



# 스터디올래

( 스터디 모임 관리 서비스 )

[ 스프링 및 JPA 기반 웹 애플리케이션 개발 ]


- 제작자 : 김 성 제
- 소스 코드 및 출처

<https://github.com/kimseongje3111/Study-Olle>



# 목차

- I. 프로젝트 개요 및 목표
- II. 개발 환경
- III. 요구 사항
- IV. 설계 구조
- V. 기본 전략 및 인프라
- VI. 동작 과정 및 상세 기능
- VII. 결과



Summary & Purpose

## 프로젝트 개요 및 목표

### ❖ **스터디 모임 관리 서비스**

주요 활동 지역과 관심 주제 태그를 통해 스터디를 만들거나 자신에게 맞는 스터디를 찾고, 스터디 모임에 참여할 수 있는 기능들을 포함한 사용자 웹 서비스 제공

### ❖ **스프링 및 JPA 기반 웹 애플리케이션**

대부분의 웹 애플리케이션이 기본적으로 갖추고 있는 기능들을 포함하여, **스프링** 및 **JPA** 뿐만 아니라 여러 **오픈 소스 기술**을 적용하여 실제 서비스에 가까운 웹 애플리케이션 개발

본 프로젝트의 **목표**는 ?

- 보편적인 게시판 제작 또는 회원 관리와 같은 간단한 기능 구현이 아닌 **실제 서비스** 개발을 통해 실전과 가까운 다양한 경험과 학습
- 신입 서버 개발자로서 발 디딤 하기 위해 평소 관심을 가지고 배웠던 **스프링** 및 **JPA** 관련 기술들을 실제로 적용해보며 자바 **백엔드** 관련 지식 학습 및 역량 강화
- 도메인과 비즈니스 로직의 설계 및 구현, 성능 이슈 해결, 테스트 등과 같이 **프로젝트 진행 과정** 및 **문제 해결 능력**의 성장

궁극적으로 **개발**의 주 목적은 사용자에게 제공할 양질의 **서비스**를 만드는 것 !

Environment

## 개발 환경

### 기본 환경

- ✓ IntelliJ
- ✓ JDK 1.8
- ✓ Spring Boot
- ✓ [Build] Gradle
- ✓ Git

## 핵심 라이브러리

### 백엔드 (Back-end)

- ✓ [Web] Spring MVC, Tomcat 9
- ✓ [ORM] JPA + Hibernate, Spring Data JPA
- ✓ [DB] PostgreSQL 12
- ✓ Spring Security
- ✓ Querydsl 4
- ✓ [Test] Junit 5

### 프론트엔드 (Front-end)

- ✓ [Template] Thymeleaf 3
- ✓ [NPM] Bootstrap 4, Open-source Libraries

※ NPM : 프로젝트 빌드 시, static 아래 package.json  
를 사용하여 정적 리소스로 사용 (버전 관리 용이)

Requirements

## 요구 사항

### 사용자 인터페이스 [1] : 계정 생성/관리

- 1) 회원 가입 및 로그인
- 2) 프로필 설정
- 3) 알림 설정
- 4) 관심 스터디 주제 / 주요 활동 지역 설정
- 5) 닉네임 변경
- 6) 패스워드 변경
- 7) 계정 삭제
- 8) 사용자 프로필 조회

### 사용자 인터페이스 [2] : 스터디 관리/참여

- 1) 스터디 생성 및 삭제
- 2) 스터디 주제 / 활동 지역 설정
- 3) 스터디 공개 및 종료
- 4) 스터디 팀원 모집
- 5) 스터디 배너 이미지 설정
- 6) 스터디 정보 변경
- 7) 스터디 삭제
- 8) 스터디 조회 / 검색
- 9) 스터디 가입 및 탈퇴

관리자

### 사용자 인터페이스 (3) : 모임 관리/참여

- 1) 모임 생성 및 취소
- 2) 모임 정보 변경
- 3) 모임 참가 신청 확인 및 거절
- 4) 모임 참가자 출석 체크
- 5) 모임 조회
- 6) 모임 참가 신청 및 취소

관리자

### 사용자 인터페이스 (4) : 알림

- 1) 읽지 않은 알림 조회
- 2) 읽지 않은 알림 모두 읽음으로 표시
- 3) 읽은 알림 조회
- 4) 읽은 알림 삭제

## 소프트웨어 인터페이스

### Application

- 1) 추천 스터디 제공
- 2) 스터디 기본 배너 이미지 제공
- 3) 스터디 자동 삭제
- 4) 알림 전송
- 5) 알림 자동 삭제
- 6) 잘못된 요청 처리

### SMTP

- 1) 사용자 인증 이메일 전송
- 2) 이메일 로그인 링크 전송

### Spring Security

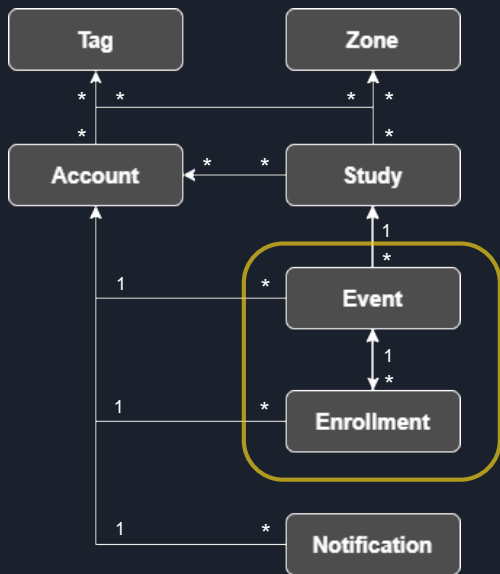
- 1) 접근 제어 및 권한 관리
- 2) 사용자 인증 및 권한 부여 (사용자 관리)
- 3) 현재 인증된 사용자 정보 참조
- 4) 로그인 유지



Structure

# 설계 구조

## 도메인 모델

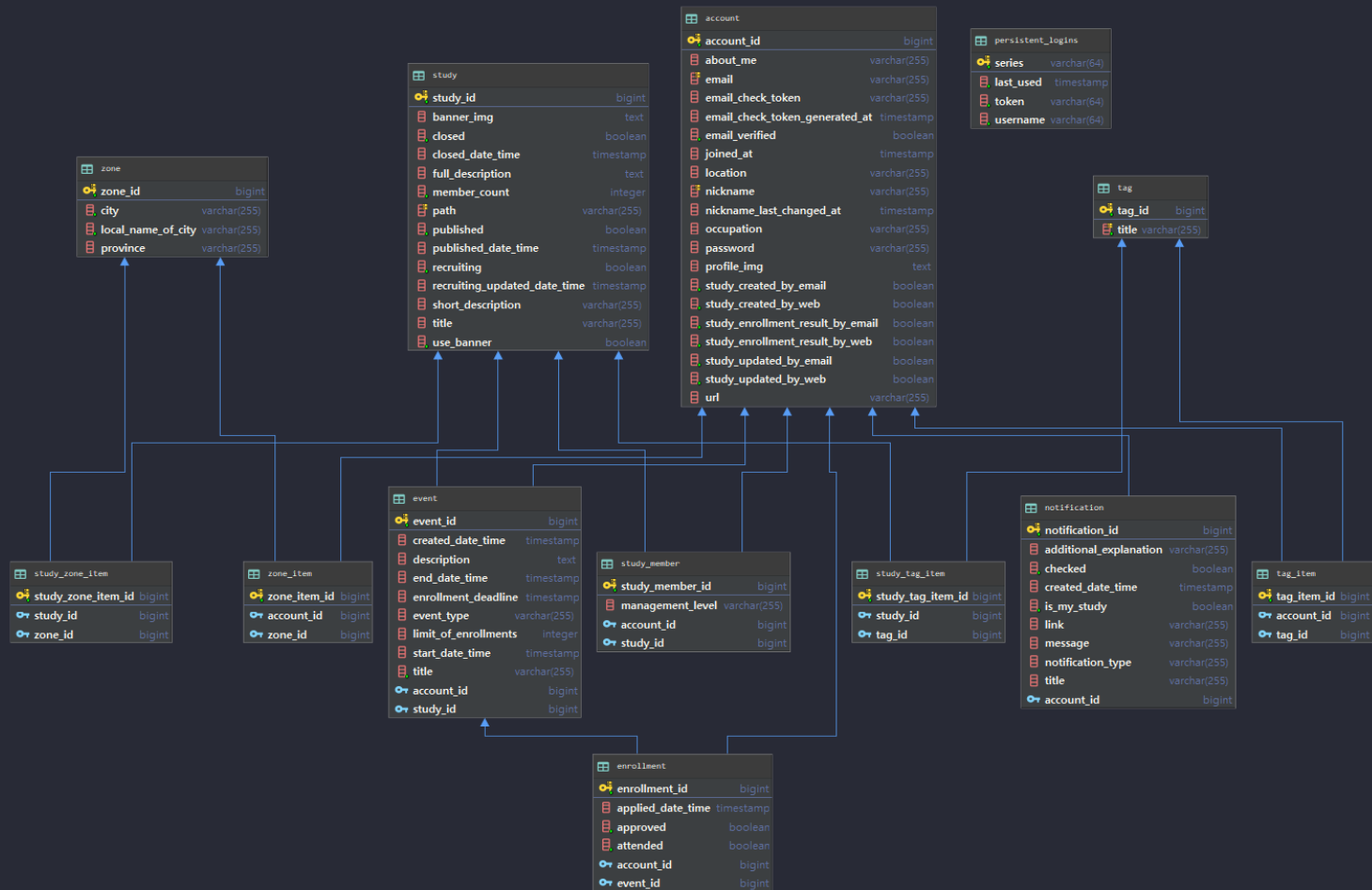


## 실제 패키지 구조

```
-- Application.java
-- infra
-- config
-- AppConfig.java
-- AppProperties.java
-- AsyncConfig.java
-- SecurityConfig.java
-- WebConfig.java
-- ConsoleMailService.java
-- EmailMessage.java
-- HtmlMailService.java
-- MailService.java
-- querydsl
-- Querydsl4RepositorySupport.java
-- modules
-- authentication
-- CurrentUser.java
-- UserAccount.java
-- controller
-- AccountController.java
-- AccountSettingController.java
-- domain
-- Account.java
-- PersistentLogins.java
-- TagItem.java
-- ZoneItem.java
-- form
-- AccountForm.java
-- NotificationsForm.java
-- PasswordForm.java
-- ProfileForm.java
-- SignUpForm.java
-- repository
-- AccountRepository.java
-- TagItemRepository.java
-- ZoneItemRepository.java
-- AccountRepositoryCustom.java
-- AccountRepositoryImpl.java
-- AccountService.java
-- AccountValidator.java
-- AccountFormValidator.java
-- PasswordFormValidator.java
-- SignUpFormValidator.java
-- mail
-- app_event
-- EnrollmentAppEventListener.java
-- EventAppEventListener.java
-- custom
-- enrollment
-- EnrollmentAppEvent.java
-- EnrollmentAppliedEvent.java
-- EnrollmentCancelledEvent.java
-- EnrollmentResultType.java
-- EnrollmentScheduledEvent.java
-- event
-- EventAppEvent.java
-- EventCancelledEvent.java
-- EventCreatedEvent.java
-- EventUpdatedEvent.java
-- controller
-- EventController.java
-- EventEnrollmentsController.java
-- domain
-- Enrollment.java
-- Event.java
-- EventType.java
-- form
-- EventForm.java
-- repository
-- EnrollmentRepository.java
-- EventRepository.java
-- custom
-- EnrollmentRepositoryCustom.java
-- EnrollmentRepositoryImpl.java
-- EventRepositoryImpl.java
-- service
-- EventService.java
-- validator
-- EventFormValidator.java
-- main
-- ExceptionAdvice.java
-- MainController.java
-- notification
-- NotificationInterceptor.java
-- NotificationMailSender.java
-- controller
-- NotificationController.java
-- domain
-- Notification.java
-- NotificationType.java
-- repository
-- NotificationRepository.java
-- custom
-- NotificationRepositoryCustom.java
-- NotificationRepositoryImpl.java
-- service
-- NotificationService.java
-- study
-- app_event
-- StudyAppEventListener.java
-- StudyCreatedEvent.java
-- StudyDeletedEvent.java
-- StudyMemberEvent.java
-- StudyMemberEventType.java
-- StudyUpdatedEvent.java
-- StudyUpdatedEventType.java
-- controller
-- StudyController.java
-- StudySettingController.java
-- domain
-- ManagementLevel.java
-- Study.java
-- StudyMember.java
-- StudyTagItem.java
-- StudyZoneItem.java
-- form
-- StudyDescriptionsForm.java
-- StudyForm.java
-- StudySearch.java
-- repository
-- StudyMemberRepository.java
-- StudyRepository.java
-- StudyTagItemRepository.java
-- StudyZoneItemRepository.java
-- custom
-- StudyMemberRepositoryCustom.java
-- StudyMemberRepositoryImpl.java
-- StudyRepositoryCustom.java
-- StudyRepositoryImpl.java
-- service
-- StudyService.java
-- validator
-- StudyFormValidator.java
-- tag
-- domain
-- Tag.java
-- form
-- TagForm.java
-- repository
-- TagRepository.java
-- service
-- TagService.java
-- zone
-- domain
-- Zone.java
-- form
-- ZoneForm.java
-- repository
-- ZoneRepository.java
-- service
-- ZoneService.java
```

# 데이터베이스

## 도메인 테이블



## 요청 처리 및 응답 (Spring MVC)

- 서버의 **Servlet Container** 에 전달된 클라이언트의 모든 **HTTP** 요청은 **DispatcherServlet** 이라는 프론트 컨트롤러에 의해 처리됨
- **DispatcherServlet** 은 적절한 컨트롤러에게 작업을 위임하고(**HandlerMapping**) 결과 뷰를 찾아(**ViewResolver**) 렌더링
- 컨트롤러에서 **@Valid** 애노테이션이나 커스텀 **Validator** 를 추가하여 입력 폼에 대한 검증
- **Interceptor** 를 등록하여 컨트롤러가 결과 뷰를 반환하기 전/후에 공통 작업 처리

## 도메인 엔티티와 연관 관계

- **ORM(Object Relational Mapping)** 기술은 객체와 관계형 데이터베이스 간의 패러다임 불일치 문제를 해결
- 하지만 객체를 테이블에 맞추어 모델링 한다면 **객체 간의 협력 관계**를 만들 수 없음 (테이블은 외래 키를 참조하여 연관 테이블을 찾지만 객체는 참조를 사용하여 연관 객체를 찾음)
- **객체 지향 모델링**을 위해 **연관 관계**를 매핑
- 양방향 연관 관계를 가지는 경우, 객체는 서로 참조를 해야 하지만 테이블은 조인으로 해결
- 때문에 엔티티 사이의 **연관 관계 주인**을 결정하여 외래 키를 관리 (연관 관계의 주인은 외래 키가 있는 곳)

## ❖ 엔티티 모델링 기본 전략

- 엔티티 연관 관계 및 연관 관계 주인을 정확하게 매핑
- 다대다 관계는 중간 엔티티를 추가하여 **일대다**, **다대일 관계**로 확장
- **연관 관계 편의 메서드**를 작성하여 테이블 변경 정보를 객체에 즉시 적용
- 모든 다대일, 일대일 관계의 객체는 기본적으로 **지연 로딩(Lazy Loading)**

## ❖ 엔티티 조회 성능 최적화

- 다대일, 일대일 관계의 객체를 함께 조회하기 위해 **페치 조인(Fetch Join)**을 활용 → **N + 1 Select 문제 해결**
- 일대다 관계의 컬렉션 객체는 직접 페치 조인 하지 않고, **Batch-fetch-size** 를 설정하여 지정 크기만큼 한번에 조회

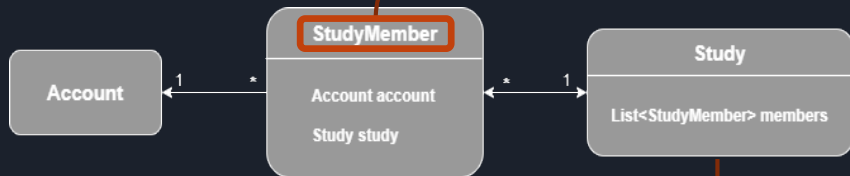
```
spring.jpa.properties.hibernate.default_batch_fetch_size = 1000
```

※ 컬렉션을 페치 조인하는 경우, 페이징이 불가하며 둘 이상의 컬렉션 페치 조인은 부정합 데이터를 초래할 수 있음

[예시]

## Account 와 Study

### 엔티티의 연관 관계



```
72 // 연관 관계 편의 메서드 //
```

```
73
```

```
74 public void addStudyMember(StudyMember studyMember) {
```

```
75     memberCount++;
```

```
76     this.members.add(studyMember);
```

```
77     studyMember.setStudy(this);
```

```
78 }
```

```
79
```

```
80 public void removeStudyMember(Account account) {
```

```
81     memberCount--;
```

```
82     this.members.removeIf(studyMember -> studyMember.getAccount().equals(account));
```

```
83 }
```

```
8 @Entity
```

```
9 @Table(name = "study_member")
```

```
10 @Getter @Setter @EqualsAndHashCode(of = "id")
```

```
11 @Builder @AllArgsConstructor @NoArgsConstructor
```

```
12 public class StudyMember {
```

```
13
```

```
14     @Id @GeneratedValue
```

```
15     @Column(name = "study_member_id")
```

```
16     private Long id;
```

```
17
```

```
18     @ManyToOne(fetch = FetchType.LAZY)
```

```
19     @JoinColumn(name = "study_id")
```

```
20     private Study study;
```

```
21
```

```
22     @ManyToOne(fetch = FetchType.LAZY)
```

```
23     @JoinColumn(name = "account_id")
```

```
24     private Account account;
```

```
25
```

```
26     @Enumerated(EnumType.STRING)
```

```
27     private ManagementLevel managementLevel;
```

```
28
```

```
29     // 생성 메서드 //
```

```
30
```

```
31     public static StudyMember createStudyMember(Study study, Account account, ManagementLevel managementLevel) {
```

```
32         return StudyMember.builder()
```

```
33             .study(study)
```

```
34             .account(account)
```

```
35             .managementLevel(managementLevel)
```

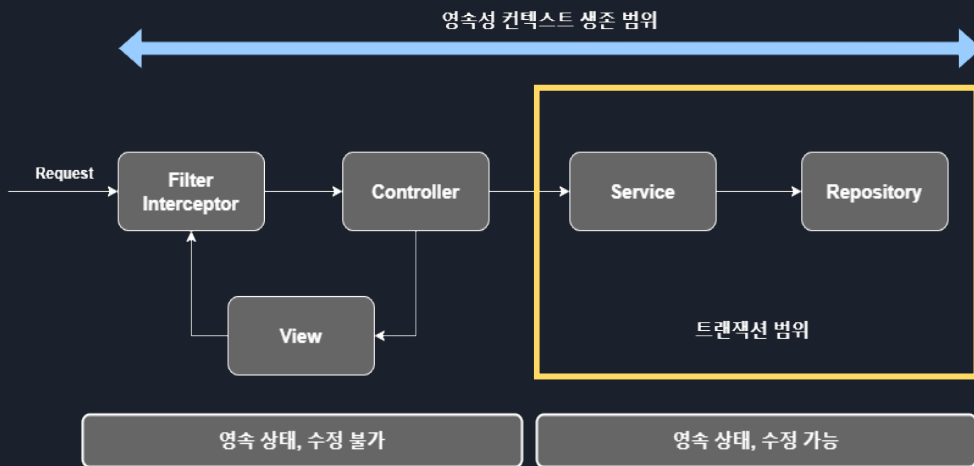
```
36             .build();
```

```
37     }
```

```
38 }
```

## ❖ OSIV (Open Session In View)

- JPA(+Hibernate) 에서의 모든 엔티티들은 엔티티 매니저를 통해 **영속성 컨텍스트** 내부에서 관리됨
- 영속성 컨텍스트가 생존하는 동안 기본적으로 **데이터베이스 커넥션**을 유지
- **OSIV** 전략은 트랜잭션 시작과 같이 최초로 데이터베이스 커넥션 리소스를 사용하는 시점부터 요청에 대한 응답이 완료될 때까지 영속성 컨텍스트와 데이터베이스 커넥션을 유지



### < 엔티티 수정 >

- 트랜잭션 커밋 시점에 **변경 감지**
- OSIV 전략에 의해 컨트롤러에서 조회한 엔티티는 서비스에게 전달되더라도 계속 **영속 상태를 유지**
- 반대로 병합은 모든 속성을 변경하며, 값이 없다면 null 로 업데이트하는 위험 발생

## ❖ OSIV 와 성능 최적화

- OSIV 전략은 기본적으로 **True**

```
spring.jpa.open-in-view = true
```

- OSIV 전략을 사용한다면 트랜잭션이 종료된 후에도 영속성 컨텍스트와 데이터베이스 커넥션이 유지되기 때문에 뷰 또는 컨트롤러 안에서 **지연 로딩** 사용 가능
- 하지만 특정 요청이 데이터베이스 커넥션 리소스를 오랫동안 점유하기 때문에 **실시간 트래픽 처리**가 중요한 애플리케이션에서는 **리소스 고갈** 상태로 인한 장애 발생
- OSIV 전략을 사용하지 않고, 영속성 컨텍스트 종료 후 데이터베이스 커넥션을 함께 반환하여 리소스 낭비를 방지  
※ 단, 모든 지연 로딩을 트랜잭션 안에서 처리해야 함
- 결과적으로 애플리케이션의 **서비스 특징**에 따라 OSIV 전략을 활용하여 **성능 최적화** 가능
- 본 프로젝트는 OSIV 기본 전략을 따름

## ❖ 인프라 (1) : 권한 및 사용자 관리 – Spring Security

- 사용자 접근 제어 및 권한 설정

예 1) 권한이 필요하지 않은 HTTP 요청 설정 (회원가입, 로그인 등)

예 2) NPM 리소스 요청에 대한 필터링 생략

- 사용자 인증(Authentication) 및 애플리케이션 내의 인증된 사용자에게 대한 권한 부여(Authorization)
- 세션-쿠키 방식의 인증 절차

- ① 사용자의 로그인 요청
- ② 사용자가 입력한 로그인 정보와 데이터베이스의 사용자 정보가 일치하는지 확인
- ③ 인증에 성공한 경우 해당 사용자의 세션 정보를 메모리에 저장하고, 세션 ID 와 함께 응답
- ④ 이후 요청에서는 쿠키에 포함된 세션 ID 의 유효성을 검증



- **UserDetailsService** 인터페이스 (데이터베이스의 사용자 정보 참조)

#### 메서드 구현

```
54      @Override
55      # public UserDetails loadUserByUsername(String emailOrUsername) throws UsernameNotFoundException {
56
57          // Authentication administration //
58
59          Account account = accountRepository.findByEmail(emailOrUsername);
60
61          if (account == null) {
62              account = accountRepository.findByNickname(emailOrUsername);
63          }
64
65          if (account == null) {
66              throw new UsernameNotFoundException(emailOrUsername);
67          }
68
69          return new UserAccount(account);
70      }
```

#### 사용자 정보

```
10      @Getter
11      public class UserAccount extends User {
12
13          private Account account;
14
15          @ public UserAccount(Account account) {
16              super(account.getNickname(), account.getPassword(),
17                  Collections.singletonList(new SimpleGrantedAuthority( role: "ROLE_USER")));
18
19              this.account = account;
20          }
21      }
```

- ✓ 리턴하는 **UserDetails** 인터페이스 타입의 객체를 **사용자 정보**로 인식
- ✓ 이후 사용자가 입력한 로그인 정보와 비교

#### < Spring Security 의 로그인 >

SecurityContext 에  
Authentication(Token) 이 존재하는가 ?



인증에 성공한 사용자 정보를 담아  
생성된 **Authentication(Token)** 을 내부  
메모리(**SecurityContext**)에 담아 관리

## ❖ 인프라 [2] : 메일 서비스 – Gmail SMTP

- **JavaMailSender** 기반, **Html** 형태의 이메일 전송

※ 사용자 인증 이메일, 로그인 이메일 링크

- 메일 서비스 **추상화** → 객체 생성 및 전달 (수신 메일 주소, 제목, 내용)
- 구글 Gmail SMTP 서버 설정

```
spring.mail.host = smtp.gmail.com
spring.mail.port = 587
spring.mail.username = #이메일 주소#
spring.mail.password = #발급된 앱 비밀번호#

spring.mail.properties.mail.smtp.auth = true
spring.mail.properties.mail.smtp.timeout = 5000
spring.mail.properties.mail.smtp.starttls.enable = true
```

### 보안 → 2단계 인증

생성된 앱 비밀번호

기기용 앱 비밀번호

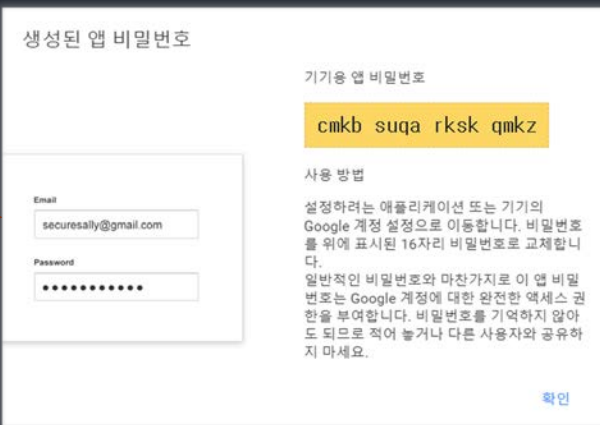
cmkb suqa rskq qmkz

사용 방법

설정하려는 애플리케이션 또는 기기의 Google 계정 설정으로 이동합니다. 비밀번호를 위에 표시된 16자리 비밀번호로 교체합니다.

일반적인 비밀번호와 마찬가지로 이 앱 비밀번호는 Google 계정에 대한 완전한 액세스 권한을 부여합니다. 비밀번호를 기억하지 않아도 되므로 적어 놓거나 다른 사용자와 공유하지 마세요.

확인



## ❖ 인프라 (3) : 리포지토리 지원 – Querydsl4RepositorySupport

- **QuerydslRepositorySupport** : Spring Data JPA 가 제공하는 Querydsl 지원 클래스
- 직접 클래스를 만들어 기존 라이브러리의 **한계점** 극복 → **Querydsl4RepositorySupport**
  - ✓ Querydsl 4.x 버전에 맞는 지원 라이브러리
  - ✓ 페이징 및 페이징 성능 개선을 편리하게 변환
  - ✓ 카운트 쿼리를 분리하여 커스터마이징 가능
  - ✓ Sort 기능 정상 작동
  - ✓ select() 또는 selectFrom() 으로 시작 가능
  - ✓ QueryFactory 제공

```
78     protected <T> JPAQuery<T> select(Expression<T> expr) { return getQueryFactory().select(expr); }
81
82     protected <T> JPAQuery<T> selectFrom(EntityPath<T> from) { return getQueryFactory().selectFrom(from); }
85
86     @
87     protected <T> Page<T> applyPagination(Pageable pageable, Function<JPAQueryFactory, JPAQuery> contentQuery) {
88         JPAQuery jpaQuery = contentQuery.apply(getQueryFactory());
89         List<T> content = getQuerydsl().applyPagination(pageable, jpaQuery).fetch();
90
91         return PageableExecutionUtils.getPage(content, pageable, jpaQuery::fetchCount);
92     }
93
94     @
95     protected <T> Page<T> applyPagination(Pageable pageable, Function<JPAQueryFactory, JPAQuery> contentQuery, Function<JPAQueryFactory, JPAQuery> countQuery) {
96         JPAQuery jpaContentQuery = contentQuery.apply(getQueryFactory());
97         JPAQuery countResult = countQuery.apply(getQueryFactory());
98         List<T> content = getQuerydsl().applyPagination(pageable, jpaContentQuery).fetch();
99
100         return PageableExecutionUtils.getPage(content, pageable, countResult::fetchCount);
101     }
```

Process & Functions

## 동작 과정 및 상세 기능

### 가입 및 로그인

- 회원 가입
- 로그인 (전/후)

### 스터디 관리

- 스터디 만들기
- 스터디 설정

### 스터디 모임 참여

- 스터디 가입
- 모임 참가 신청

### 계정

- 프로필 조회
- 계정 설정

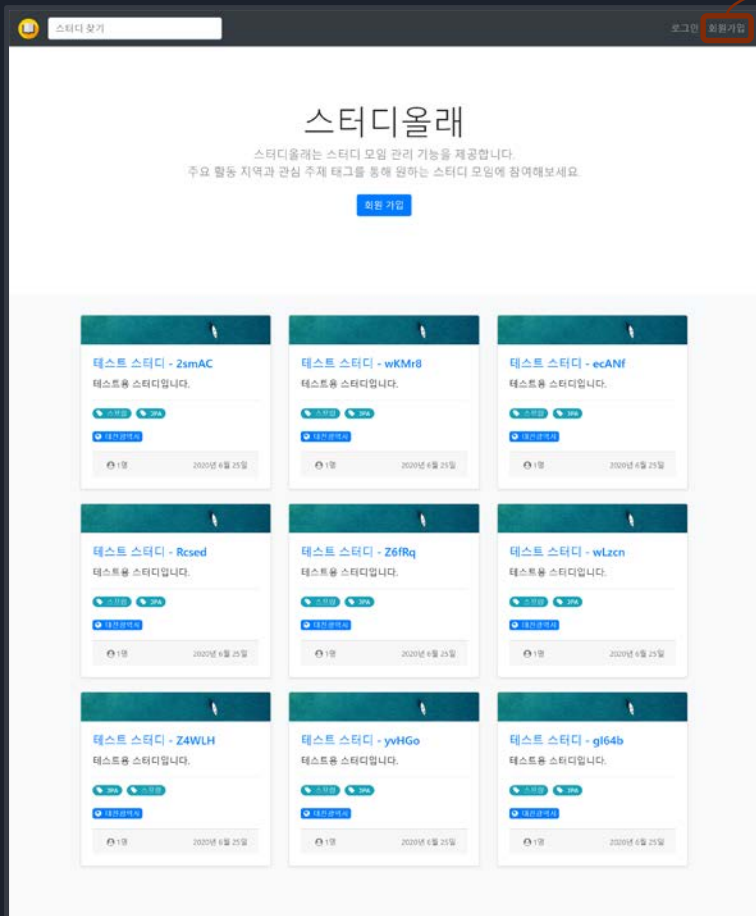
### 모임 관리

- 모임 만들기
- 모임 참가 신청 관리

### 알림 및 기타

- 알림 조회
- 스터디 검색 결과

## 로그인 전 홈 화면



### 회원가입

닉네임  
공백없이 문자와 숫자로만 이루어진 형식이며 3자 이상 20자 이내로 입력하세요.

이메일  
your@email.com  
스터디올래는 사용자의 이메일을 공개하지 않습니다.

패스워드  
8자 이상 50자 이내로 입력하세요. 공백을 제외하고 영문자, 숫자, 특수기호를 사용할 수 있습니다.

약관에 동의하시면 가입하기 버튼을 클릭하세요.

가입 완료

### • 로그인 전 첫 페이지

- ✓ 최근에 열린 9개의 스터디 조회

### • 회원 가입

- ✓ **패스워드 인코딩** 필수
- ✓ 가입 완료 후 자동 로그인
- ✓ 이메일 인증을 위한 토큰 생성
- ✓ 서비스 이용을 위한 계정 인증 이메일 전송

## • 계정 인증 이메일

- ✓ 링크를 클릭하거나 URL 입력을 통해 회원 가입 완료
- ✓ URL 의 토큰 값과 해당 계정의 토큰 값을 비교
- ✓ 계정 인증 이메일 재전송은 1시간에 한번 가능

☆ [스터디올래] 회원 가입 완료를 위한 계정 인증 메일입니다. 📧

보낸사람 VIP <tidwp3s@gmail.com>

안녕하세요. tidwp3s 님

스터디올래 서비스를 사용하기 위해 아래 링크를 클릭하여 인증을 완료해주세요.

[이메일 인증하기](#)

링크가 동작하지 않는 경우, 아래 URL을 웹 브라우저에 복사해서 붙여 넣으세요.

<http://localhost:8080/account/check-email-token?token=3f31d062-04ef-4839-b4ba-4704041edd4d&email=tidwp3s@naver.com>

스터디올래 2020

## 로그인

이메일 (또는 닉네임)

가입할 때 사용한 이메일 또는 닉네임을 입력하세요.

패스워드

패스워드가 기억나지 않는다면, [패스워드 없이 로그인하기](#)

☐ 로그인 유지

스터디올래에 처음 오셨다면, [계정을 먼저 만드세요.](#)

로그인

## • 로그인

- ✓ 입력된 로그인 정보 확인 (Spring Security)
- ✓ **로그인 유지** 기능

## • 패스워드 없이 로그인

- ✓ 이미 계정 인증이 완료된 유효 이메일
- ✓ [이메일 로그인 링크](#) 전송
- ✓ 계정 인증과 동일하게 1시간에 한번 가능

## ❖ 로그인 기억하기

- 세션이 만료되더라도 **로그인 상태 유지**
- **쿠키**에 인증 정보를 남겨두고 세션이 만료됐을 때, 쿠키의 **인증 정보**를 참조
- **Improved Persistent Login**

- 인증 정보 : Username, 토큰(랜덤, 매번 변경), 시리즈(랜덤, 고정)
- 쿠키를 탈취당한 경우 희생자는 유효하지 않은 토큰과 유효한 시리즈로 접속을 하게 되는데, 이때 모든 토큰을 삭제하여 탈취된 쿠키를 더 이상 사용하지 못하도록 방지

### 영속화 기반의 Spring Security 설정

```
46 http.rememberMe()  
47     .userService(userDetailsService)    // User details service  
48     .tokenRepository(tokenRepository()); // Token in DB
```

```
61 @Bean  
62 public PersistentTokenRepository tokenRepository() {  
63     JdbcTokenRepositoryImpl jdbcTokenRepository = new JdbcTokenRepositoryImpl();  
64     jdbcTokenRepository.setDataSource(dataSource);  
65  
66     return jdbcTokenRepository;  
67 }
```

✓ 사용자 인증 정보 테이블  
(Persistent\_logins)

✓ 현재 인증 사용자 정보  
(UserDetailsService)

✓ 토큰 리포지토리

## 로그인 후 홈 화면

스터디 찾기

스터디올래에서 나에게 맞는 스터디를 찾아보세요.  
관심있는 스터디의 주제나 활동 지역을 입력해보세요.

로그아웃  
프로필  
스터디  
설정  
로그아웃

내 정보

참석 예정 모임

테스트 모임 - 1

테스트 스터디 - NcDjb

내일 오전 11:27 15시간 후

자세히 보기

최근 완료 모임

완료된 모임이 없습니다.

추천 스터디 (주요 활동의 지역의 관심 주제 스터디)

테스트 스터디 - NcDjb

테스트용 스터디입니다.

소셜 30%

대전광역시

2명 2020년 7월 12일

테스트 스터디 - EERH2

테스트용 스터디입니다.

소셜 30%

대전광역시

1명 2020년 7월 12일

테스트 스터디 - 9HRjn

테스트용 스터디입니다.

소셜 30%

대전광역시

1명 2020년 7월 12일

© 스터디올래 2020.

### 스터디 검색

- ✓ 입력한 **키워드**를 포함한 스터디 검색
- ✓ 이름, 주제, 활동 지역

### 내 정보 확인

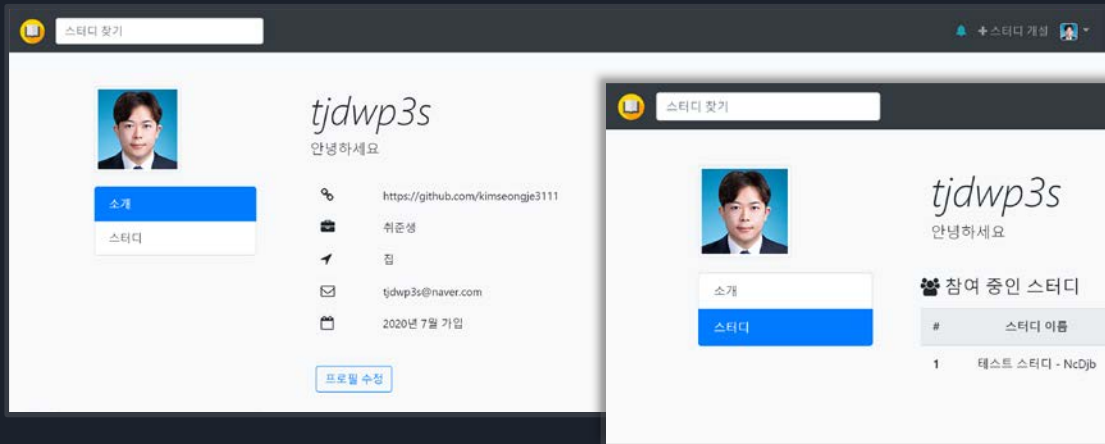
- ✓ 관심 스터디 주제, 주요 활동 지역
- ✓ 참여 중인 스터디 목록
- ✓ 화면 좌측에 사이드 바 형태로 표시 (mmenu.js)

### 참석 예정 모임 / 최근 완료 모임

### 추천 스터디 제공

- ✓ 나의 주요 활동 **지역**에 오픈된 **관심 주제**의 스터디 목록
- ✓ 최근에 공개된 날짜 순
- ✓ 슬라이드 뷰 (swiper.js)





## • 소개 (프로필 정보)

- ✓ 링크
- ✓ 메일 주소
- ✓ 직업
- ✓ 가입 날짜
- ✓ 활동 지역

## • 내 스타디

- ✓ 참여 중인 스타디 목록
- ✓ 종료된 스타디 목록

- 현재 사용자가 프로필을 수정할 수 있는 권한이 있는가?
- 현재 인증 사용자 정보 참조

- 뷰 : Thymeleaf 의 Spring Security 지원
- 컨트롤러 : 커스텀 애노테이션 설정



### Thymeleaf 의 Spring Security 지원

```
<div sec:authorize="isAuthenticated()"></div>
<div th:text="{#authentication.name}"></div>
```

### 커스텀 애노테이션 : @CurrentUser

```
10 @Retention(RetentionPolicy.RUNTIME)
11 @Target(ElementType.PARAMETER)
12 @AuthenticationPrincipal(expression = "#this == 'anonymousUser' ? null : account")
13 public interface CurrentUser {
14
15     // Custom annotation //
16
17     // 스프링 시큐리티에 의해 인증되지 않은 사용자 : null //
18     // 관리미더로 적용, 전달일까지 유지 //
19     // @AuthenticationPrincipal : SpEL을 사용해 Principal 내부 정보 접근 //
20 }
```

# 계정 설정

**프로필**

패스워드 변경

새 패스워드

최저 8자 이상 50자 이내로 입력하세요. 공백을 제외하고 영문자, 숫자, 특수기호를 사용할 수 있습니다.  
현재의 패스워드와 새 패스워드를 입력하세요.

새 패스워드 확인

새 패스워드 확인을 위해 한번 더 입력해주세요.

패스워드 변경

**프로필**

알림 설정

주요 활동 지역에 관심있는 주제의 스터디가 만들어졌을 때, 알림을 받을 방법을 설정하세요.

이메일로 받기 ☒ 웹으로 받기

참여 중인 스터디에 대한 알림을 받을 방법을 설정하세요.

이메일로 받기 ☒ 웹으로 받기

스터디 오프닝 신청 및 결과에 대한 알림을 받을 방법을 설정하세요.

이메일로 받기 ☒ 웹으로 받기

저장하기

**프로필**

닉네임

닉네임을 변경하면 프로필 페이지 링크도 바뀝니다!

tjdw3s

공백없이 문자와 숫자로만 이루어진 형식이며 3자 이상 21자 이내로 입력하세요.  
닉네임 변경은 하루에 한번만 가능합니다.

닉네임 변경

계정 삭제

이 계정을 삭제할 수 없습니다.

계정 삭제

스터디 찾기

**프로필**

패스워드

알림

관심 주제

활동 지역

계정

tjdw3s

한 줄 소개

안녕하세요.

필지 약게 35자 이내로 입력해 주세요.

링크

https://github.com/kimseongje3111

블로그, 유튜브 또는 포털블로그 등 문단을 표현할 수 있는 링크를 추가해 보세요.

직업

취준생

개발자? 디자이너? 취준생? 대학생?

활동 지역

집

주요 활동(사는 곳이나 직장을 다니는 곳 또는 놀러 다니는 곳) 지역의 도시 이름을 알려주세요.

프로필 수정

내 프로필 이미지

변경 이미지 선택

Browse

## • 프로필 설정

- ✓ 한 줄 소개, 링크, 직업, 활동 지역
- ✓ 프로필 이미지 편집 및 등록 (Cropper.js)
- ✓ 등록된 프로필 이미지가 없다면 닉네임에 따른 기본 이미지 표시 (Jdenticon.js)

## • 알림 설정

- ✓ 추천 스터디 알림
- ✓ 참여 중인 스터디 알림
- ✓ 모임 신청 및 결과 알림

## • 계정 삭제

## • 닉네임 / 패스워드 변경

프로필

태스워드

알림

관심 주제

활동 지역

계정

## 주요 활동 지역

주요 활동 지역을 태그로 등록해보세요.

해당 지역에 스티커가 등록되면 알림을 받을 수 있습니다.

시스템에 등록된 지역만 선택할 수 있습니다. 입력 장을 클릭하여 확인하세요.

Gangneung (강릉시) / Gangwon

Geoje (거제시) / South Gyeongsang

Gyeongsan (경산시) / North Gyeongsang

Gyeongju (경주시) / North Gyeongsang

Gyeong (계룡시) / South Chungcheong

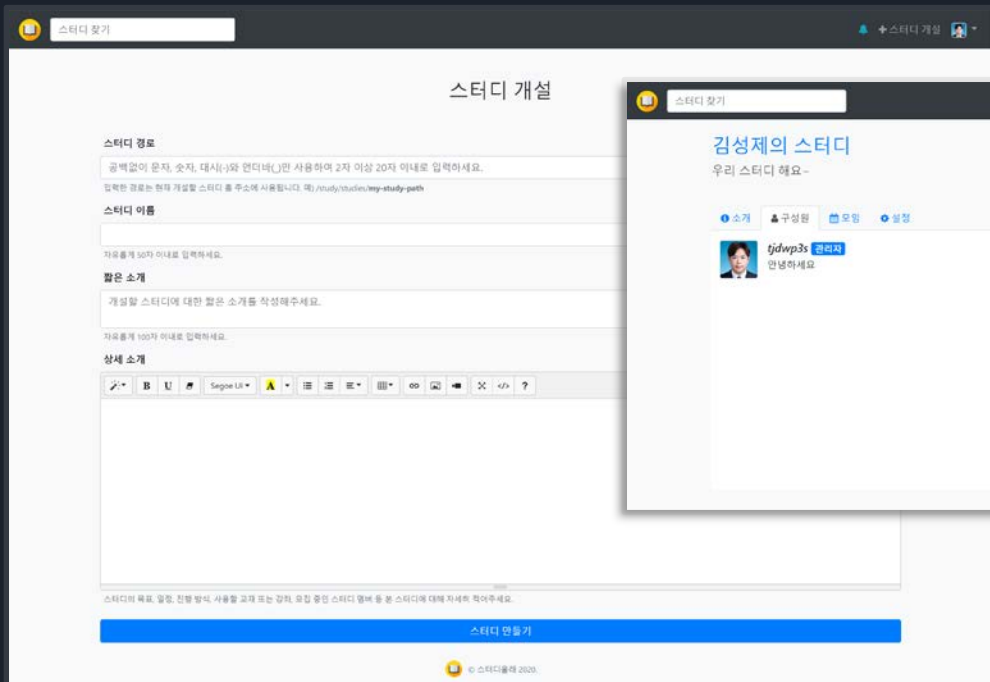
Goyang (고양시) / Gyeonggi

Gongju (공주시) / South Chungcheong

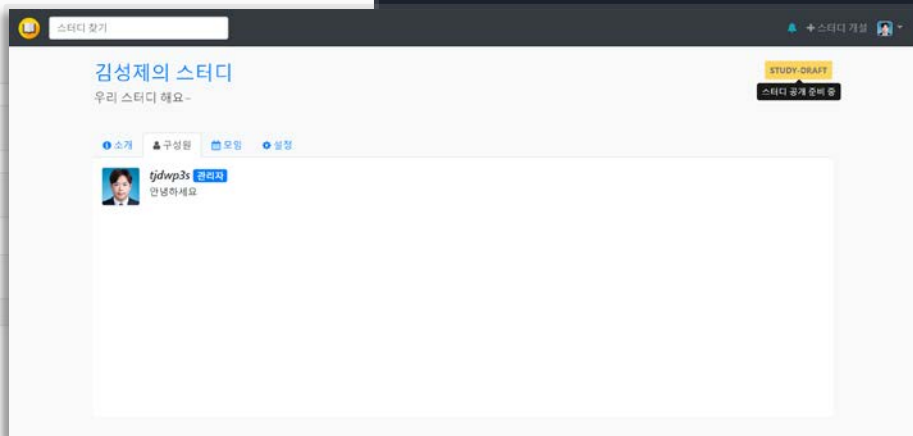
Gwacheon (과천시) / Gyeonggi

Gwanmugang (관매동) / Gyeonggi

# 스터디 만들기



## 스터디 홈



### • 스터디 개설

- ✓ 경로, 이름, 짧은 소개, 상세 소개
- ✓ 에디터 (Summernote.js)
- ✓ 스터디 상태 : 공개 전

### • 스터디 홈 메뉴

- ✓ 소개, 구성원, 모임 목록, 설정 (관리자)

## 스터디 자동 삭제



STUDY-OPEN APPLY-ON  
 + 모임 만들기  
 팀원 모집 중

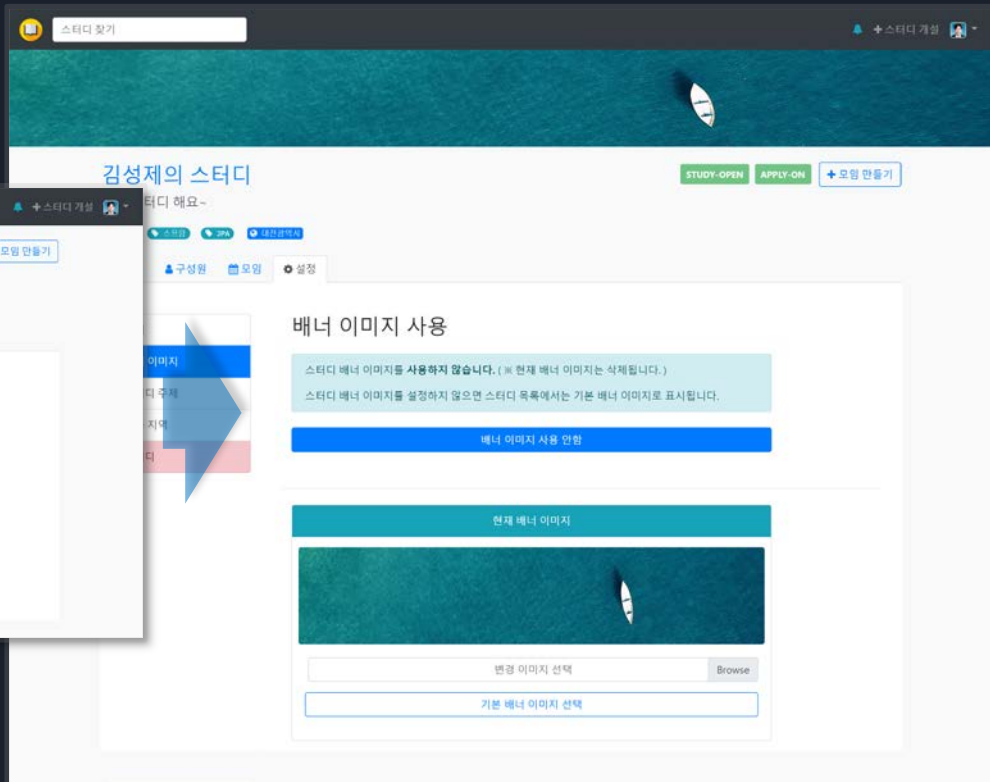
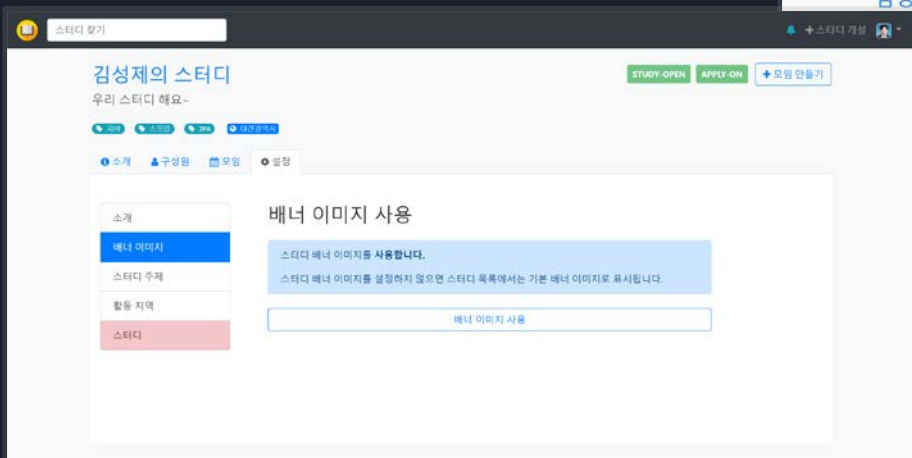
스터디 상태 표시,  
모임 생성 가능

- 스터디 공개 및 종료
- 스터디 팀원 모집
- 스터디 이름 / 경로 변경
- 스터디 삭제

# 스터디

## 배너 이미지

### 배너 이미지 사용



### 스터디 배너

- ✓ 스터디 홈 화면 상단에 표시
- ✓ 배너 이미지 편집 및 등록 / 기본 배너 이미지 제공
- ✓ 이미지를 **DataUrl** 형태로 변환하여 저장
  - 의존 라이브러리 및 빈 설정 (`eu.maxschuster:dataurl`)

## 모임 만들기

**김성재의 스티디 / 새 모임 만들기**

## 새 모임 만들기

**모임 이름**  
  
(자유롭게 50자 이내로 입력하세요.)

**모집 방법**  

선택순

2

모집 방법은 두 가지입니다.  
 선택순 : 모집 인원 내의 모임 신청은 자동으로 확정됩니다. 제한 인원을 초과한 경우 모집 대가 상대가 되어 이후에 확정자 중 취소자가 발생하면 선택순으로 다음 대기자가 확정됩니다. 단, 등록 마감일 이후에는 신청을 취소해도 확정 여부가 바뀌지 않습니다.  
 관례화 확인 : 모임 및 스티디 관리자가 모임 신청 목록을 조회하고, 직접 확정 여부를 결정할 수 있습니다. 단, 등록 마감일 이후에는 변경할 수 없습니다.

최대 수용 가능한 모임 참석 인원을 설정하세요. 최소 2인 이상이어야 합니다.

**모임 시작 일시**

**모임 종료 일시**

**등록 마감 일시**

모임 시작 일시를 입력하세요. 상세한 모임 일정은 본문에 적어주세요. | 모임 종료 일시가 지나면 해당은 자동

**모임 끝낼**

7월 2020														
일	월	화	수	목	금	토								
28	29	30	1	2	3	4								
5	6	7	8	9	10	11	03	:	43	<input type="button" value="확인"/>				
12	13	14	15	16	17	18								
19	20	21	22	23	24	25								
26	27	28	29	30	31	1								
2	3	4	5	6	7	8								

모임에서 다루는 주제, 장소, 진행 방식 등을 자세히 적어 주세요.

**모임 만들기**

- 새 모임 만들기

- ✓ 공개된 스터디의 관리자
  - ✓ 이름, 모집 방법, 모집 인원, 설명
  - ✓ 모임 시작 / 종료 일시, 등록 마감 일시
- (Tempusdominus.js, Moment.js)

- 모집 방법

- ✓ **선착순** : 모집 인원 내 선착순 모집
- ✓ **관리자 확인** : 모임 신청 직접 관리
- ✓ 등록 마감 일 이후 확정 여부 변경 불가
- ✓ 모임 참가 신청이 아직 확정되지 않은 경우 대기 상태

## 모임 관리

### 모임 정보 - 관리자 확인, 참가 신청 마감 전

테스트 스터디 - NcDjb / 테스트 모임 - 2

APPLY-ON | 모임 수정 | 모임 취소

참가자 모임 홈

테스트 모임 - 2

모임 참가 신청 현황 (2)

#	사용자	참가 신청 날짜	접수 상태	참가 신청 관리	출석 관리
1	admin	2020년 7월 12일 오후 7:29	확정		
2	tydxp3s	2020년 7월 12일 오후 7:51	대기중	신청 수락	

모임 정보

모임 이름: 관리자 확인

모임 인원: 5명

참가 신청 마감: 2020년 7월 16일 목요일 오전 10:29

모임 시작: 2020년 7월 18일 토요일 오전 10:29

모임 종료: 2020년 7월 19일 일요일 오전 10:30

모임 관리자: admin

### 모임 정보 - 선착순, 모임 시작 후

테스트 스터디 - NcDjb / 테스트 모임 - 1

APPLY-OFF | 모임 수정 | 모임 취소

테스트 모임 - 1

모임 참가 신청 현황 (2)

#	사용자	참가 신청 날짜	접수 상태	출석 관리
1	admin	2020년 7월 12일 오후 7:28	확정	취소
2	tydxp3s	2020년 7월 12일 오후 7:32	확정	재확인

모임 정보

모임 이름: 선착순

모임 인원: 5명

참가 신청 마감: 2020년 7월 13일 일요일 오전 3:27

모임 시작: 2020년 7월 13일 일요일 오전 11:27

모임 종료: 2020년 7월 14일 화요일 오전 10:28

모임 관리자: admin

#### 모임 수정 / 취소

- ✓ 모집 방법 : **변경 불가**
- ✓ 모집 인원 변경 : 이미 확정된 인원 수 이상

#### 모임 참가 신청 현황

- ✓ 공통 : 사용자, 참가 신청 날짜, 접수 상태
- ✓ 참가 신청 날짜 순

#### 관리자 확인 모임

- ✓ 참가 신청 관리 (참가 신청 마감 일 이전)
- ✓ 모집 인원 내 **신청 수락** 또는 **취소**

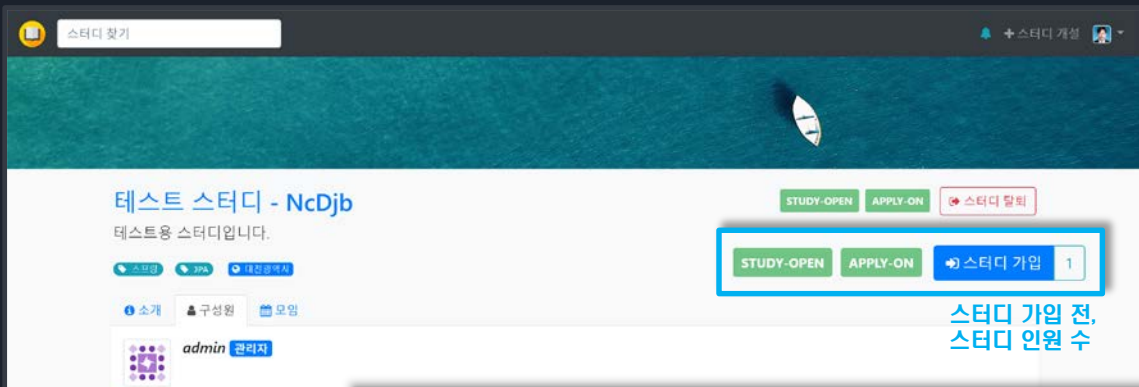
#### 선착순 모임

- ✓ 모집 인원 내 선착순 모집
- ✓ 취소자가 발생했거나 모집 인원을 늘린 경우  
선착순으로 대기 중인 인원 자동 확정

#### 출석 관리



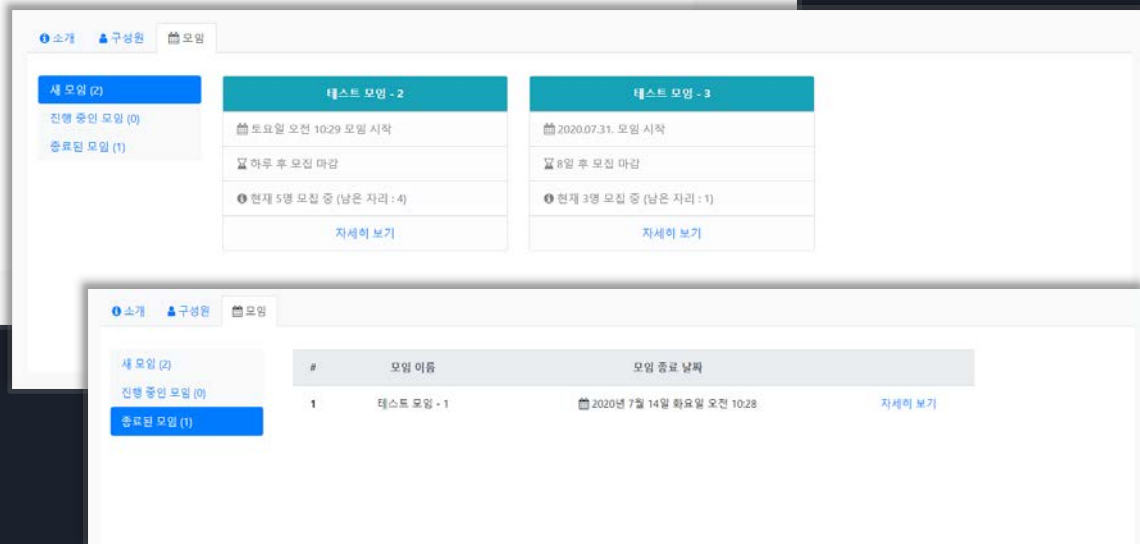
# 스터디 가입



## 스터디 홈 화면

- ✓ 관리 권한이 없는 일반 사용자
- ✓ 스터디 가입 / 탈퇴
- ✓ 스터디 설정 불가

## 모임 목록



모임

참가 신청

스터디 찾기

스터디 - NcDjb / 테스트 모임 - 3

APPLY-ON

참가 신청

테스트 모임 - 3

모임 참가 신청 현황 (1)

#	사용자	참가 신청 날짜	접수 상태	출석
1	admin	2020년 7월 13일 오후 10:50	확정	모임 시작 전

모임 정보

모임 방법  
선착순

모임 인원  
3명

참가 신청 마감  
2020년 7월 23일 목요일 오후 1:49

모임 시작  
2020년 7월 31일 금요일 오후 1:49

모임 종료  
2020년 8월 1일 토요일 오후 1:50

모임 관리자  
admin

스터디 찾기

스터디 - NcDjb / 테스트 모임 - 3

APPLY-ON

참가 신청

테스트 모임 - 3

해당 모임에 참가하시겠습니까?  
일정을 캘린더에 등록해 두시면 좋습니다.  
확인을 눌러 모임 참가 신청을 완료해주세요.

※ 모집 방법에 따른 참가 신청 처리 안내

선착순: 모집 인원 내의 모임 신청은 자동으로 확정됩니다. 제한 인원을 초과한 경우 모집 대기 상태가 되어, 이후에 확정자 중 취소자가 발생하면 선착순으로 다음 대기자가 확정됩니다.

관리자 확인: 모임 및 스터디 관리자가 모임 신청 목록을 조회하고, 직접 확정 여부를 결정합니다.

닫기

확인

테스트 모임 - 3

모임 참가 신청 현황 (1)

#	사용자	참가 신청 날짜	접수 상태	출석
1	admin	2020년 7월 13일 오후 10:50	확정	모임 시작 전

모임 정보

모임 방법  
선착순

모임 인원  
3명

참가 신청 마감  
2020년 7월 23일 목요일 오후 1:49

모임 시작  
2020년 7월 31일 금요일 오후 1:49

모임 종료  
2020년 8월 1일 토요일 오후 1:50

모임 관리자  
admin

## 알림 조회

The image displays two screenshots of a web application interface for managing notifications. The interface is divided into two main sections: '읽은 알림' (Read Notifications) and '읽지 않은 알림' (Unread Notifications).

**Left Screenshot (Read Notifications):**

- Header:** '스터디 찾기' (Find Study) search bar.
- Left Sidebar:**
  - 읽지 않은 알림 (Unread Notifications): 0
  - 읽은 알림 (Read Notifications): 8
  - 모임 신청 및 결과 알림 (Meeting Application and Result Notification): 4
  - 참여 중인 스터디 알림 (Participating Study Notification): 2
  - 읽은 알림 삭제 (Delete Read Notifications): 1
- Main Content:**
  - 모임 신청 및 결과 관련 소식이 있습니다.** (There are notifications related to meeting applications and results.)
    - 테스트 스터디 - NoDj: [모임 신청/결과] '테스트 모임 - 3' 모임 신청이 확정되었습니다. 모임 일정을 확인하세요.
    - 테스트 스터디 - NoDj: [모임 신청/결과] '테스트 모임 - 2' 모임 신청이 확정되었습니다. 모임 일정을 확인하세요.
    - 테스트 스터디 - NoDj: [모임 신청/결과] '테스트 모임 - 2' 모임 신청이 거절되었습니다. 모임 관리자에게 문의하세요.
    - 테스트 스터디 - NoDj: [모임 신청/결과] '테스트 모임 - 2' 모임 신청이 확정되었습니다. 모임 일정을 확인하세요.
    - 테스트 스터디 - NoDj: [모임 신청/결과] '테스트 모임 - 2' 모임 신청을 완료하였습니다. 확정 시 알림이 전송됩니다.
    - 테스트 스터디 - NoDj: [모임 신청/결과] '테스트 모임 - 1' 모임 신청이 확정되었습니다. 모임 일정을 확인하세요.
  - 참여 중인 스터디 관련 소식이 있습니다.** (There are notifications related to studies you are participating in.)
    - 테스트 스터디 - NoDj: [새 모임] '테스트 모임 - 3' 모임이 새로 등록되었습니다. 모임에 참여해보세요.
    - 테스트 스터디 - NoDj: [스터디 가입/탈퇴] 스터디에 가입하였습니다. 이제부터 해당 스터디에서 활동할 수 있습니다.

**Right Screenshot (Unread Notifications):**

- Header:** '스터디 찾기' (Find Study) search bar.
- Left Sidebar:**
  - 읽지 않은 알림 (Unread Notifications): 4
  - 읽은 알림 (Read Notifications): 8
  - 새 스터디 알림 (New Study Notification): 1
  - 모임 신청 및 결과 알림 (Meeting Application and Result Notification): 1
  - 참여 중인 스터디 알림 (Participating Study Notification): 2
  - 모두 읽음으로 표시 (Mark All as Read): 1
- Main Content:**
  - 주요 활동 지역에 관심있는 주제의 스터디가 새로 오픈하였습니다.** (New studies have opened in the main activity area that you are interested in.)
    - 김성재의 스터디: [새 스터디 오픈] 새로 오픈한 스터디 중 나에게 맞는 스터디를 확인해보세요. (2분 전)
  - 모임 신청 및 결과 관련 소식이 있습니다.** (There are notifications related to meeting applications and results.)
    - 테스트 스터디 - NoDj: [모임 신청 취소] '테스트 모임 - 2' 모임 신청을 취소하였습니다. (4분 전)
  - 참여 중인 스터디 관련 소식이 있습니다.** (There are notifications related to studies you are participating in.)
    - 내 스터디: 김성재의 스터디: [스터디 변경 사항] 이제부터 팀원 모집을 시작합니다. 팀원 모집 상태는 1시간에 한번만 변경할 수 있습니다. (1분 전)
    - 내 스터디: 김성재의 스터디: [새 스터디 오픈] 스터디를 공개하였습니다. 이제 팀원을 모집해보세요. (2분 전)

### • 읽은 알림

- ✓ 읽은 알림 일괄 삭제 버튼
- ✓ 한 달이 지난 알림은 자동 삭제

### • 읽지 않은 알림

- ✓ 읽지 않은 알림이 있는 경우 상단 **알림 표시**
- ✓ 모두 읽음으로 표시 버튼

## ❖ 알림 전송

- 애플리케이션의 서비스 처리 흐름에 영향을 주지 않고, 해당 작업을 **비동기 처리** → **스레드 풀**
- 처리 과정

- ① 특정 서비스의 알림 전송이 필요한 메서드 내에서 EventPublisher에 의한 애플리케이션 이벤트 발생 → 해당 알림에 대응하는 객체 생성 이벤트
- ② 각 이벤트에 맞는 EventListener가 동작하여 알림을 생성하고, 저장 (트랜잭션 내) 또는 알림 메일 전송
- ③ 해당 작업을 스레드 풀 내에서 비동기적으로 처리 → **서비스 처리 흐름**과 무관

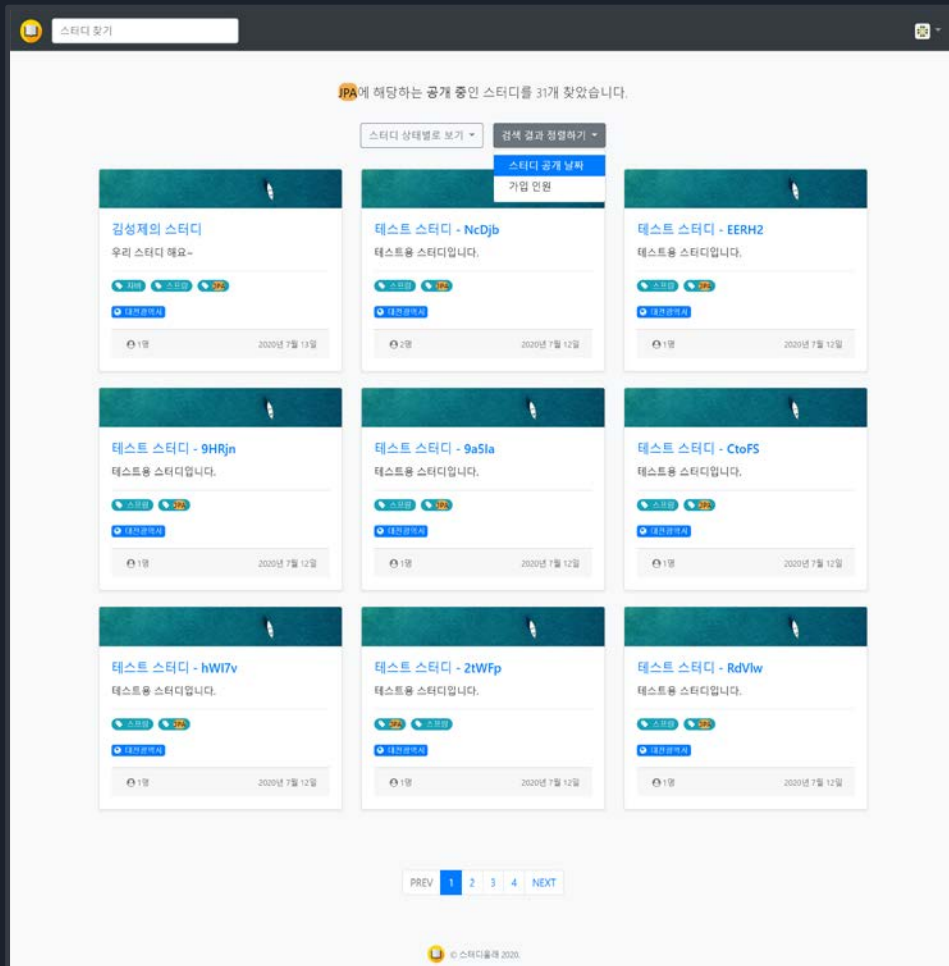
- **알림 표시**를 위한 읽지 않은 알림 여부

- **Interceptor Post Handler**를 등록하여, 어떤 요청에 대한 뷰를 렌더링하기 전에 해당 사용자가 읽지 않은 알림이 있는지 판단
- 결과 리다이렉트 요청 제어

### 스레드 풀 설정

```
13  @Slf4j
14  @Configuration
15  @EnableAsync
16  public class AsyncConfig implements AsyncConfigurer {
17
18      // 스레드 풀 크기, 스레드 풀
19
20
21      @Override
22      public Executor getAsyncExecutor() {
23          ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();
24          int processor = Runtime.getRuntime().availableProcessors();
25
26          executor.setCorePoolSize(processor);
27          executor.setMaxPoolSize(processor * 2);
28          executor.setQueueCapacity(50);
29          executor.setKeepAliveSeconds(60);
30          executor.initialize();
31
32          return executor;
33      }
34
35      @Override
36      public AsyncUncaughtExceptionHandler getAsyncUncaughtExceptionHandler() {
37          log.error("Error message : 'Async exception'");
38          return new SimpleAsyncUncaughtExceptionHandler();
39      }
40  }
```

# 스터디 검색 결과



## 스터디 검색 결과

- ✓ 스터디 이름 / 주제 / 활동 지역에 입력 **키워드**를 포함하는 스터디 목록
- ✓ 비로그인 상태에서도 검색 가능
- ✓ 검색 키워드 하이라이트 (Mark.js)
- ✓ 검색 결과를 9개씩 **페이징**
- ✓ **스터디 상태 별로 보기** : 공개 중 / 종료 됨
- ✓ **검색 결과 정렬** : 스터디 공개 날짜 / 가입 인원
- ✓ 기본적으로 검색 결과는 공개 중인 스터디에 대해 공개 날짜 순으로 정렬되어 표시
- ✓ 배너를 설정하지 않은 스터디는 기본 배너 이미지로 표시

## ❖ 페이징

- 스프링 데이터의 Pageable 인터페이스와 Querydsl4RepositorySupport 지원 클래스를 활용한 **페이징**
- **사용자 정의 리포지토리** 구현 → Querydsl4RepositorySupport 지원 클래스 상속
- 컨트롤러로부터 받은 Pageable 객체와 실제 조회 쿼리를 전달하여 **페이징 적용 메서드** 호출

※ 필요 시, 카운터 쿼리를 분리하여 성능 최적화

### 메서드 구현 - 스터디 검색 결과

#### 〈 스터디 조회 쿼리 〉

- 컬렉션 Left Join 과 distinct  
→ 애플리케이션 내 객체 **중복 제거**
- Where 절 다중 파라미터 사용  
→ **동적 쿼리** 해결

```
46      @Override
47      public Page<Study> searchAllByStudySearchAndPaging(StudySearch studySearch, Pageable pageable) {
48          List<Study> findStudies = queryFactory
49              .selectFrom(study)
50              .distinct()
51              .leftJoin(study.tags)
52              .leftJoin(study.zones)
53              .where(
54                  study.published.isTrue(),
55                  study.closed.eq(studySearch.isClosed()),
56                  keywordContainsInTitleOrTagsOrZones(studySearch.getKeyword())
57              )
58              .fetch();
59
60          return applyPagination(pageable,
61              contentQuery -> contentQuery.selectFrom(study).where(study.in(findStudies)));
62      }
63
64      private BooleanExpression keywordContainsInTitleOrTagsOrZones(String keyword) {
65          return study.title.containsIgnoreCase(keyword)
66              .or(study.tags.any().tag.title.containsIgnoreCase(keyword))
67              .or(study.zones.any().zone.localNameOfCity.containsIgnoreCase(keyword));
68      }
```

## ❖ 개발 기간

- 2020.05.15 ~ 06.26

## ❖ 보완할 점

- 부족한 **테스트 코드** → 유지 보수 및 변경 대응에 어려움
- **엔티티**를 직접 외부에 노출하는 문제 (특히, REST API 일 때)
  - 엔티티의 모든 값이 노출될 뿐만 아니라 스펙 변경에 어려움
  - 응답 스펙에 맞춰 별도의 **DTO** 반환
- 복잡한 화면 출력을 위한 **조회용 쿼리** 분리

## ❖ 개발 중 장애 요인과 해결 방안

- 일대다 관계의 컬렉션을 **batch-fetch-size** 설정으로 한번에 조회하여 사용할 때 발생했던 **PSQLException**
  - 해당 컬렉션의 원소가 **CLOB 타입**의 필드를 보유한 다른 엔티티를 지연 로딩으로 참조할 경우, 초기화 되지 않은 프록시 객체에 의한 **트랜잭션 범위 밖**에서의 CLOB 타입 데이터 조회 문제
  - 필요한 데이터에 맞춰 컬렉션 대신 **fetch join**과 **함께 중간 테이블을 직접 조회하거나 미리 프록시 객체를 초기화하여 해결**
- 이미지 파일에 대한 **서버 요청 크기 초과** 문제
  - 기존 **byte[] 타입**의 이미지 파일을 의존 라이브러리 추가 및 빈 설정을 통해 미리 **DataUri** 형태로 변환하여 요청

## ❖ 느낀 점 및 향후 계획

누군가 '이 프로젝트를 통해 어떤 성과를 거두었나'라는 질문을 한다면 가장 먼저 '개발에 대한 자신감을 키울 수 있었던 경험이었다'라고 대답할 것입니다. 짧지 않은 개발 기간 동안 설계부터 서비스 개발 및 테스트까지 프로젝트 전 과정에 걸쳐, 스스로 문제들을 고민하고 해결하며 완성된 결과물을 만들 수 있었던 이 경험을 통해 개발에 대한 자신감을 얻었습니다.

스프링, JPA, Querydsl 까지 평소 관심을 가지고 배웠던 자바 백엔드 기술들과 여러 오픈 소스들을 기반으로 직접 실제 서비스를 개발하는 과정은 매우 흥미롭고 의미 있는 경험이었습니다. 또한 소위 '백문이 불여일타'라고 하듯이, 신입 개발자로서 성장하기 위한 학습과 역량 개발에 있어서 스스로 코드를 작성해보려는 노력이 왜 중요한지 다시 한번 느꼈습니다.

마지막으로 본 프로젝트를 마무리하며 가장 아쉬웠던 점은 실제 서비스 배포까지 포함하지 못한 것입니다. 그래서 기회가 된다면 이번 프로젝트를 진행하며 겪었던 시행착오와 다양한 경험들을 바탕으로 보다 완성도 높고 실제 사용자들에게 제공할 제 자신만의 서비스를 만들어 보고 싶습니다.