ESCAPE ROOM
[ECR] Web Dev

What is this Project?

다양한 방탈출 카페 데이터를 통합 관리하며,
검색, 예약, 리뷰 기능을 제공하는 종합 플랫폼입니다.

What are the main features?

- 방탈출 카페 검색 및 필터링
- 예약 가능한 시간대 실시간 확인 및 예약
- 사용자 리뷰 관리 및 평가

3-1. 프로젝트 – 프로젝트 소개

프로젝트 소개 (ECR)

[Escape Room]

- 방탈출 스마트 스토어 웹 서비스 개발 (백엔드 & 프론트엔드 개발)
- 팀 프로젝트 총 5명
- 2024. 09.02 – 2024. 09. 27
- GitHub - https://github.com/kwanghunk/ECR-WEB-Project

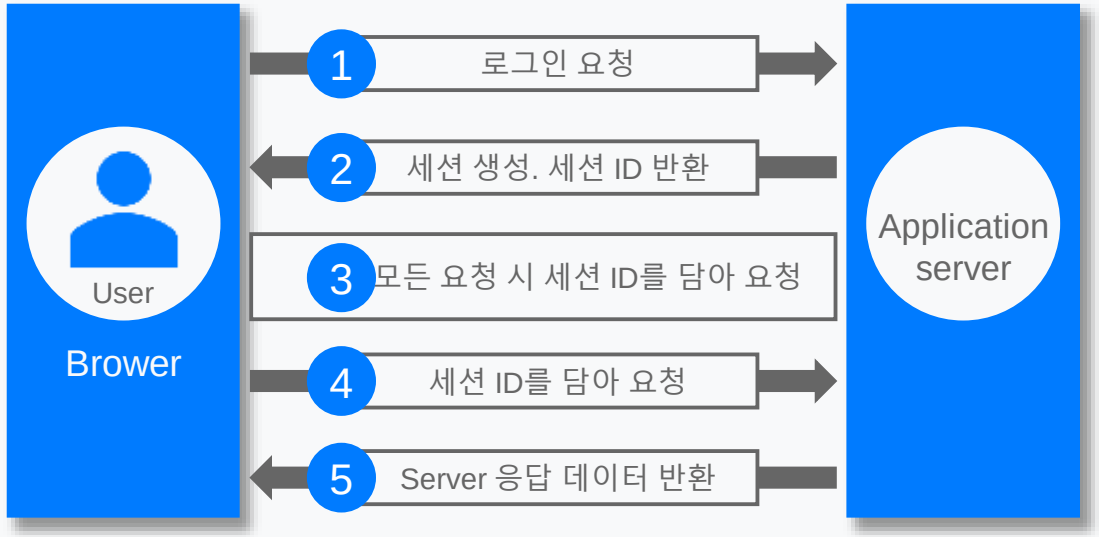
[Tech Stack]

공통	사용자	관계자	관리자
회원가입, 로그인 상품 목록 및 상품 상세 조회	상품 예약 관리 예약 내역 관리 리뷰 관리	상품 등록/삭제	사용자/관계자 회원관리 예약 상품 관리 상품, 리뷰, 공지사항 관리

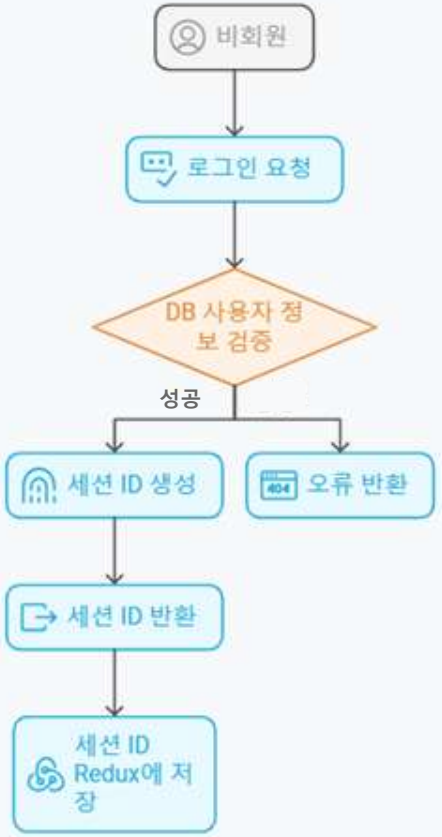
3-1. 프로젝트 – 시스템 흐름도

시스템 흐름도

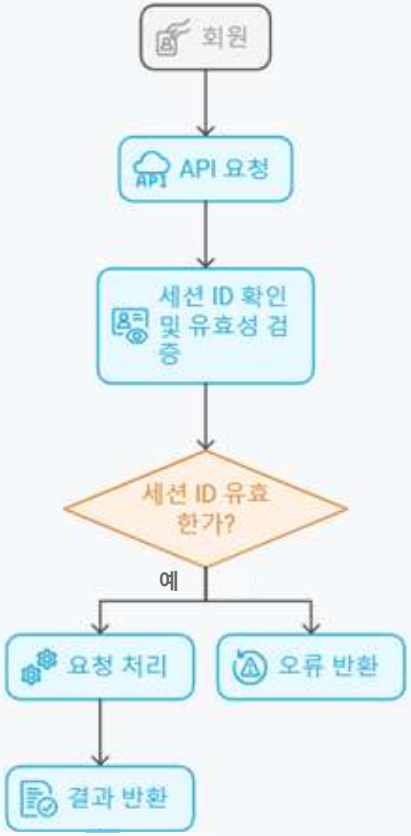
- 사용자 로그인 및 API 요청 처리 과정을 시각화 한 것입니다.
- 로그인 시 세션 ID를 생성하고, Redux에 저장하여 인증 상태를 관리합니
- API 요청은 세션 ID 유효성을 검증한 뒤 처리 결과를 반환합니다.
- 효율적인 세션 관리와 데이터 일관성을 위해 설계되었습니다.
- 확장성과 보안을 고려한 구조입니다.



로그인 흐름도



API 요청 처리 흐름도



3-1. 프로젝트 – 담당역할

담당역할 - 예약시스템

1. 주요 기능
   - “동시성 제어를 활용한 예약 시스템 구축”
2. 문제 정의
   - 다중 사용자가 동시에 예약할 경우의 데이터 충돌을 방지하고 데이터 무결성을 보장하기 위해 동시성 제어를 적용했습니다.
3. 핵심 코드
   - @Transactional을 사용하여 예약 도중 오류가 발생하면 롤백되도록 설정
   - 예약 가능 여부를 먼저 확인하고, 같은 시간대에 중복 예약이 있으면 예약을 차단
4. 결과
   - 동시 요청 발생 상황에서도 데이터 무결성 유지와 안정적인 예약 시스템이 작동.

```java
@Transactional
public boolean reserve(Reservation reservation) {
    Optional<Reservation> existingReservation =
        reservationRepository.findByTema_TemaNoAndUseDateAndUseTime(
            reservation.getTema().getTemaNo(),
            reservation.getUseDate(),
            reservation.getUseTime()
    );

    if (existingReservation.isPresent()) return false;

    reservation.setPaymentStatus("결제 대기");
    reservationRepository.save(reservation);
    return true;
}
```

3-1. 프로젝트 – 담당역할

담당역할 – 예약관리시스

1. 주요 기능
   - 사용자 예약 내역 조회 및 취소 요청
   - 관리자 예약 승인 및 취소 처리
2. 문제 정의
   - 다중 요청으로 인한 데이터 무결성 문제 발생
   - 인증되지 않은 사용자의 예약 접근 제한 필요
3. 핵심 코드
   - 트랜잭션 적용으로 데이터 무결성 보장
   - 인증 로직 추가로 보안 강화
4. 결과
   - 트랜잭션과 인증 로직으로 안정적인 예약 관리 구현

```java
@Transactional
public Reservation approveReservation(Long reservationCode) {
    Optional<Reservation> reservation =
            reservationRepository.findById(reservationCode);
    if(reservation.isPresent()) {
        Reservation res = reservation.get();
        res.setPaymentStatus("예약확정");
        return reservationRepository.save(res);
    }
    return null;
}
```

```java
@GetMapping("/findUserReserveAll")
public ResponseEntity<?> findUserReserveAll(
        @RequestParam(name = "page") int page,
        @RequestParam(name = "size") int size,
        HttpServletRequest request) {
    try {
        Member sessionMember = AuthenticationUtils.authClient(request);
        String  userId = sessionMember.getMemberId();

        List<Reservation> result =
                reservationService.findUserReservations(userId, page, size);
        return result != null && !result.isEmpty()
                ? ResponseEntity.ok(result)
                : ResponseEntity.status(204).body("예약 내역이 없습니다.");
    } catch (UnauthorizedException e) {
        return ResponseEntity.status(403).body(e.getMessage());
    } catch (Exception e) {
        return ResponseEntity.status(500).body("서버 내부 오류 발생: " + e.getMessage());
    }
}
```

3-1. 프로젝트 – 담당역할

담당역할 – 회원관리시스템

1. 주요 기능

• 관리자: 일반회원 및 제휴업체 회원 관리

• 회원 조회(페이징 처리) 및 회원 삭제 기능 제공

2. 문제 정의

• 대량의 데이터를 처리할 때 조회 성능 저하 문제

• 회원 삭제 시 무결성 원칙 위배 오류 발생 문제

3. 핵심 코드

• 페이징 처리로 데이터 조회 성능 최적화

• 연관된 데이터 삭제를 위해 각 레포지토리를 순차적 호출

4. 결과

• 페이징 처리로 대량 데이터 처리 성능 개선

• @Transactional로 데이터를 안전하게 삭제

```
@Transactional
public void deleteMemberAndReviewsAndTemas(String memberId) {
    if(memberRepository.existsById(memberId)) {
        reviewRepository.deleteByUserId(memberId);
        temaRepository.deleteByMemberId(memberId);
        memberRepository.deleteById(memberId);
    } else {
        throw new IllegalArgumentException("해당 회원을 찾을 수 없습니다");
    }
}
```

```
@GetMapping("/findClientAll")
public ResponseEntity<?> getAllClients(
        @RequestParam(name = "loginType") int loginType,
        @RequestParam(name = "page") int page,
        @RequestParam(name = "size") int size,
        HttpServletRequest request) {
    try {
        AuthenticationUtils.authClient(request);
        List<Member> list = memberService.getAllClients(loginType, page, size);
        return ResponseEntity.ok(list);
    } catch (UnauthorizedException e) {
        return ResponseEntity.status(403).body(e.getMessage());
    }
}
```