java
@Override
public synchronized EditProfileResponse editUserProfile(EditProfileRequest requestBody, User user){
    user.setUsername(requestBody.getUsername());
    user.setFullname(requestBody.getFullname());
    userRepository.saveAndFlush(user);
```

```java
private final ExecutorService executor = Executors.newFixedThreadPool(Runtime.getRuntime().availableProcessors());
```

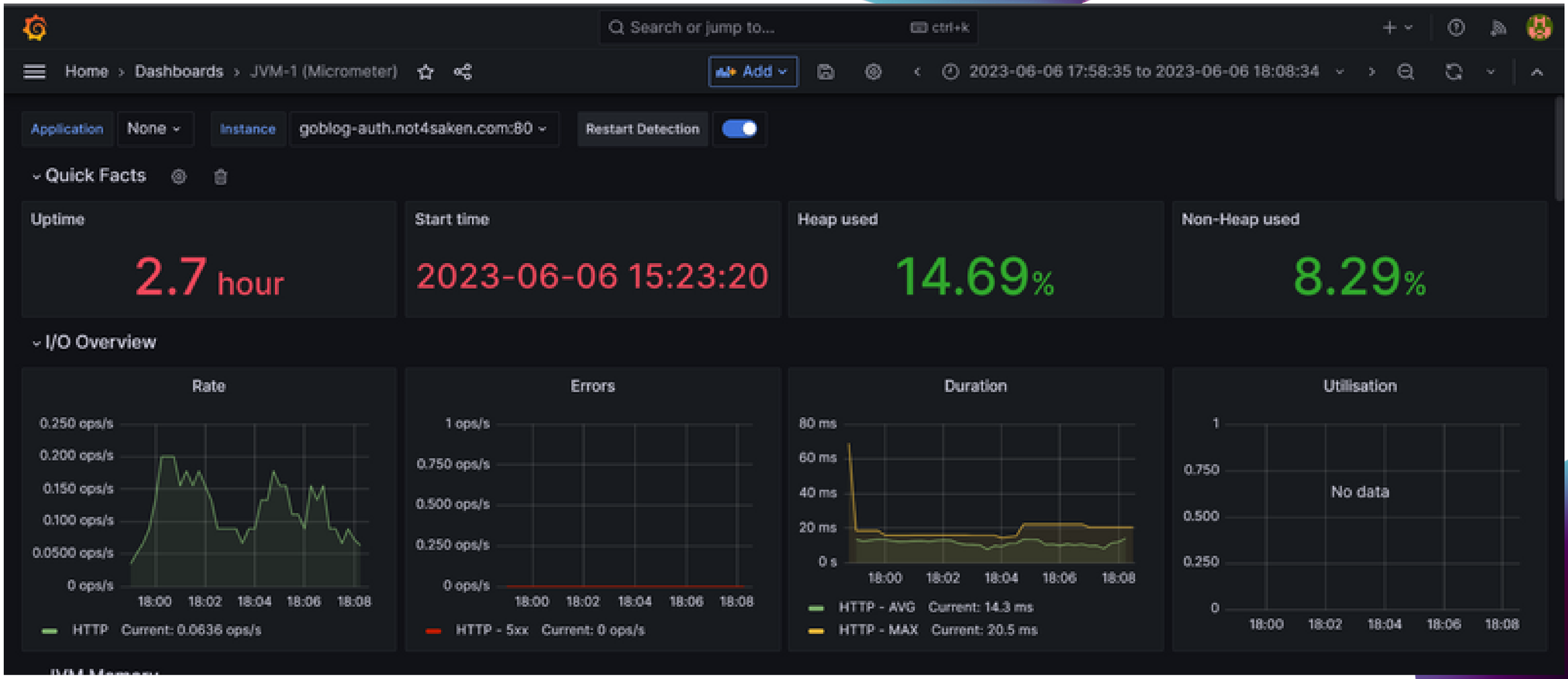
```java
@Override
public CompletableFuture<Post> create(CreatePostRequest request) {
    return CompletableFuture.supplyAsync(() -> {
        var user = userRepository.findById(request.getCreatorId()).orElseThrow();
```

```java
@Override
public CompletableFuture<Post> update(UpdatePostRequest request) {
    return CompletableFuture.supplyAsync(() -> {
        Optional<Post> post = postRepository.findById(request.getPostId());
```

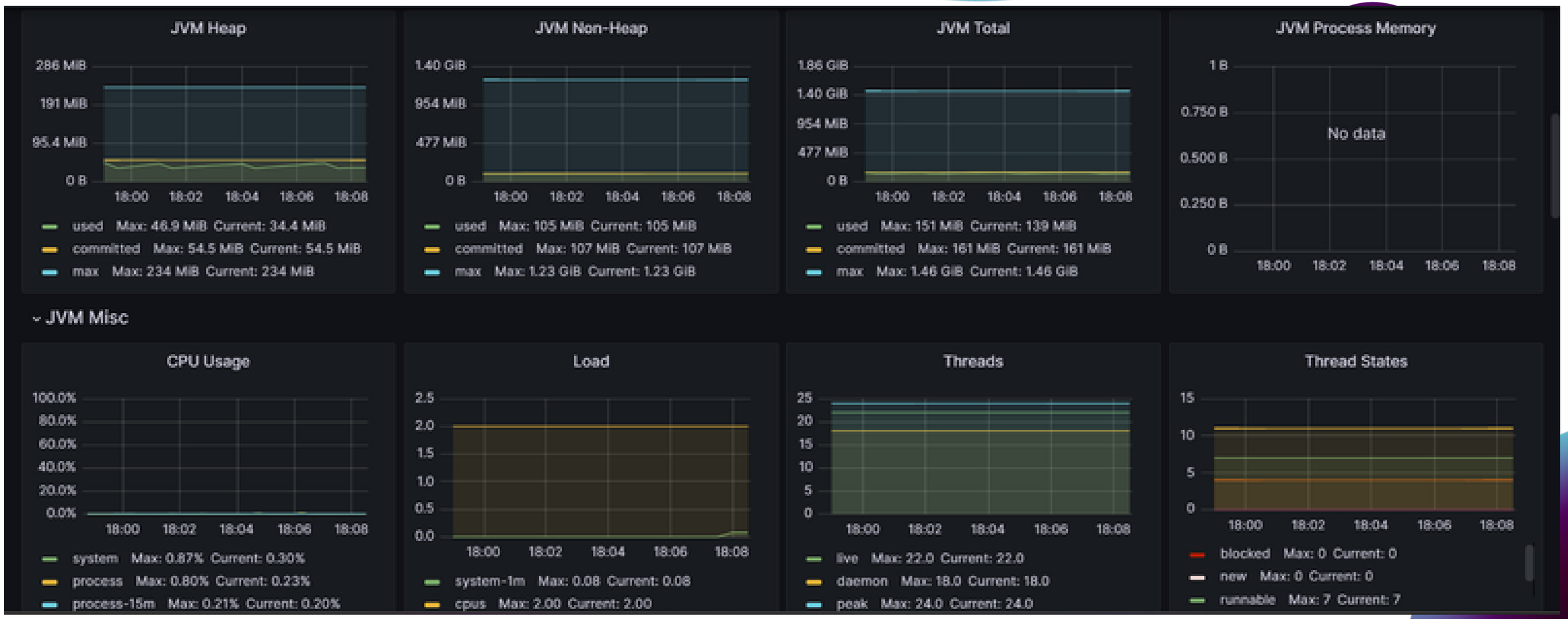
Concurrency & Asynchronous

```
public List<Notification> getNotifications(Integer userId) {

    List<Notification> notifications =  notificationRepository.getNotifications(userId);
    Thread updateNotification = new Thread(() -> {

            notifications.forEach(notif -> notif.setNotified(true));

            notificationRepository.saveAllAndFlush(notifications);
    });

    updateNotification.start();
    return notifications;
```

Profiling


Search or jump to... ctrl+k
Home › Dashboards › JVM-1 (Micrometer)
Add
2023-06-06 17:58:35 to 2023-06-06 18:08:34
Application None
Instance goblog-auth.not4saken.com:80
Restart Detection
Quick Facts
Uptime
2.7 hour
Start time
2023-06-06 15:23:20
Heap used
14.69%
Non-Heap used
8.29%
I/O Overview
Rate
0.250 ops/s
0.200 ops/s
0.150 ops/s
0.100 ops/s
0.0500 ops/s
0 ops/s
18:00 18:02 18:04 18:06 18:08
HTTP Current: 0.0636 ops/s
Errors
1 ops/s
0.750 ops/s
0.500 ops/s
0.250 ops/s
0 ops/s
18:00 18:02 18:04 18:06 18:08
HTTP - 5xx Current: 0 ops/s
Duration
80 ms
60 ms
40 ms
20 ms
0 s
18:00 18:02 18:04 18:06 18:08
HTTP - AVG Current: 14.3 ms
HTTP - MAX Current: 20.5 ms
Utilisation
1
0.750
0.500
0.250
0
No data
18:00 18:02 18:04 18:06 18:08

Profiling


JVM Heap
286 MiB
191 MiB
95.4 MiB
0 B
18:00 18:02 18:04 18:06 18:08
used Max: 46.9 MiB Current: 34.4 MiB
committed Max: 54.5 MiB Current: 54.5 MiB
max Max: 234 MiB Current: 234 MiB
JVM Non-Heap
1.40 GiB
954 MiB
477 MiB
0 B
18:00 18:02 18:04 18:06 18:08
used Max: 105 MiB Current: 105 MiB
committed Max: 107 MiB Current: 107 MiB
max Max: 1.23 GiB Current: 1.23 GiB
JVM Total
1.86 GiB
1.40 GiB
954 MiB
477 MiB
0 B
18:00 18:02 18:04 18:06 18:08
used Max: 151 MiB Current: 139 MiB
committed Max: 161 MiB Current: 161 MiB
max Max: 1.46 GiB Current: 1.46 GiB
JVM Process Memory
1 B
0.750 B
0.500 B
0.250 B
0 B
No data
18:00 18:02 18:04 18:06 18:08
JVM Misc
CPU Usage
100.0%
80.0%
60.0%
40.0%
20.0%
0.0%
18:00 18:02 18:04 18:06 18:08
system Max: 0.87% Current: 0.30%
process Max: 0.80% Current: 0.23%
process-15m Max: 0.21% Current: 0.20%
Load
2.5
2.0
1.5
1.0
0.5
0.0
18:00 18:02 18:04 18:06 18:08
system-1m Max: 0.08 Current: 0.08
cpus Max: 2.00 Current: 2.00
Threads
25
20
15
10
5
0
18:00 18:02 18:04 18:06 18:08
live Max: 22.0 Current: 22.0
daemon Max: 18.0 Current: 18.0
peak Max: 24.0 Current: 24.0
Thread States
15
10
5
0
18:00 18:02 18:04 18:06 18:08
blocked Max: 0 Current: 0
new Max: 0 Current: 0
runnable Max: 7 Current: 7

Profiling

Label matcher copied to clipboard

2023-06-06 14:49:23

{**error**="none", **exception**="none", **instance**="goblog-blog.not4saken.com:80", **job**="spring_boot_app", **method**="DELETE", **outcome**="SUCCESS", **status**="200", **uri**="/api/v1/comment/delete/{commentId}"}	0.038753992
{**error**="none", **exception**="none", **instance**="goblog-blog.not4saken.com:80", **job**="spring_boot_app", **method**="DELETE", **outcome**="SUCCESS", **status**="200", **uri**="/api/v1/post/delete_post/{id}"}	0.0399692025
{**error**="none", **exception**="none", **instance**="goblog-blog.not4saken.com:80", **job**="spring_boot_app", **method**="GET", **outcome**="SUCCESS", **status**="200", **uri**="/actuator/prometheus"}	0.013053528857142857
{**error**="none", **exception**="none", **instance**="goblog-blog.not4saken.com:80", **job**="spring_boot_app", **method**="GET", **outcome**="SUCCESS", **status**="200", **uri**="/api/v1/notification/get_new_notification/{userId}"}	0.048675564081081084
{**error**="none", **exception**="none", **instance**="goblog-blog.not4saken.com:80", **job**="spring_boot_app", **method**="GET", **outcome**="SUCCESS", **status**="200", **uri**="/api/v1/notification/get_old_notification/{userId}"}	0.0438754818
{**error**="none", **exception**="none", **instance**="goblog-blog.not4saken.com:80", **job**="spring_boot_app", **method**="GET", **outcome**="SUCCESS", **status**="200", **uri**="/api/v1/post/get_all_post"}	0.058433795875
{**error**="none", **exception**="none", **instance**="goblog-blog.not4saken.com:80", **job**="spring_boot_app", **method**="GET", **outcome**="SUCCESS", **status**="200", **uri**="/api/v1/post/get_post/{id}"}	0.0360520045
{**error**="none", **exception**="none", **instance**="goblog-blog.not4saken.com:80", **job**="spring_boot_app", **method**="OPTIONS", **outcome**="SUCCESS", **status**="200", **uri**="UNKNOWN"}	0.0013912049863013698
{**error**="none", **exception**="none", **instance**="goblog-blog.not4saken.com:80", **job**="spring_boot_app", **method**="POST", **outcome**="SUCCESS", **status**="200", **uri**="/api/v1/comment/create"}	0.055862025800000006
{**error**="none", **exception**="none", **instance**="goblog-blog.not4saken.com:80", **job**="spring_boot_app", **method**="POST", **outcome**="SUCCESS", **status**="200", **uri**="/api/v1/post/create_post"}	0.03853941285714286
{**error**="none", **exception**="none", **instance**="goblog-blog.not4saken.com:80", **job**="spring_boot_app", **method**="POST", **outcome**="SUCCESS", **status**="200", **uri**="/api/v1/post/search"}	0.04478026222222223
{**error**="none", **exception**="none", **instance**="goblog-blog.not4saken.com:80", **job**="spring_boot_app", **method**="PUT", **outcome**="SUCCESS", **status**="200", **uri**="/api/v1/post/update_post"}	0.07962826766666667

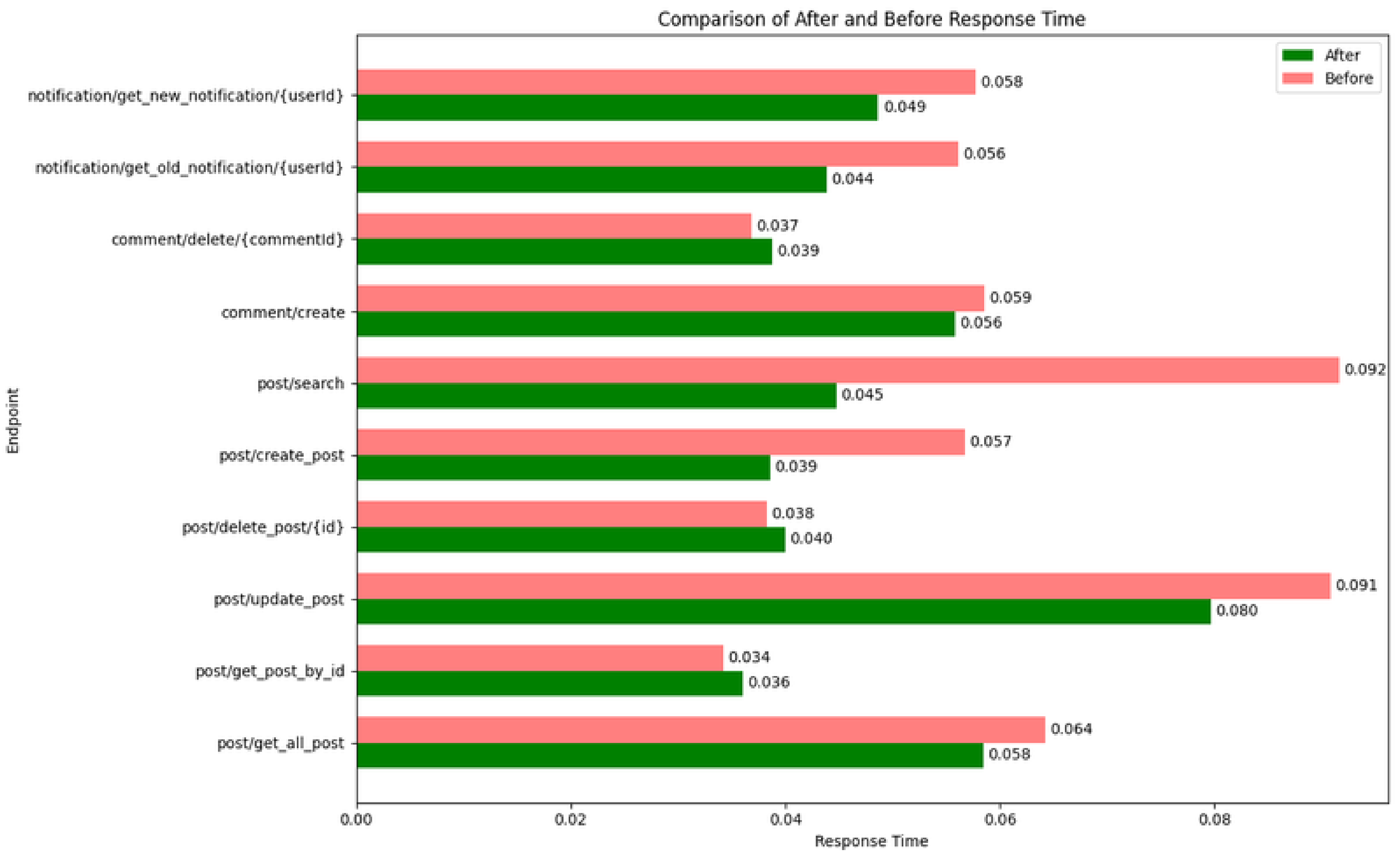
Profiling

Fitur yang dapat dioptimisasi:

1. **Search**
    - Penggunaan logic yang tidak optimal dengan melakukan loop untuk matching pattern.
    - Solusi: Memakai query ILIKE
2. **Post**
    - Menghindari blocking call dari update dan create post
    - Solusi: Mengimplementasikan async dan menambah thread
3. **Notification**
    - Return response sebelum mengupdate seluruh notifikasi menjadi notified.
    - Solusi: Membuat thread yang berjalan secara asinkronus.

Fitur yang sudah optimal:

Semua fitur selain fitur di atas sudah optimal


Comparison of After and Before Response Time
After
Before
notification/get_new_notification/{userId}
0.058
0.049
notification/get_old_notification/{userId}
0.056
0.044
comment/delete/{commentId}
0.037
0.039
comment/create
0.059
0.056
post/search
0.092
0.045
post/create_post
0.057
0.039
post/delete_post/{id}
0.038
0.040
post/update_post
0.091
0.080
post/get_post_by_id
0.034
0.036
post/get_all_post
0.064
0.058
Endpoint
0.00
0.02
0.04
0.06
0.08
Response Time

THANK YOU