소프트웨어 설계 명세서

Software Design Specification

프로젝트명 CodeReview

문서 버전 V 1.0

최종 작성일 2022.11.18.

팀원 손배준
송대선
유승훈
지종권
최현승

< 목 차 >

1. Introduction ····· 3

2. System Overview ····· 4

3. Design Considerations ····· 6

4. Architectural Strategies ····· 7

5. System Architecture ····· 9

6. Policies and Tactics ····· 10

7. Detailed System Design ····· 11

8. Glossary ····· 17

1. Introduction

1.1. Purpose

이 문서에서는 코드 리뷰 게시판을 위한 소프트웨어 아키텍처 및 설계를 자세히 설명한다. 본 문서에서는 시스템의 이해를 돕기위해 시스템 설계에 대한 여러 가지 관점을 제공한다.

1.2. Scope

이 문서는 코드 리뷰 시스템 1.0의 아키텍처와 설계를 제공한다. 소프트웨어 요구사항 명세서에 설명되어 있는 기능 및 비기능 요구사항을 어떻게 디자인할 것 인지를 보여준다.

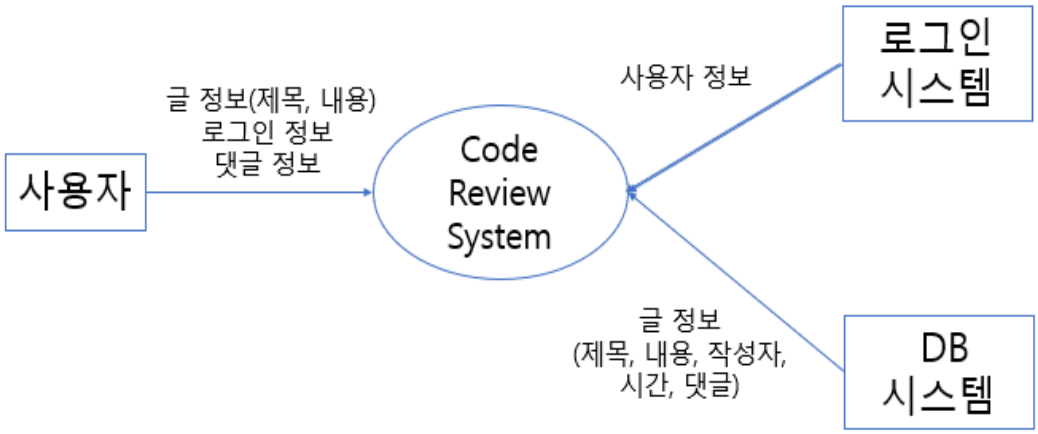
1.3. References

- SDS_example 01.pdf
- SDS_example 02.pdf
- SDS_example 03.pdf
- https://slidesplayer.org/slide/16562028/
- https://snowdeer.github.io/sw-architecture/2021/05/14/sw-quality-attribute-example/
- https://k-mozzi.tistory.com/36
- https://mommoo.tistory.com/62

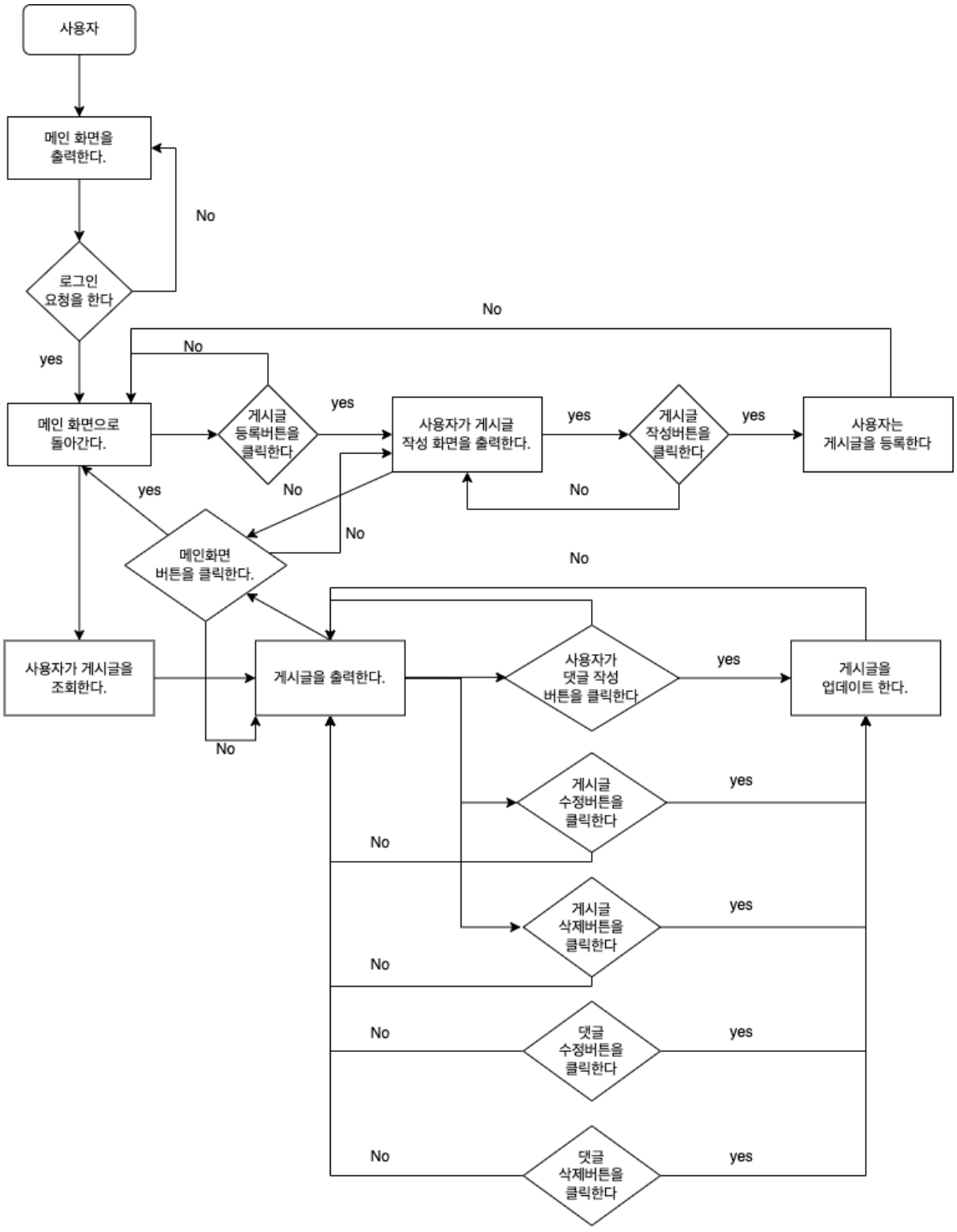
2. System Overview

이 프로젝트는 게시판 시스템이다. 먼저 로그인 시스템은 사용자의 정보를 저장하고 로그인 정보 요청 시 데이터를 내보내는 시스템이다. 사용자는 코드 리뷰 시스템에 로그인하여 글의 제목과 내용을 적고 글을 등록할 수 있다. 본인 게시물을 포함하여 다른 인원의 게시물에 댓글을 남길 수도 있다.
DB 시스템은 게시판에 글 등록, 삭제, 수정, 조회, 댓글 작성, 댓글 수정, 댓글 삭제를 하고 난 후 데이터를 저장하고 관리하는 시스템이다. 이 소프트웨어는 수정이나 삭제를 수행한 작업에 대해서는 복구할 수 없다.
코드 리뷰 시스템은 자신의 코드를 공유하여 사람들과 소통 할 수 있는 소프트웨어 시스템이다.

2.1. System Context Diagram



2.2. System Flow



3. Design Considerations

3.1. General Constraints

- OS - 윈도우에서만 가능
- 로그인 - 구글 계정만 사용할 수 있다.
- 사용자 정보 보호 - 개인정보 보호법에 따라 사용자의 인적 사항 정보 보호가 필요하다.
- 웹브라우저 - 게시판 서비스에서 지원 가능한 웹브라우저를 이용해야 한다.
- 웹 접근성 지침 - “한국형 웹 콘텐츠 접근성 지침 2.0”을 따른다.
- 소스 코드 최적화 - 소프트웨어 제작 시 소스 코드의 최적화를 통하여 시스템 효율을 향상시킨다.

3.2. Goals and Guidelines

- DRY - Don't Repeat Yourself : 중복되는 함수나 코드는 하나의 공통의 컴포넌트에 넣고 사용한다. 큰 시스템을 여러 조각으로 나누고 서로 참조한다.

- KISS - Keep it simple, stupid : KISS는 소프트웨어 디자인을 간단하고 단순하게 하는 것이다.

- YAGNI - You Ain't Gonna Need it : 미리 함수나 코드를 작성하지 말고 지금 필요한 기능만 추가해야 한다. 미리 함수나 코드를 작성해두면 분석 자체가 더 어려워지고 버그가 발생할 가능성이 커진다.

- 시스템은 예정된 시간 내에 완전히 작동할 수 있어야 한다.

3.3. Development Methods

- 폭포수 모델 - 전 단계가 수행되어 완료되기 전에는 다음 단계로 진행할 수 없도록 제한하는 방법을 채택했다.

4. Architectural Strategies

소프트웨어 아키텍처 설계 전략을 4가지로 구성했다.
1. 변경용이성
2. 성능
3. 사용성
4. 보안

4.1. 변경용이성

특정 변경 요구사항이 시스템에 반영될 수 있도록 하는 소프트웨어의 능력을 의미한다.

- 조회수 : 조회수를 확인하여 인기가 있는 게시물에 들어가 코드에 대한 서로의 의견을 더 많이 공유하게 된다. 이로써 코드리뷰 게시판의 활성화가 이루어지게 된다.

- 디자인 : 코드리뷰 게시판의 컬러나 폰트 스타일, 레이아웃 등의 디자인을 게시판의 특성에 맞게 적용함으로써 코드리뷰 게시판의 가독성을 높여주게 된다.

4.2. 성능

사용자의 요구 기능을 최소의 자원을 사용하면서 얼마나 빨리 많은 기능을 수행하는가를 육안, 또는 도구를 통하여 점검하는 것을 말한다. 이를 책정하기 위한 지표는 주어진 시간에 처리할 수 있는 트랜잭션의 수인 가용성, 사용자의 입력이 끝난 후 응답 출력이 게시될 때까지의 시간이 응답 시간, 사용자가 요구를 입력한 시점으로부터 트랜잭션 처리 후 출력을 완료할 때까지의 시간인 경과 시간이 있다. 이러한 트랜잭션의 특징은 크게 원자성, 일관성, 독립성, 지속성 4가지로 이루어져 있다.

- 원자성 : 트랜잭션이 데이터베이스에 모두 반영되던가, 아니면 전혀 반영되지 않아야 한다는 것이다. 트랜잭션 단위로 데이터가 처리되지 않는다면, 설계한 사람은 데이터 처리 시스템을 이해하기 힘들 뿐 아니라, 오작동했을 시 원인을 찾기 힘들다.

- 일관성 : 트랜잭션의 작업 처리결과가 항상 일관성이 있어야 한다. 트랜잭션이 진행되는 동안 데이터베이스가 변경되더라도 업데이트된 데이터베

이스로 트랜잭션이 진행되는 것이 아니라, 처음의 트랜잭션을 진행하기 위해 참조된 데이터베이스로 진행한다.

- 독립성 : 어떤 하나의 트랜잭션이라도 다른 트랜잭션의 연산에 끼어들 수 없다는 점이다. 특정 트랜잭션이 완료될 때까지 다른 트랜잭션이 특정 트랜잭션의 결과를 참조할 수 없다.

- 지속성 : 트랜잭션이 성공적으로 완료됐을 경우 결과는 영구적으로 반영되어야 한다는 점이다.

우리는 이러한 트랜잭션의 특징 4가지를 고려하여 성능을 최적화할 수 있다.

4.3. 사용성

사용자가 어떤 도구를 특정 목적을 달성하기 위해 사용할 때 어느 정도 사용하기 쉬운가를 의미한다. 사용자와 컴퓨터 간에 일어나는 상호 작용에서 경험하는 '사용의 품질'을 의미한다.

- 페이징 : 한 페이지에 출력될 게시물 수, 한 화면에 출력될 페이지 수, 현재 페이지 번호를 지정한다. 따라서 사용자는 검색 결과에 대해 어떤 페이지까지 볼 것인지, 어디까지 자세히 읽고 어디까지 그냥 지나쳐도 좋을지 쉽게 결정을 내릴 수 있게 된다.

- 코드 블록 : 코드를 작성할 때 코드 블록을 사용하여 코드를 작성하는 칸을 따로 지정하여 작성한다. 코드리뷰 게시판의 가독성과 가시성이 향상되어 사용자들이 이용하는 데에 편리함을 제공할 수 있다.

4.4. 보안

안전한 소프트웨어 개발을 위해 소스 코드 등에 존재할 수 있는 잠재적인 보안 취약점을 제거하고, 보안을 고려하여 기능을 설계 및 구현하는 등 소프트웨어 개발 과정에서 지켜야 할 일련의 보안 활동을 말한다. 이러한 보안의 설계항목에는 여러 가지가 있다.

- DBMS 조회 및 결과 검증 : DBMS 조회 시 질의문(SQL) 내 입력값과 그 조회 결과에 대한 유효성 검증 방법(필터링 등) 설계 및 유효하지 않은 값에 대한 처리 방법 설계

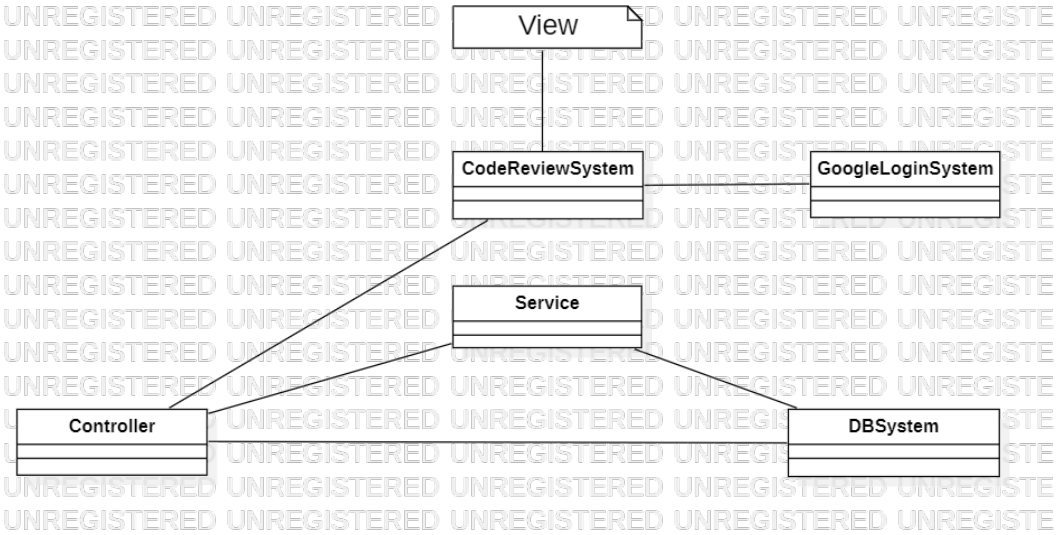
- 중요정보 저장 : 중요정보(비밀번호, 개인정보 등)를 저장 또는 보관하는 방법이 안전하도록 설계

- 중요정보 전송 : 중요정보(비밀번호, 개인정보 등)를 전송하는 방법이 안전하도록 설계

- 예외 처리 : 오류메시지에 중요정보가 노출되거나, 부적절한 에러 또는 오류 처리로 의도치 않은 상황이 발생하지 않도록 설계

5. System Architecture

5.1 Primary Components



- Controller : 사용자의 요청이 진입하는 지점이며 요청에 따라 어떤 처리를 할지 결정해준다. 단, controller는 단지 결정만 해주고 실질적인 처리는 서비스에서 담당한다. 그리고 사용자에게 View(또는 서버에서 처리된 데이터를 포함하는 View)를 응답으로 보내준다.

- Service : DAO 가 DB에서 받아온 데이터를 전달받아 가공하는 것이다. 자신을 어떤 컨트롤러가 호출하든 상관없이 필요한 매개변수만 준다면 자신의 비즈니스 로직을 처리하게 된다. 단순 Web 기반이 아니라 추후 native app으로 view 단이 변경되더라도 Service는 view에 종속적인 코드가 없으므로 그대로 재사용 할 수 있어야 한다. 그리고 추가적인 요청사항이 들어오면 기존 소스를 수정하는 것이 아니라 기존 service 인터페이스를 구현한 다른 클래스를 구현해 그 객체를 사용하게끔 하는 것이다.

- DB System : 데이터를 DB에 저장하고 DBMS를 사용하여 필요한 정보를 생성하는 컴퓨터 중심의 시스템이다. 구성 요소로는 데이터베이스, 데이터베이스 관리 시스템, 데이터 언어, DB 사용자, DB 컴퓨터가 있다. 데이터베이스는 데이터를 저장하고, 데이터베이스 관리 시스템은 DB를 생성, 관리, 조직함으로써 사용자와 DB를 연결해주는 소프트웨어이고, 데이터 언어는 DB 정의와 조작, 제어를 위한 DB 전용 언어이다. DB 사용자는 데이터 언어를 사용해서 DB에 접근하는 사람으로, 일반 사용자와 응용 프로그래머, DB 관리자로 구분한다. 마지막으로 DB 컴퓨터는 효율적인 유지/관리를 위해서 DB에 대한 연산을 전담하는 DB 관리 전용 컴퓨터이다. 이 시스템에서 DB System은 게시판에 대한 정보와 사용자의 정보를 저장하고 시스템이 이러한 정보를 불러올 때는 Controller, Service를 이용해서 정보를 불러온다. DB System 안에서 사용자가 원하는 데이터를 가져오게 하는 역할을 함께 묶어놓고 그렇게 함으로써 원하는 데이터를 가져온다.

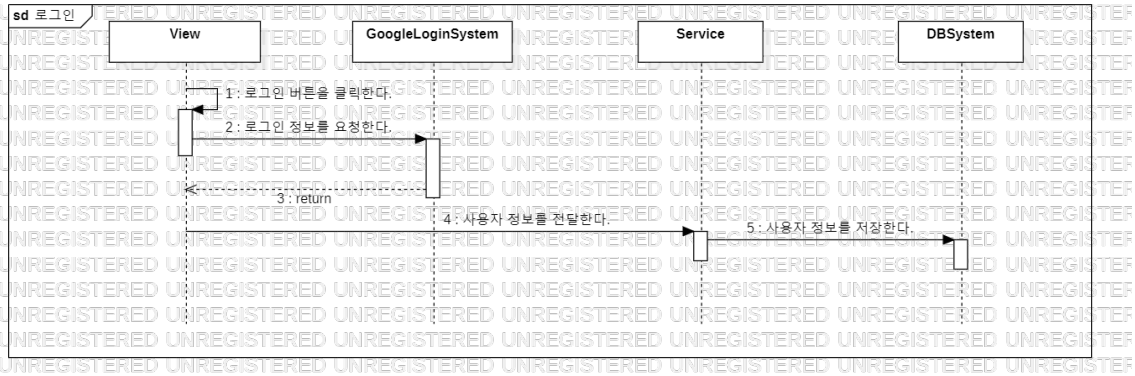
- 구글 로그인 시스템 : 사용자가 로그인하려고 할 때 정보를 구글 로그인 시스템에서 확인한다. 이 시스템은 코드 리뷰 시스템과는 별개로 로그인을 위한 사용자의 정보는 구글이 가지고 있다. 그렇게 구글에서 로그인한 후, 정보는 DB System으로 넘겨서 사용자의 정보를 저장한다. 이후에 사용자의 정보가 필요할 시에는 구글 로그인 시스템이 아닌 DB System에서 들고 온 뒤 사용하게 된다.

6. Policies and Tactics

- 자바 코딩 규칙(Java Code Conventions)을 따른다.
- HTML 코드 작성 규칙을 따른다.
- Maria DB 코드 작성 규칙을 따른다.
- 스프링 부트에서 공식으로 지원하는 템플릿 엔진인 Mustache를 사용한다.
- 구성 요소 간의 인터페이스를 명확하게 정의하려고 시도한다.

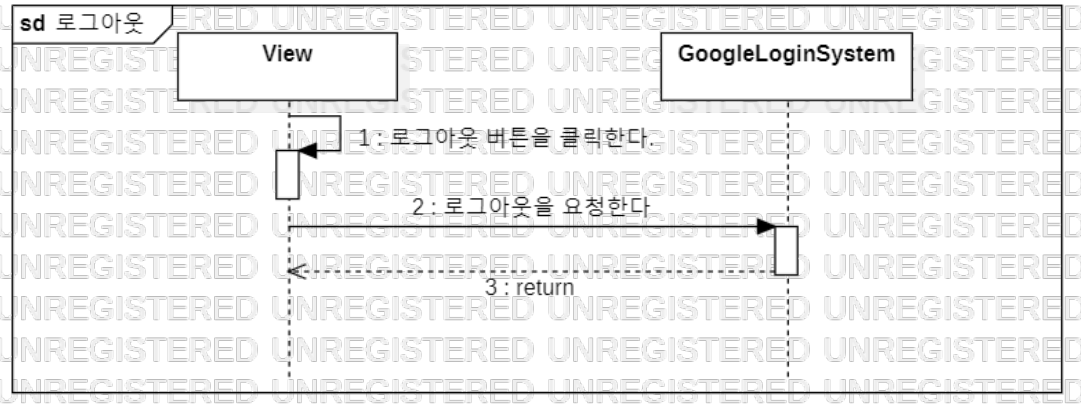
7. Detailed System Design

7.1. 로그인



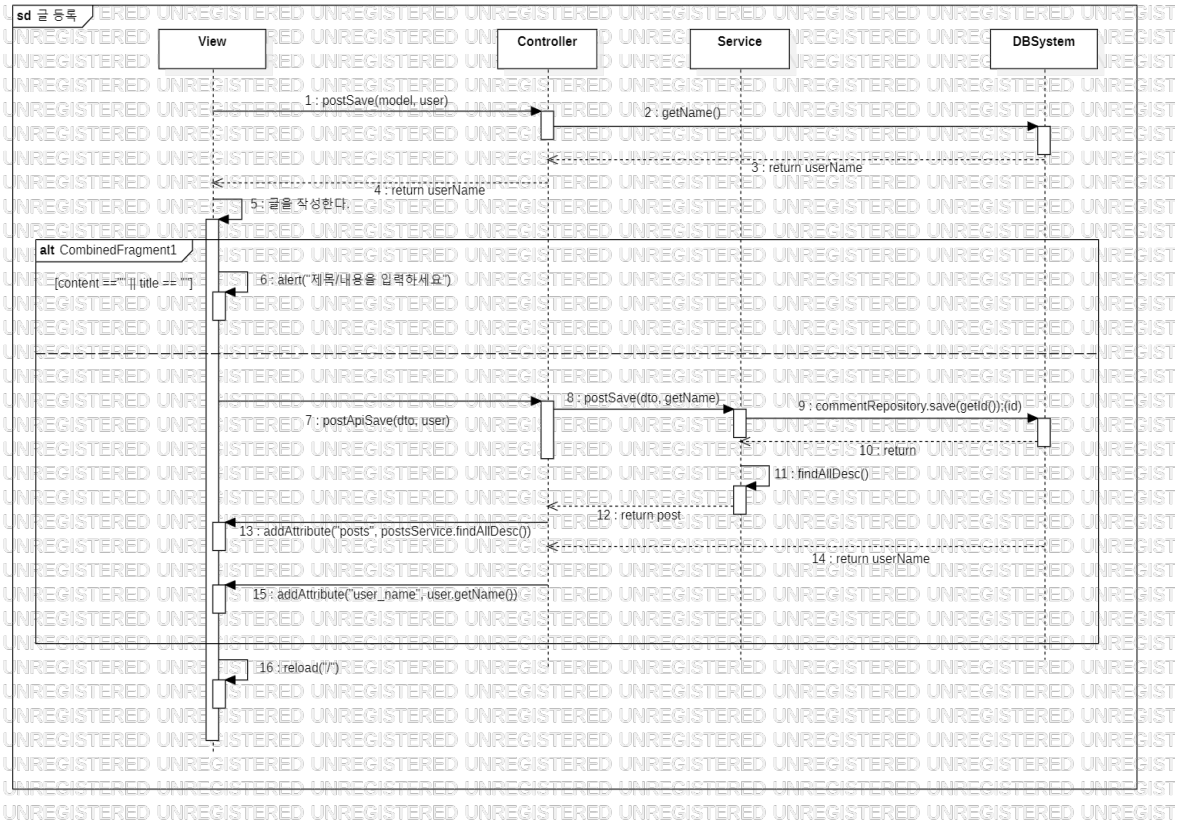
1. 로그인 버튼을 클릭한다.
2. 구글 로그인 시스템에 로그인 정보를 요청한다.
3. 요청 결과를 반환한다.
4. Service에 사용자 정보를 전달한다.
5. 데이터베이스에 사용자 정보를 저장한다.

7.2. 로그아웃



1. 로그아웃 버튼을 클릭한다.
2. 구글 로그인 시스템에 로그아웃을 요청한다.
3. 요청한 결과를 반환한다.

7.3. 글 등록

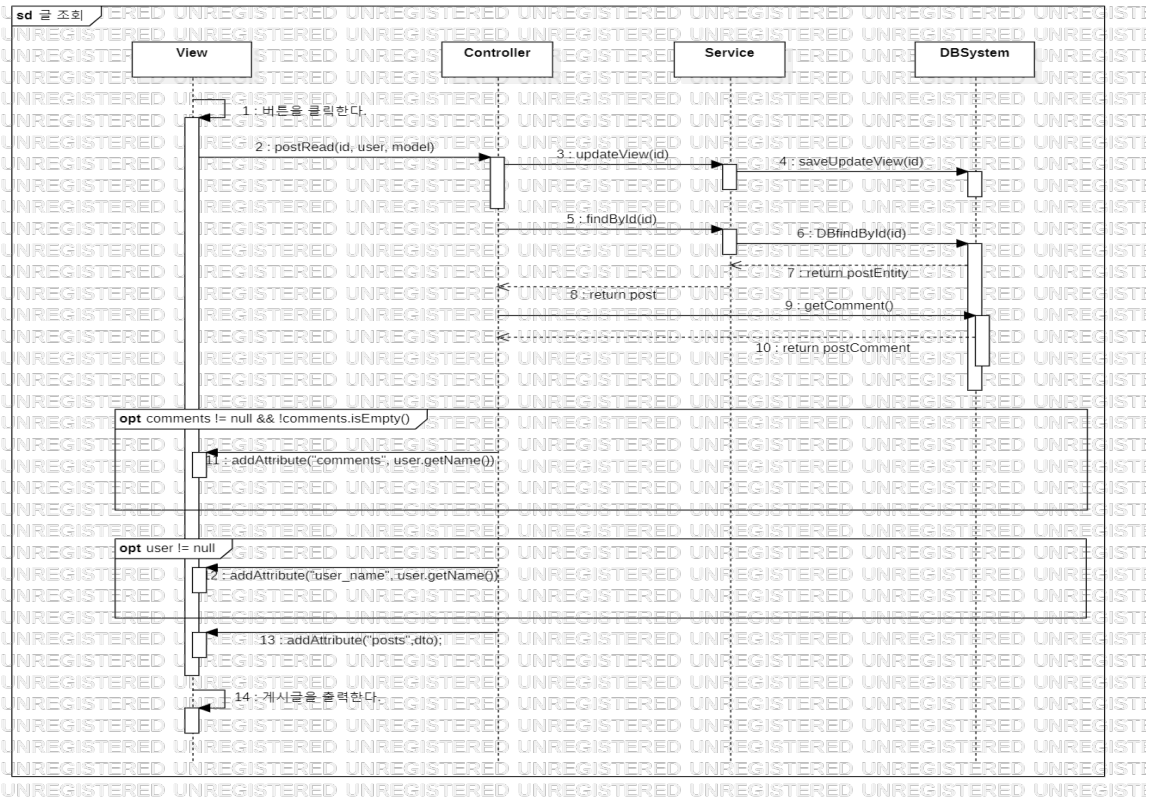


1. Controller에게 게시글 저장을 요청한다.
2. DB에 게시글 이름을 넘겨준다.
3. Controller에 사용자 이름을 반환해준다.
4. 화면에 사용자 이름을 반환한다.
5. 글을 작성한다.

[alt] 제목 또는 본문의 내용을 입력하지 않았을 경우 6번 실행, 아닐 시 7~15번 실행한다.

6. “제목/내용을 입력하세요” 팝업창을 띄운다.
7. Controller에게 게시글의 API 저장을 요청한다.
8. Service에게 게시글 저장을 요청한다.
9. DB System에 댓글 정보 저장을 요청한다.
10. Service에 요청한 결과를 반환해준다.
11. 모든 칼럼을 내림차순으로 조회한다.
12. Controller에 게시글을 반환해준다.
13. 게시글들을 내림차순으로 화면에 불러온다.
14. Controller에 사용자 이름을 반환해준다.
15. 화면에 게시글 이름을 보여준다.
16. 화면을 다시 불러온다.

7.4. 글 조회



1. 버튼을 클릭한다.
2. Controller에게 게시글을 읽기를 요청한다.
3. Service에게 게시판의 업데이트를 요청한다.
4. 업데이트한 게시판을 DB에 저장한다.
5. Service에게 아이디 검색을 요청한다.
6. DB에서 아이디를 검색한다.
7. Service에 게시글 엔티티를 반환해준다.
8. Controller에 요청한 결과를 반환해준다.
9. DB에서 댓글 정보를 요청한다.
10. Controller에게 게시글의 댓글 정보를 반환한다.

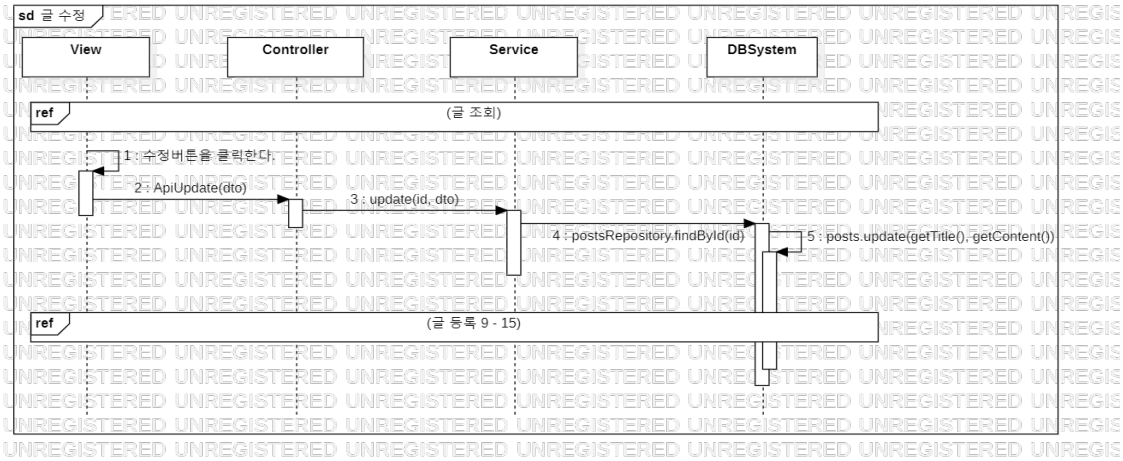
[opt] 댓글이 있고 댓글이 내용이 있으면 11번을 실행한다.

11. 댓글과 댓글 쓴 사용자의 이름을 불러온다.

[opt] 사용자가 있으면 12번을 실행한다.

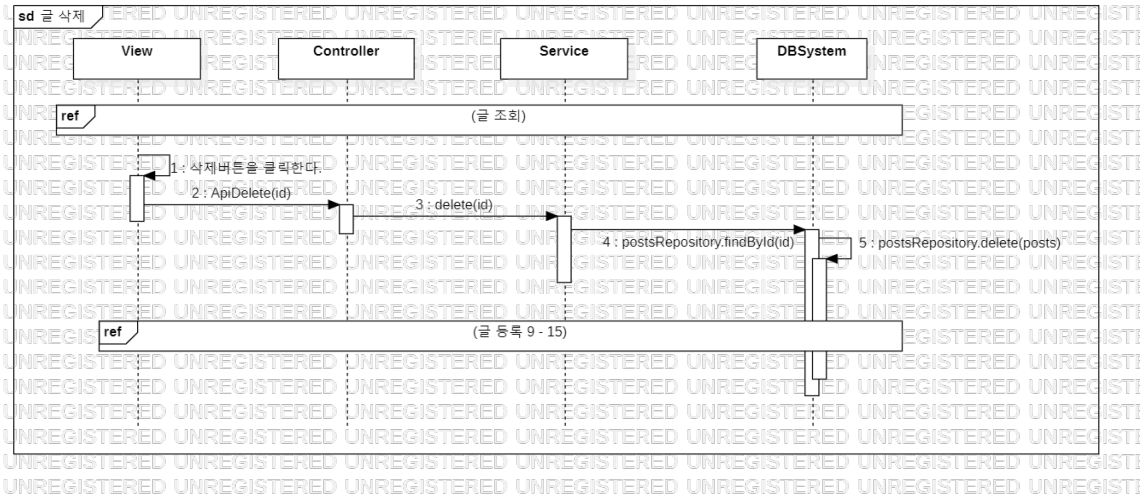
12. 사용자의 이름을 불러온다.
13. 화면에 게시글들을 불러온다.

7.5. 글 수정



[ref] 글 조회
1. 수정 버튼을 클릭한다.
2. Controller에게 API 업데이트를 요청한다.
3. Service에게 게시글 업데이트를 요청한다.
4. DB에서 아이디를 통해 해당 게시물을 검색한다.
5. 게시물의 제목과 내용을 업데이트한다.

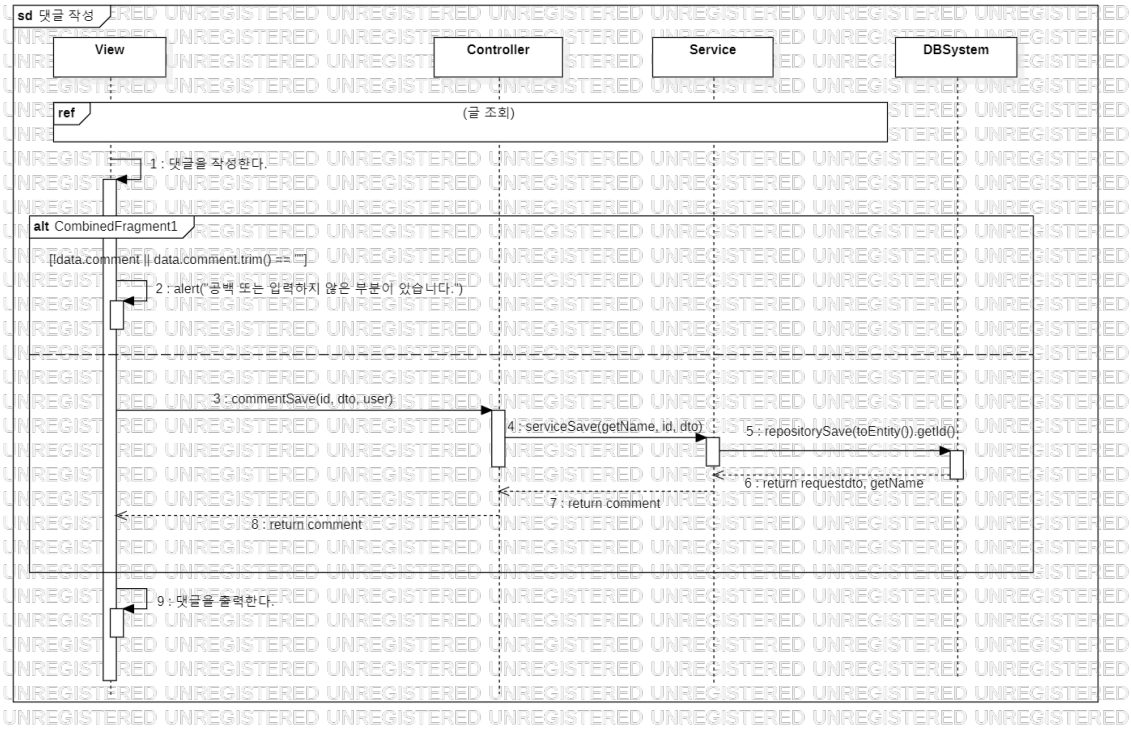
7.6. 글 삭제



[ref] 글 조회
1. 삭제 버튼을 클릭한다.
2. Controller에게 API 삭제를 요청한다.
3. Service에게 글 삭제를 요청한다.
4. DB에서 아이디를 통해 해당 게시물을 검색한다.
5. 해당 게시물을 삭제한다.

[ref] 글 등록 9~15번

7.7. 댓글 작성



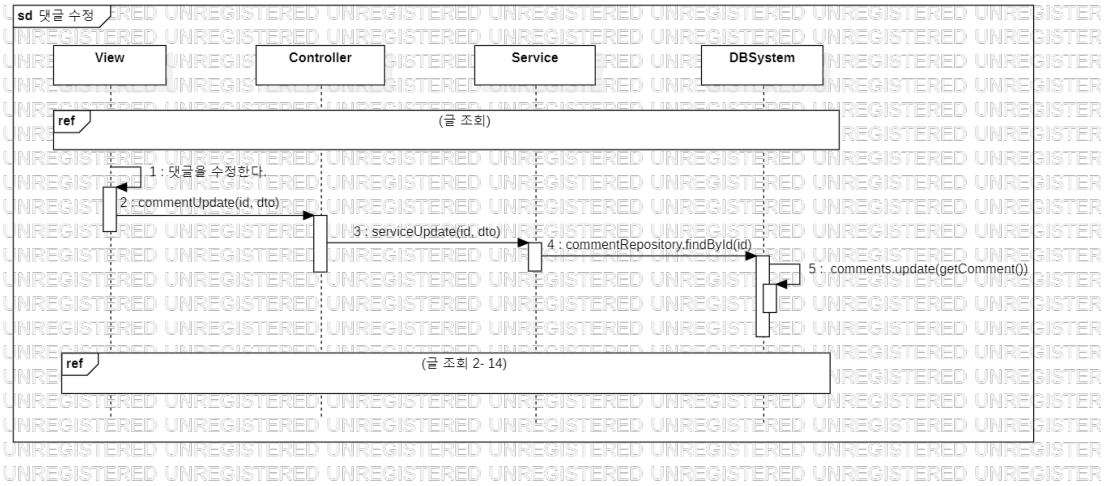
[ref] 글 조회

1. 댓글을 작성한다.

[alt] 댓글 내용이 공백 또는 입력하지 않은 부분이 있을 시 2번 실행, 아닐 시 3~8번 실행한다.

2. “공백 또는 입력하지 않은 부분이 있습니다.” 팝업창을 띄운다.
3. Controller에게 댓글 저장을 요청한다.
4. Service에게 댓글 정보 저장을 요청한다.
5. DB에 댓글 정보를 저장한다.
6. Service에 요청했던 결과를 반환한다.
7. Controller에 요청했던 결과를 반환한다.
8. 댓글을 화면에 반환해준다.
9. 화면에 댓글을 출력한다.

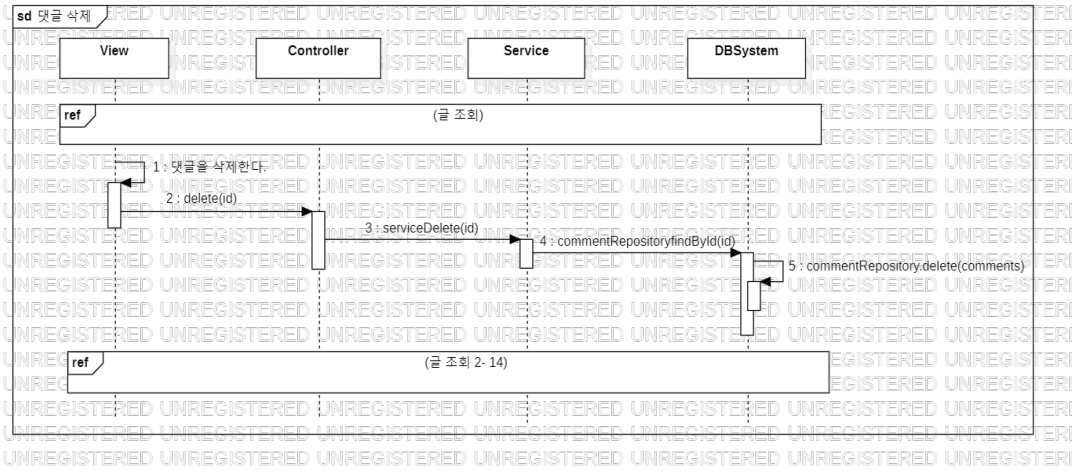
7.8. 댓글 수정



[ref] 글 조회
1. 댓글을 수정한다.
2. Controller에게 댓글 업데이트를 요청한다.
3. Service에게 업데이트를 요청한다.
4. DB에서 아이디를 이용해 해당 댓글을 검색한다.
5. 해당 댓글을 새로 업데이트해준다.

[ref] 글 조회 2~14번

7.9. 댓글 삭제



[ref] 글 조회
1. 댓글을 삭제한다.
2. Controller에게 댓글의 삭제를 요청한다.
3. Service에게 댓글 삭제를 요청한다.
4. DB에서 아이디를 이용해 해당 댓글을 검색한다.
5. 해당 댓글을 삭제한다.

[ref] 글 조회 2~14번

8. Glossary

- DBMS : 데이터베이스에 적재된 데이터 작업을 수행할 뿐만 아니라 데이터베이스를 보호하고 보안을 제공한다.

- OS : 사용자의 하드웨어, 시스템 리소스를 제어하고 프로그램에 대한 일반적 서비스를 지원하는 시스템 소프트웨어를 말한다.

- 트랜잭션 : 데이터베이스의 상태를 변화시키기 위해 수행하는 작업의 단위를 뜻한다.

- Paging : 사용자의 이용성을 고려하여 문서를 분할하여 출력 해 주는 방법이다.

- SQL : '구조화된 질의 언어'라는 뜻으로 관계형 데이터베이스에서 사용되는 언어이다.

- DAO : 데이터 사용 기능 담당 클래스이다. DB 데이터 조회나 수정, 입력, 삭제와 같은 로직을 처리하기 위해 사용한다.