스와브팅

Assigned To	㉾ 금준호 김형은
Due Date	@2024년 12월 29일
Meetings	프론트 기획 회의, 발표 구성 회의
Projects	번호팅 프로젝트
Status	Done
Time Status	❗ Overdue

왜?

1. 기존의 번호팅은 번호를 넣는 사람의 본인 인증이 되지 않음.
2. 사람이 일일이 하다보니 늦기도 하고 실수 가능성도 있음.
3. 뽑히는 사람의 입장에서 필터링을 하고 싶을수도 있지만 안됨
   (예, 같은 학과 제외, 너무 많이 차이나는 학번 제외 등)
4. 카톡 프사가 없는 사람들이 있는 문제가 있음
5. 나의 번호를 얼마나 많은 사람들이, 누가 가져갔는지 알 수 없음

그래서?

- 본인의 번호를 인증
- 상대 필터링 추가

- 서비스에서 프로필 사진을 넣어 프로필 없는 사용자 방지
- 나의 번호를 누가 뽑아갔는지도 확인 할 수 있도록

목표는?

- 개인정보를 보다 안전하게 서비스
- 보다 편리하게 서비스

어떻게?

카카오 로그인

→ 언제든지 시스템에 들어와서 확인 가능

번호 인증

→ 다른 사람의 번호를 넣는 것을 방지

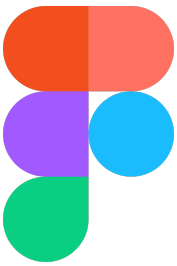
프로필 등록 및 승인

→ 프로필 사진이 없는 사용자 방지

학과 학번 필터링

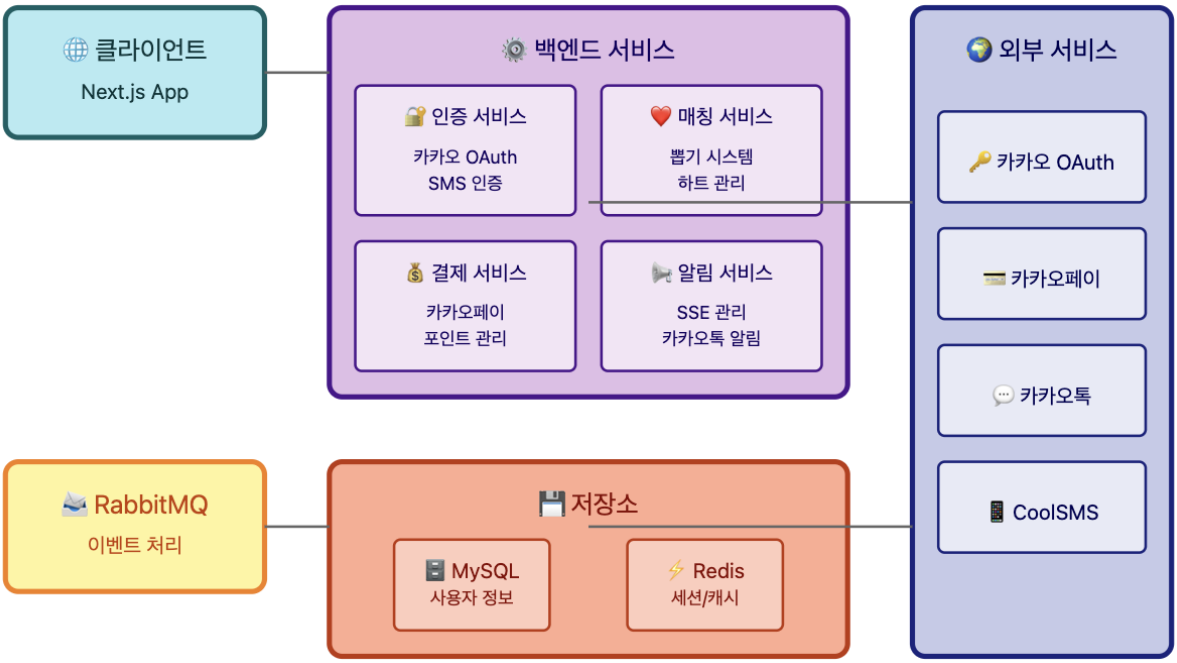
→ 원치 않는 공유 방지

그럼 한번 보시죠!

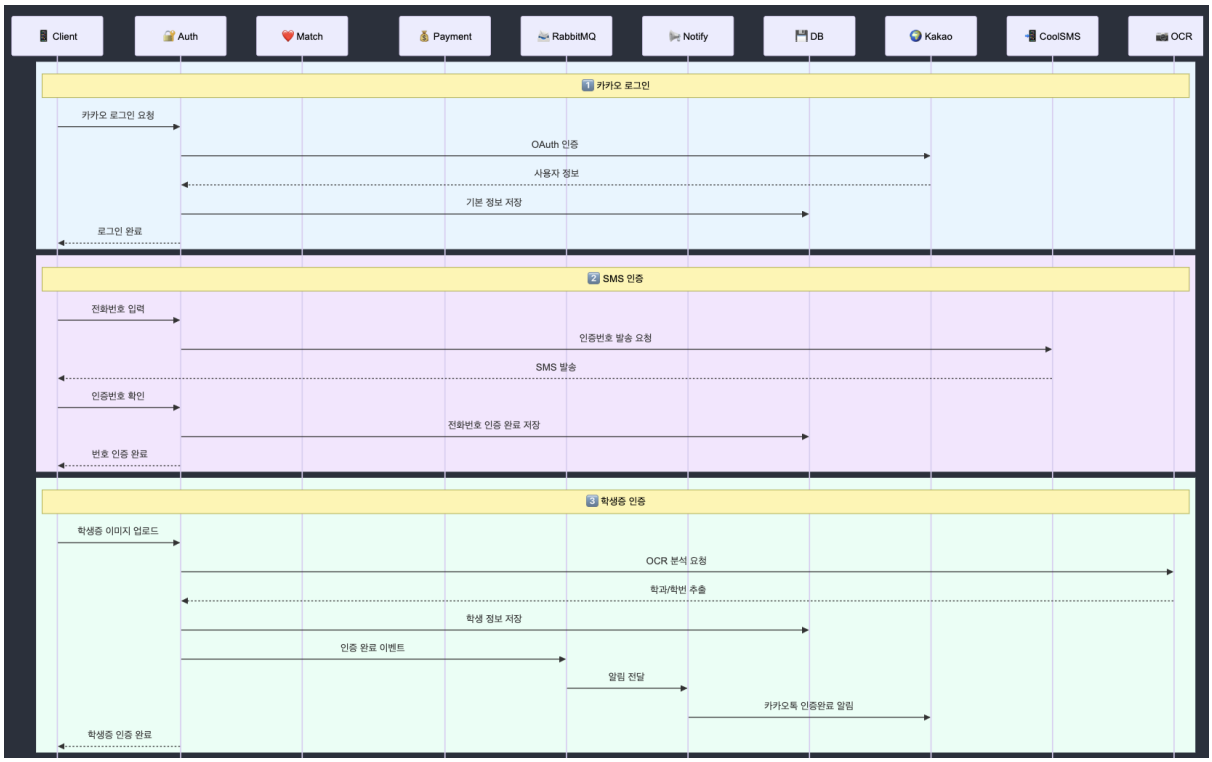


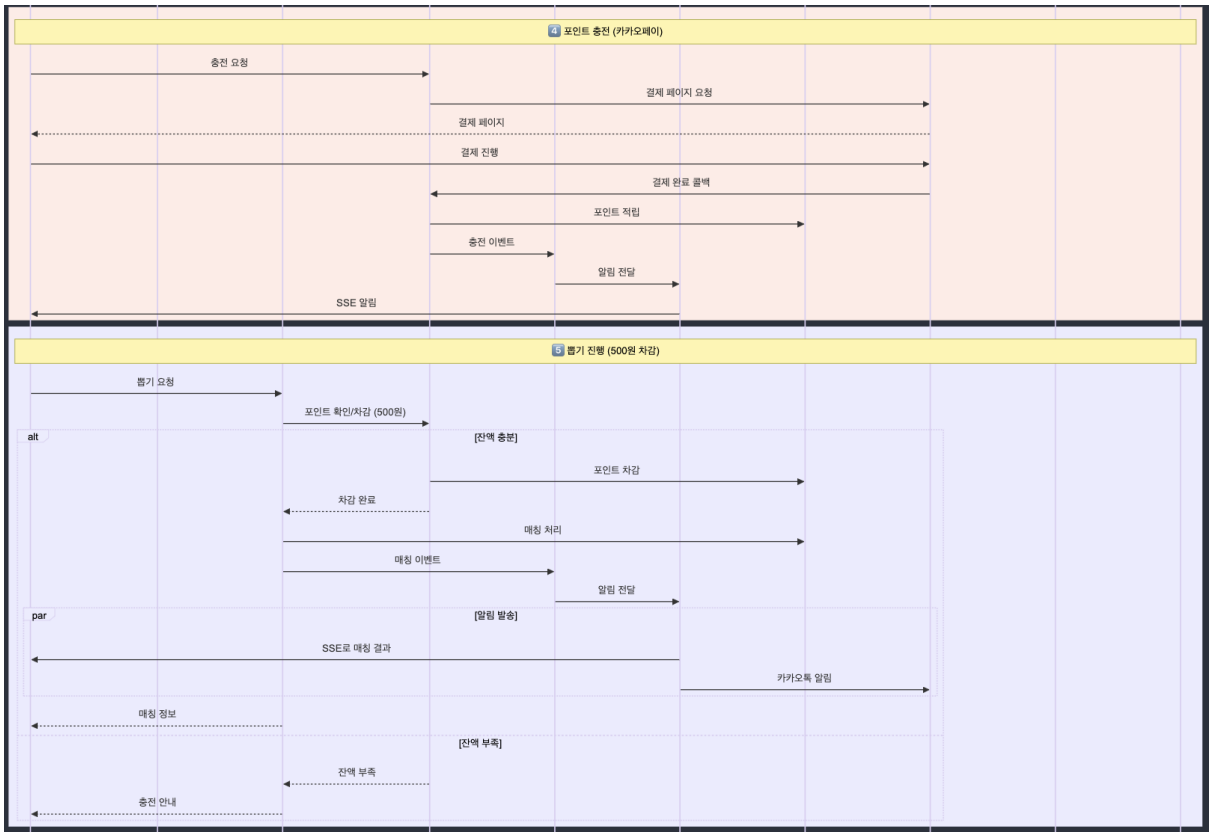
2024 도넛 프로젝트.pdf

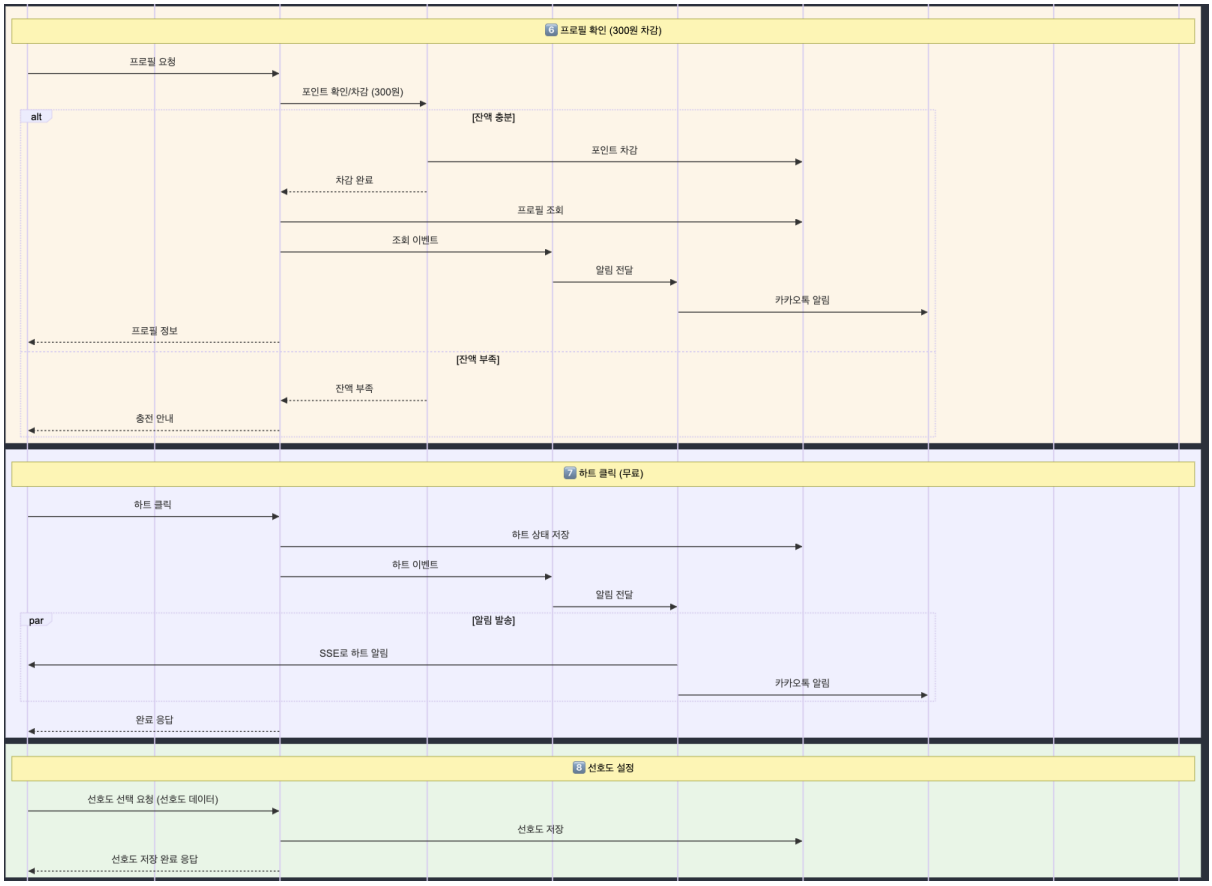
▼ Service Architecture

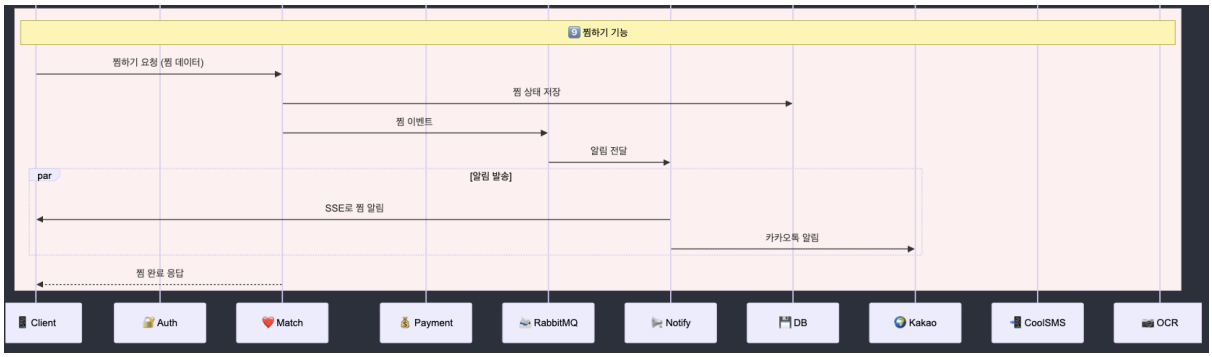


▼ Sequence Diagram




4 포인트 충전 (카카오페이)
충전 요청
결제 페이지 요청
결제 페이지
결제 진행
결제 완료 콜백
포인트 적립
충전 이벤트
알림 전달
SSE 알림
5 뽑기 진행 (500원 차감)
뽑기 요청
포인트 확인/차감 (500원)
alt
[잔액 충분]
포인트 차감
차감 완료
매칭 처리
매칭 이벤트
알림 전달
par
[알림 발송]
SSE로 매칭 결과
카카오톡 알림
매칭 정보
[잔액 부족]
잔액 부족
충전 안내





▼ Data models

1. USERS (사용자 테이블)

Column	Type	Constraints	Description
id	UUID	PK	사용자 고유 식별자
kakao_id	VARCHAR(100)	UNIQUE NOT NULL	카카오 소셜 로그인 ID
phone_number	VARCHAR(20)	UNIQUE NOT NULL	인증된 휴대폰 번호
email	VARCHAR(100)	NULL	알림 수신용 이메일
status	VARCHAR(20)	NOT NULL	상태(PENDING/ACTIVE/SUSPENDED/DELETED)
points	INTEGER	NOT NULL DEFAULT 0	보유 포인트

created_at	TIMESTAMP	NOT NULL	생성일시
updated_at	TIMESTAMP	NOT NULL	수정일시

2. PROFILES (프로필 테이블)

Column	Type	Constraints	Description
id	UUID	PK	프로필 고유 식별자
user_id	UUID	FK(users) UNIQUE	사용자 ID
nickname	VARCHAR(50)	NOT NULL	표시 이름
age	INTEGER	NOT NULL	나이
age_range	VARCHAR(20)	NOT NULL	연령대 구분
main_image_url	VARCHAR(255)	NULL	대표 프로필 이미지 URL
created_at	TIMESTAMP	NOT NULL	생성일시
updated_at	TIMESTAMP	NOT NULL	수정일시

3. USER_VERIFICATION (인증 테이블)

Column	Type	Constraints	Description
id	UUID	PK	인증 고유 식별자
user_id	UUID	FK(users)	사용자 ID
department	VARCHAR(100)	NOT NULL	학과명
student_id	VARCHAR(20)	NOT NULL	학번
type	VARCHAR(20)	NOT NULL	인증유형(STUDENT_CARD/PHONE/EMAIL)
status	VARCHAR(20)	NOT NULL	인증상태(UNVERIFIED/PENDING/VERIFIED/REJECTED)
reject_reason	TEXT	NULL	반려 사유
verified_at	TIMESTAMP	NULL	인증 완료 시각

4. USER_IMAGES (이미지 테이블)

Column	Type	Constraints	Description
id	UUID	PK	이미지 고유 식별자
user_id	UUID	FK(users)	사용자 ID
image_url	VARCHAR(255)	NOT NULL	이미지 URL
image_type	VARCHAR(20)	NOT NULL	이미지 용도(PROFILE/STUDENT_CARD)
is_active	BOOLEAN	NOT NULL	활성화 여부
created_at	TIMESTAMP	NOT NULL	생성일시

5. MATCH_PREFERENCES (매칭 선호도 테이블)

Column	Type	Constraints	Description
id	UUID	PK	선호도 고유 식별자
user_id	UUID	FK(users) UNIQUE	사용자 ID
exclude_departments	JSONB	NOT NULL	제외할 학과 목록

exclude_years	JSONB	NOT NULL	제외할 학번 목록
exclude_age_ranges	JSONB	NOT NULL	제외할 연령대 목록
is_active	BOOLEAN	NOT NULL	활성화 여부
updated_at	TIMESTAMP	NOT NULL	수정일시

6. MATCHES (매칭 테이블)

Column	Type	Constraints	Description
id	UUID	PK	매칭 고유 식별자
from_user_id	UUID	FK(users)	매칭 요청자 ID
to_user_id	UUID	FK(users)	매칭 대상자 ID
status	VARCHAR(20)	NOT NULL	매칭상태(PENDING/MATCHED/EXPIRED/CANCELLED)
is_photo_revealed	BOOLEAN	NOT NULL	사진 공개 여부
is_contact_revealed	BOOLEAN	NOT NULL	연락처 공개 여부
price	INTEGER	NOT NULL	매칭 비용(500원)
matched_at	TIMESTAMP	NOT NULL	매칭 시각
expired_at	TIMESTAMP	NULL	만료 시각

7. HEARTS (하트 테이블)

Column	Type	Constraints	Description
id	UUID	PK	하트 고유 식별자
from_user_id	UUID	FK(users)	보낸 사용자 ID
to_user_id	UUID	FK(users)	받은 사용자 ID
is_read	BOOLEAN	NOT NULL	읽음 여부
is_contact_revealed	BOOLEAN	NOT NULL	연락처 공개 여부
created_at	TIMESTAMP	NOT NULL	생성일시

8. PAYMENTS (결제 테이블)

Column	Type	Constraints	Description
id	UUID	PK	결제 고유 식별자
user_id	UUID	FK(users)	사용자 ID
payment_key	VARCHAR(100)	UNIQUE NOT NULL	결제 고유 키
type	VARCHAR(20)	NOT NULL	결제수단(KAKAOPAY)
amount	INTEGER	NOT NULL	결제금액
status	VARCHAR(20)	NOT NULL	결제상태 (READY/IN_PROGRESS/COMPLETED/FAILED/CANCELLED)
order_id	VARCHAR(100)	NOT NULL	주문번호
payment_info	JSONB	NULL	결제 상세 정보
paid_at	TIMESTAMP	NULL	결제완료 시각

9. POINT_TRANSACTIONS (포인트 거래 테이블)

Column	Type	Constraints	Description
id	UUID	PK	거래 고유 식별자
user_id	UUID	FK(users)	사용자 ID
payment_id	UUID	FK(payments) NULL	결제 ID
type	VARCHAR(20)	NOT NULL	거래유형 (CHARGE/POLL_PAYMENT/PROFILE_VIEW/REFUND)
amount	INTEGER	NOT NULL	거래금액
balance	INTEGER	NOT NULL	거래 후 잔액
description	TEXT	NULL	거래 설명
created_at	TIMESTAMP	NOT NULL	생성일시

10. NOTIFICATIONS (알림 테이블)

Column	Type	Constraints	Description
id	UUID	PK	알림 고유 식별자
user_id	UUID	FK(users)	사용자 ID
reference_id	UUID	NULL	참조 ID
type	VARCHAR(20)	NOT NULL	알림유형
message	TEXT	NOT NULL	알림 내용
notification_data	JSONB	NULL	알림 상세 데이터
is_read	BOOLEAN	NOT NULL	읽음 여부
read_at	TIMESTAMP	NULL	읽은 시각
created_at	TIMESTAMP	NOT NULL	생성일시

11. LIKES (찜하기 테이블)

Column	Type	Constraints	Description
id	UUID	PK	찜 고유 식별자
user_id	UUID	FK(users)	사용자 ID
target_user_id	UUID	FK(users)	찜한 사용자 ID
is_active	BOOLEAN	NOT NULL DEFAULT TRUE	활성화 여부
created_at	TIMESTAMP	NOT NULL	찜 추가일시

1. **사용자 도메인 (User Domain)**

   USERS (사용자 기본 정보)

   - 기본키: UUID 형식의 id
   - 카카오 로그인 연동을 위한 kakao_id (중복 불가)

- SMS 인증된 phone_number (중복 불가)
- 알림 수신용 email
- 사용자 상태 관리 (UserStatus: PENDING/ACTIVE/SUSPENDED/DELETED)
- 포인트 잔액 관리 (points)
- 생성/수정 시각 추적

PROFILES (사용자 프로필)

- Users와 1:1 관계
- 표시용 닉네임
- 나이 정보 (age, age_range)
- 대표 프로필 이미지 URL
- 생성/수정 시각 추적

USER_VERIFICATION (학생 인증 정보)

- Users와 1:N 관계 (여러 번의 인증 시도 이력 관리)
- 학과/학번 정보
- 인증 유형 (STUDENT_CARD/PHONE/EMAIL)
- 인증 상태 (UNVERIFIED/PENDING/VERIFIED/REJECTED)
- 반려 사유 저장
- 인증 완료 시각

USER_IMAGES (사용자 이미지)

- Users와 1:N 관계
- 이미지 URL 저장
- 이미지 용도 구분 (프로필/학생증)
- 활성 상태 관리
- 생성 시각

2. **매칭 도메인 (Matching Domain)**

   MATCH_PREFERENCES (매칭 선호도)

   - Users와 1:1 관계
   - JSON 형태로 제외할 학과/학번/연령대 저장
   - 활성 상태 관리
   - 수정 시각 추적

   MATCHES (매칭 정보)

   - Users와 다중 관계 (from_user_id, to_user_id)
   - 매칭 상태 관리 (PENDING/MATCHED/EXPIRED/CANCELLED)

- 사진/연락처 공개 여부
- 매칭 비용 (500원)
- 매칭 시각과 만료 시각

HEARTS (관심 표현)

- Users와 다중 관계 (보낸 사람, 받은 사람)
- 읽음 여부
- 연락처 공개 여부
- 생성 시각

3. **결제 도메인 (Payment Domain)**

   PAYMENTS (결제 정보)

   - Users와 1:N 관계
   - 카카오페이 결제키 (중복 불가)
   - 결제 금액과 상태 관리
   - 결제 상세 정보 JSON 저장
   - 결제 완료 시각

   POINT_TRANSACTIONS (포인트 거래)

   - Users, Payments와 관계
   - 거래 유형 구분
   - 거래 금액과 거래 후 잔액
   - 거래 설명과 생성 시각

4. **알림 도메인 (Notification Domain)**

   NOTIFICATIONS (알림)

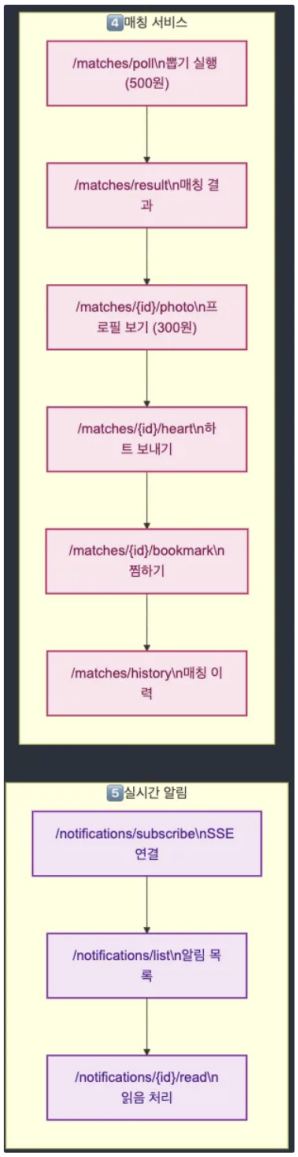
   - Users와 1:N 관계
   - 참조 ID로 관련 데이터 연결
   - 알림 유형과 메시지
   - 상세 데이터 JSON 저장
   - 읽음 상태와 시각 관리

특징적인 설계 포인트:

1. UUID 사용으로 보안성 강화
2. JSON 컬럼 활용으로 유연한 데이터 구조
3. 타임스탬프를 통한 이력 관리
4. FK를 통한 참조 무결성 보장
5. UK를 통한 중복 데이터 방지

6. Soft Delete 지원 (상태 필드 사용)

▼ API Flows



각 단계별 API 호출 상세:

1. **회원가입 & 로그인**

```
POST /api/v1/auth/kakao-login
GET  /api/v1/auth/status
POST /api/v1/auth/verify-phone
```

```
POST /api/v1/auth/verify-student
GET  /api/v1/auth/admin-verify
```

2. **프로필 설정**

```
POST /api/v1/users/profile
POST /api/v1/users/preferences
POST /api/v1/users/preferences/save
```

3. **포인트 충전**

```
POST /api/v1/payments/prepare
POST /api/v1/payments/approve
GET  /api/v1/users/points
```

4. **매칭 서비스**

```
POST /api/v1/matches/poll
GET  /api/v1/matches/result
GET  /api/v1/matches/{id}/photo
POST /api/v1/matches/{id}/heart
POST /api/v1/matches/{id}/bookmark
GET  /api/v1/matches/history
```

5. **실시간 알림**

```
GET  /api/v1/notifications/subscribe
GET  /api/v1/notifications/list
PUT  /api/v1/notifications/{id}/read
```

각 API의 주요 응답:

1. **인증 응답**

```
{
  "status": "SUCCESS",
  "data": {
    "userId": "uuid",
    "verificationStatus": "VERIFIED",
    "accessToken": "jwt-token"
  }
}
```

2. **매칭 응답**

```
{
  "status": "SUCCESS",
  "data": {
    "matchId": "uuid",
    "targetUser": {
      "department": "학과명",
      "studentYear": "학번",
      "isPhotoRevealed": false,
      "isContactRevealed": false
    }
  }
}
```

3. **포인트 응답**

```
{
  "status": "SUCCESS",
  "data": {
    "currentPoints": 1000,
    "transactions": [
      {
        "type": "CHARGE",
        "amount": 500,
        "timestamp": "2024-01-01T00:00:00"
      }
    ]
  }
}
```

4. **선호도 저장 응답**

```
{
  "status": "SUCCESS",
  "data": {
    "message": "선호도가 성공적으로 저장되었습니다."
  }
}
```

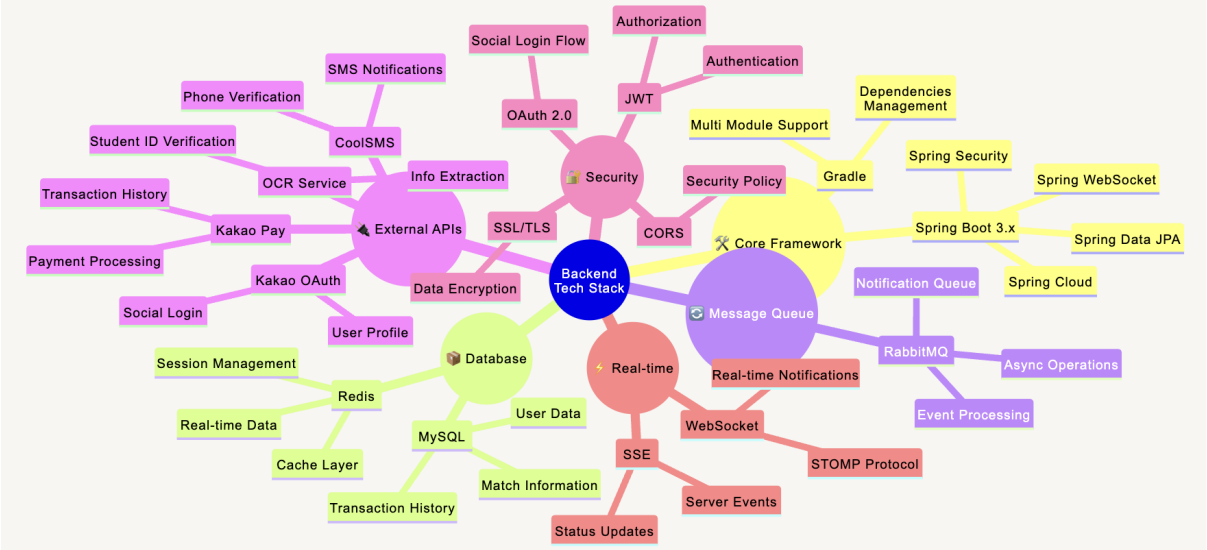
5. **찜하기 응답**

```
{
  "status": "SUCCESS",
  "data": {
    "message": "프로필이 찜 목록에 추가되었습니다."
```

```
    }
}
```

에러 응답 형식:

▼ Tech of backend

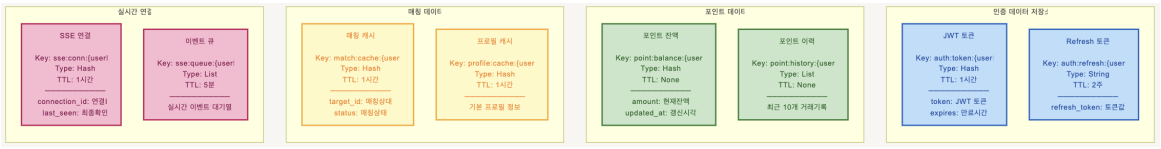


각 기술 선택에 대한 상세 의사 결정 내용:

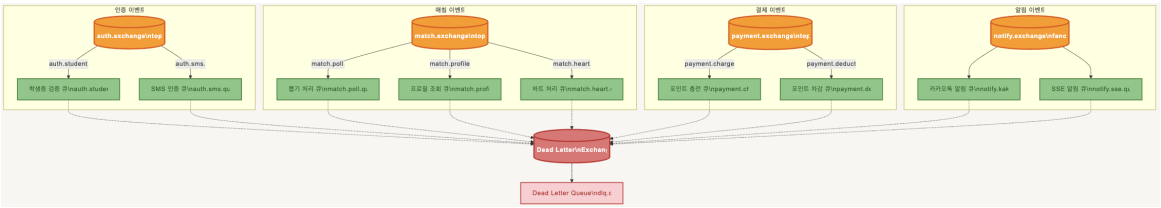
1. **Spring Boot 3.x 선택(개발진에서 공통으로 사용할 수 있는 언어 및 프레임워크)**
   - 결정 이유:
     - Java 17+ 지원으로 최신 언어 기능 활용 가능
     - 강력한 DI/IoC 지원
     - 광범위한 라이브러리 생태계
     - 안정적인 성능과 검증된 프레임워크
   - 대안 검토:
     - Node.js: 비동기 처리는 우수하나 타입 안정성 부족 → js에 대한 이해도 부족으로 timeline에 문제 가능성 검토
     - Kotlin/Spring: 러닝 커브가 상대적으로 높음
2. **아키텍처 접근 방식**
   - 결정 이유:
     - 초기에는 Monolithic으로 시작
     - 필요시 점진적으로 MSA로 전환 가능

- 개발 및 배포 프로세스 단순화
- 대안 검토:
  - MSA: 초기 복잡도 증가
  - Serverless: 콜드 스타트 이슈

3. **데이터베이스 - MySQL**
   - 결정 이유:
     - 강력한 트랜잭션 지원
     - 데이터 정합성 보장
     - 포인트 시스템에 적합
   - 대안 검토:
     - MongoDB: 스키마 유연성은 높으나 트랜잭션 제약
     - PostgreSQL: 기능은 우수하나 운영 복잡도 증가
4. **캐시 시스템 - Redis**



   - 결정 이유:
     - 인메모리 성능
     - Pub/Sub 기능 지원
     - 세션 관리 용이
   - 대안 검토:
     - Memcached: 단순 캐시에 특화
     - Hazelcast: 러닝 커브가 높음
5. **메시지 큐 - RabbitMQ**



   - 결정 이유:
     - 대용량 처리에 대한 실시간 처리 가능
     - 다양한 메시징 패턴 지원

    - 운영 관리 도구 제공
    - aws를 통한 쉬운 관리 가능
- 대안 검토:
    - Kafka: 대용량 처리에 특화되어 있으나 초기 설정 복잡
    - ActiveMQ: 성능이 상대적으로 낮음