

POPSPOT

좋은 공간과
새로운 경험을
연결하는 팝업스토어 플랫폼

01 POPSHOT BACKGROUND

- OVERVIEW
- BACKGROUND
- ANALYSIS IPA, TOP3 분석

04 POPSHOT 기능 시연

02 PROJECT DIRECTION

- OBJECTIVES 프로젝트 목표

04 프로젝트 후기

- Q&A 질의응답

03 PROJECT ARCHITECTURE

- DESIGN CONCEPT 디자인 컨셉
- ARCHITECTURE 사용 기술 스택
- ERD DIAGRAM
- UML DIAGRAM
- CODE 주요 기능 코드

OVERVIEW

장기간의 코로나19로 경험을 중시하는 소비문화

“대세는 POP-UP STORE”



플레이슈머

유행에 따라 소비,
소비를 놀이처럼 즐기는 사람



편슈머

물건을 구매할 때, 재미를
추구하는 소비자



스토리슈머

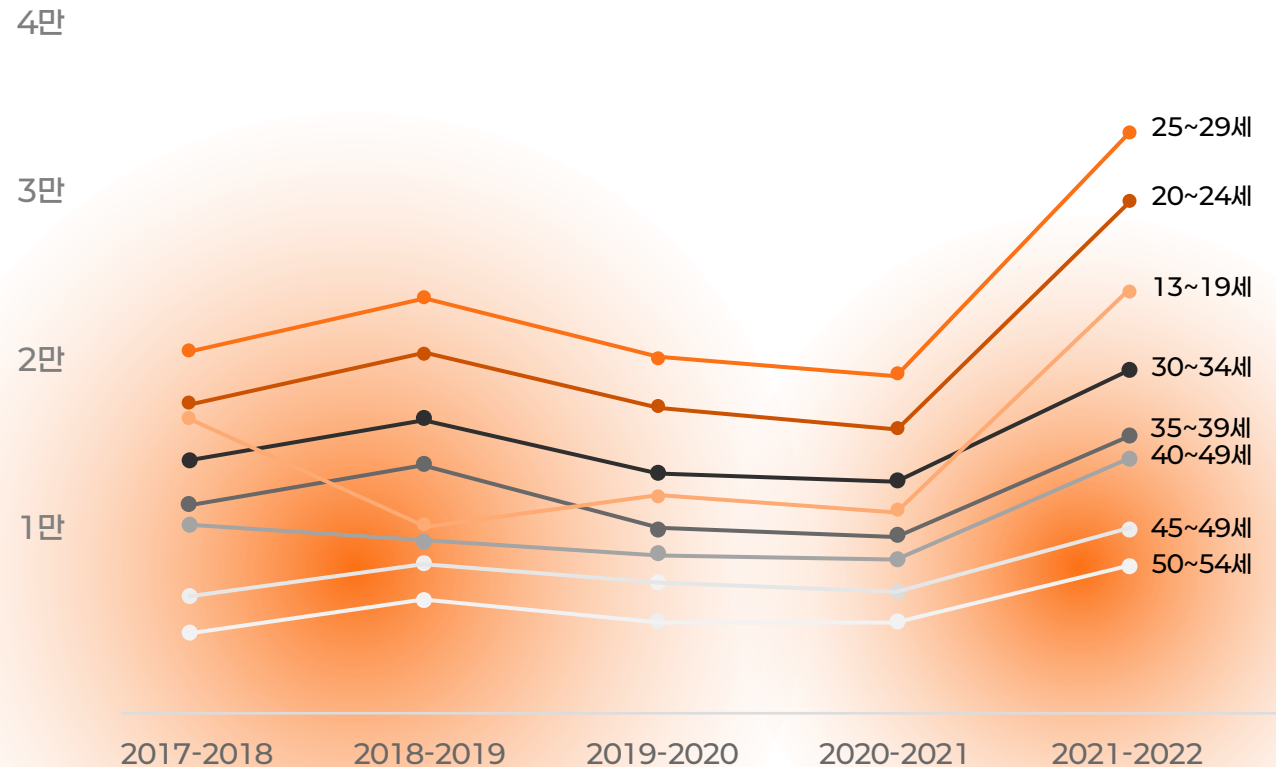
이야기를 찾는 소비

BACKGROUND

팝업스토어, 세대와 나이를 넘어선 **트렌드 아이콘**

2021년부터 2022년까지 팝업 스토어에 대한 검색량은 11만 5,801건으로 전년 대비 5천 건 가까이 상승하였고, 가장 많이 검색한 연령은 20대로 그 수치가 전년 대비 2배가량 늘었고, 10대와 30대가 그 뒤를 이으며 특히, MZ세대에게 인기 있는 키워드임이 밝혀졌습니다.

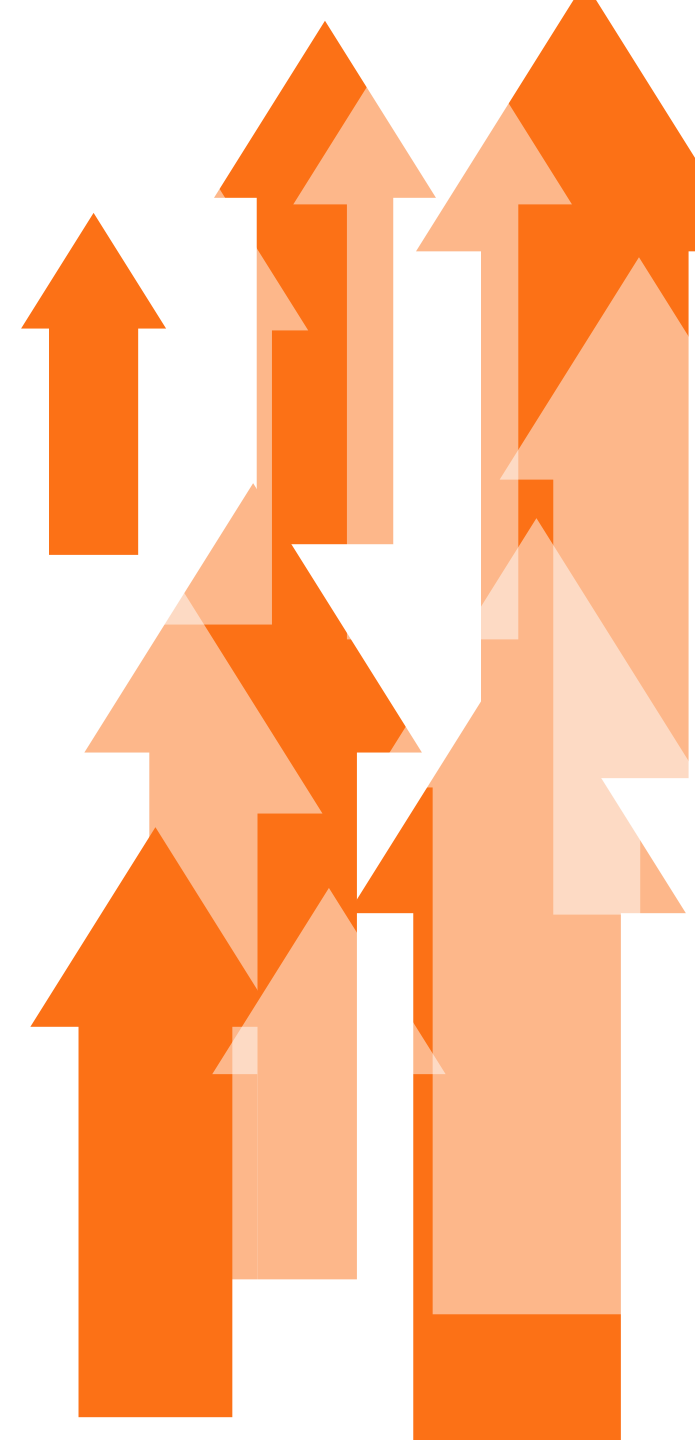
연령대별 팝업 스토어 검색량



BACKGROUND

팝업 스토어 플랫폼의 성장

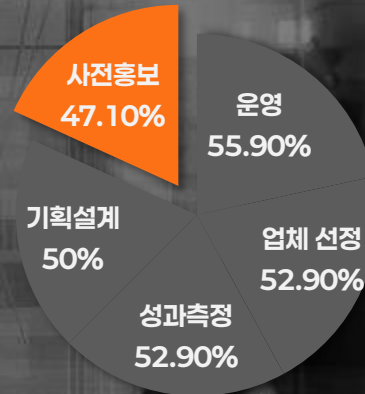
이커머스 시장이 천문학적으로 성장하며 플랫폼 사업의 다변화
개인으로 부족한 홍보 및 복잡한 절차를 간편하게 해결 가능.



BACKGROUND

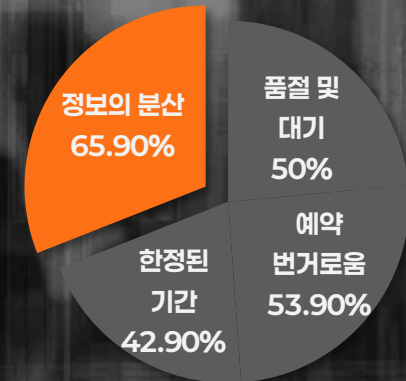
팝업 스토어의 성장 속 새로운 고민

팝업스토어가 전성기를 맞이하며
다양한 기회를 제공하고 있지만,
소비자와 기업 마케터 모두 새로운 과제에 직면



기업 마케터의 고민 사항

현장 운영부터 대행업체 선정성과
측정 및 마케팅에 이르는
복잡한 절차를 고민



소비자의 고민 사항

분산된 정보, 번거로운 예약,
짧은 운영 기간등으로 인해
팝업스토어 방문에 고민

유사 사이트 분석

TOP3 ANALYSIS

POPPLY

소비자 기준 편리함

소비자 에게 다양한
컨텐츠 및 기능 제공

V.가치공간



기획자 기준 편리함

기획자가 공간을
간편하게 관리할 수 있음

**hey
POP**



디자인 적인 사이트

다양한 컨텐츠를
디자인적이게 제공

IPA ANALYSIS

디자인적

POPPLY

DATE POP

기능적

HYPEBEAST

오늘 하루예술

R

[sweetspot]

V.가치공간

오늘의집

PRESENTATION



“ 늘어만 가는 팝업 스토어,
한 눈에 볼 수 있게
정리된 사이트는 없을까? ”

😬 일단 저장은 해놓지만 예약을 따로 해야한다는 번거로움.

😬 무수한 중첩된 홍보 게시물로 시기가 지난 줄 몰라 예약하지 못한 아쉬움.

짧은 기간 운영하는 팝업 스토어. 한 눈에 보고 관리 할 수 있는 서비스가 필요했습니다.

OBJECTIVES_NEED

개발 목표

리서치를 통해 도출한 문제점을 바탕으로 팝업스토어 솔루션 제공



무분별한 팝업 스토어

온라인 시장에서 무분별하게
분포 되어 있는 팝업 스토어



중복된 콘텐츠

다양한 사이트에 분포되어 있는
팝업 스토어 콘텐츠



고객의 시선

고객의 시선을 오래 머물게
하고 싶음



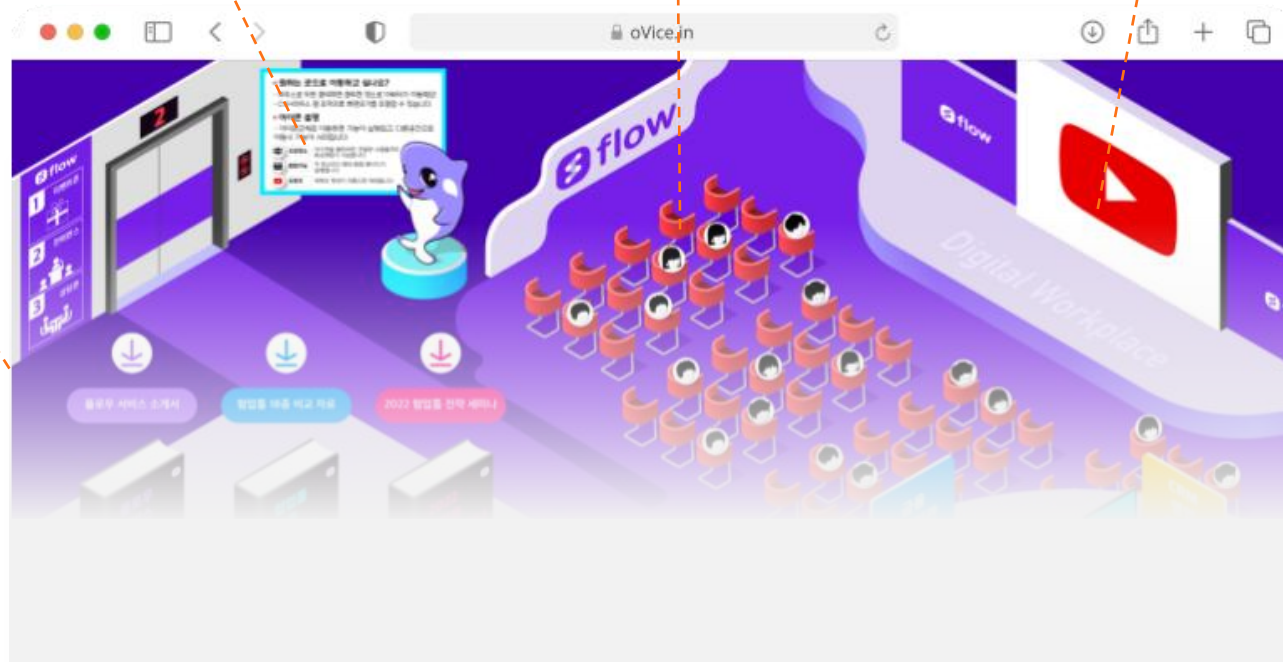
전략

다수의 고객들의 반응을 통해
향후 전략을 세우고 싶음



홍보

홍보 과정 중 문제점 도움 요청의
어려움



개발 목표

리서치를 통해 도출한 문제점을 바탕으로 팝업스토어 솔루션 제공



손쉬운 접근성

팝업스토어 중개 사이트로
사용자와 기획자의 손쉬운 접근성



카테고리 필터링 제공

종류와 분류별로 확인 가능한 정확한
카테고리 필터링 제공



고객의 시선

디자인적인 페이지
레이아웃 제공



소비자 피드백

소비자 피드백 및 만족도
조사 제공



전문가의 피드백

전문가의 피드백 제공





PROJECT ARCHITECTURE

POPSPOT 계획

DESIGN CONCEPT

DESIGN CONCEPT

VISUAL CONCEPT

이미지를 중심으로 깔끔하면서도 트렌디하게 디자인하였으며,
파란 계열의 강조색으로 현대적이고 감각적인 느낌을 살렸습니다.

BRAND TARGET

- 팝업 스토어 관심있는 누구나
- 소규모 브랜드 및 신생 스타트업
- 취미와 라이프스타일에 민감한 소비자
- 행사 기획자 및 마케터

FONTSTYLE

DEPARTURE_MONO 수트

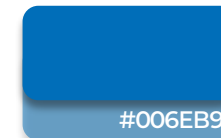
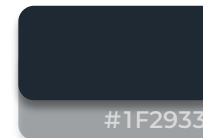
프리젠테이션

LOGO



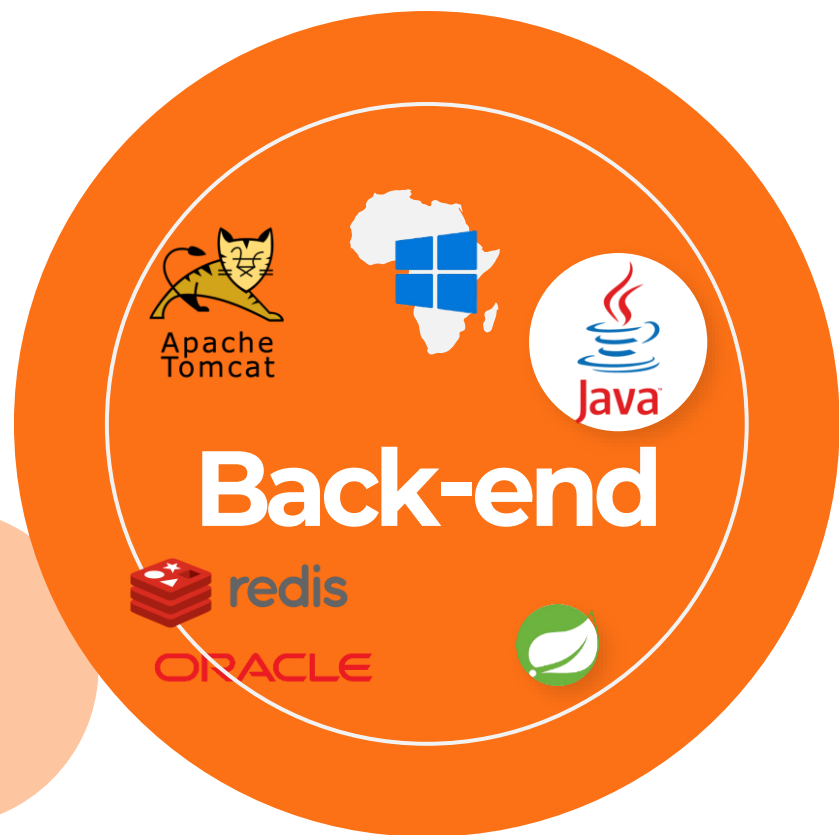
COLOR

트렌디하고 깔끔한 MZ 세대 감성을 담아 블루컬러로 활기찬 이미지를 강조,
화이트와 블랙으로 시각적 균형과 세련된 잡지 스타일을 완성했습니다.



사용 기술 스택

ARCHITECTURE



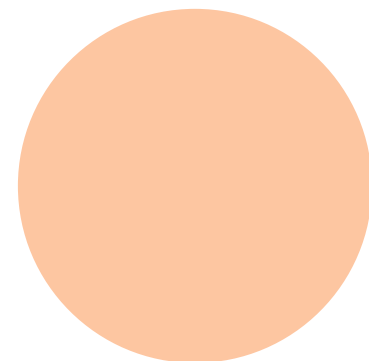
front-end



tools



tools



ERD 다이어그램

ERD DIAGRAM

IEWS_CNT

EVENT_NO	NUMBER	행사 번호, PK, FK
VIEW_DATE	VARCHAR2	시간이 포함된 날짜, PK
VIEWCOUNT	NUMBER	조회수
CREATED_DATE	DATE	생성날짜
MODIFIED_DATE	DATE	수정날짜

EVENT

USER_ID	VARCHAR2	아이디, PK, FK
EVENT_NO	NUMBER	행사 번호, PK
NAME	VARCHAR2(15)	이름
TITLE	VARCHAR2(90)	제목
COMPANY	VARCHAR2(30)	사업체명
CONTENT	VARCHAR2(1000)	내용
START_DATE	DATE	시작일
END_DATE	DATE	종료일
OPEN_TIME	VARCHAR2(4)	개장시간
CLOSE_TIME	VARCHAR2(4)	폐장시간
TYPE	VARCHAR2(1)	이벤트 타입
TAGS	VARCHAR2(20)	태그
CREATED_DATE	DATE	생성일
IS_DELETED	NUMBER(1)	삭제 여부
DELETED_DATE	DATE	삭제일

LIKECOUNT

EVENT_NO	NUMBER	행사 번호, PK, FK
USER_ID	VARCHAR2(12)	아이디, PK, FK
CREATED_DATE	DATE	생성일
MODIFIED_DATE	DATE	수정일

USER

USER_ID	VARCHAR2(12)	아이디, PK
-	NUMBER	생성번호
USER_PWD	VARCHAR2(20)	패스워드
NAME	VARCHAR2(15)	이름
EMAIL	VARCHAR2(50)	이메일
PHONE	VARCHAR2(11)	전화번호
IS_RECEPTION	NUMBER(1)	수신여부 동의
IS_SNS_LOGIN	NUMBER(1)	SNS 로그인 여부
TYPE	NUMBER(1)	회원 타입
CREATED_DATE	DATE	회원가입일
IS_WITHDRAW	NUMBER(1)	회원탈퇴 여부
WITHDRAW_DATE	DATE	회원탈퇴일

REVIEW

EVENT_NO	NUMBER	행사 번호, PK
USER_ID	VARCHAR2(12)	아이디, PK, FK
REVIEW_NO	NUMBER	리뷰 번호, PK
REVIEW_CONTENT	VARCHAR2	리뷰 내용
RATE	NUMBER(3)	별점
CREATED_DATE	DATE	작성일
MODIFIED_DATE	DATE	수정일
IS_DELETED	NUMBER(1)	삭제여부
DELETED_DATE	DATE	삭제일

USER_SUPPORT

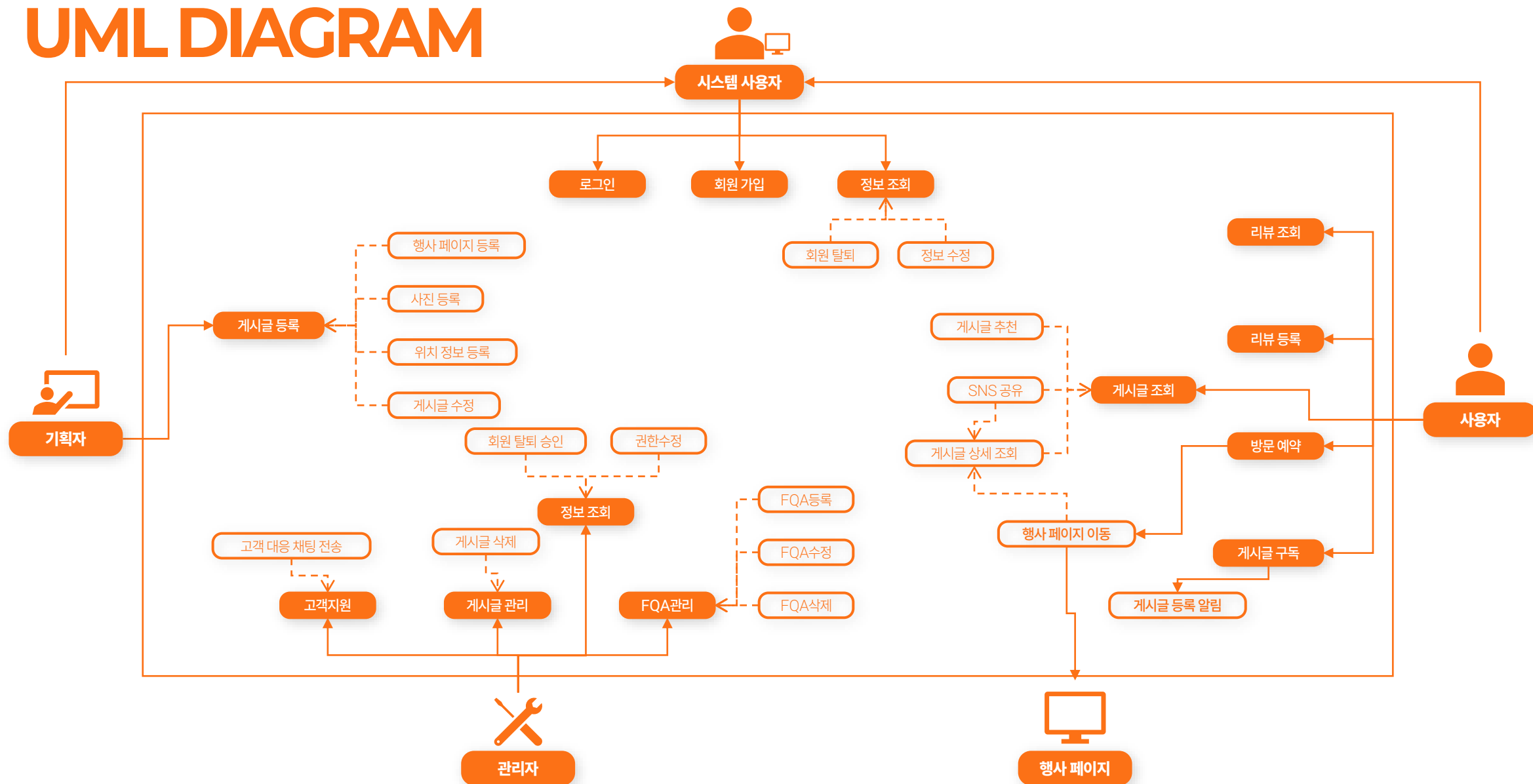
USER_ID	VARCHAR2(12)	아이디, PK, FK
INQUIRY_NO	NUMBER	고객문의 번호, PK
MODIFIED_DATE	DATE	수정일
TITLE	VARCHAR2(50)	문의 제목
INQUIRY	VARCHAR2(4000)	문의 내용
TYPE	NUMBER(1)	문의 타입
CREATED_DATE	DATE	문의 일시
IS_DELETED	NUMBER(1)	삭제 여부
DELETED_DATE	DATE	삭제일

ANSWER

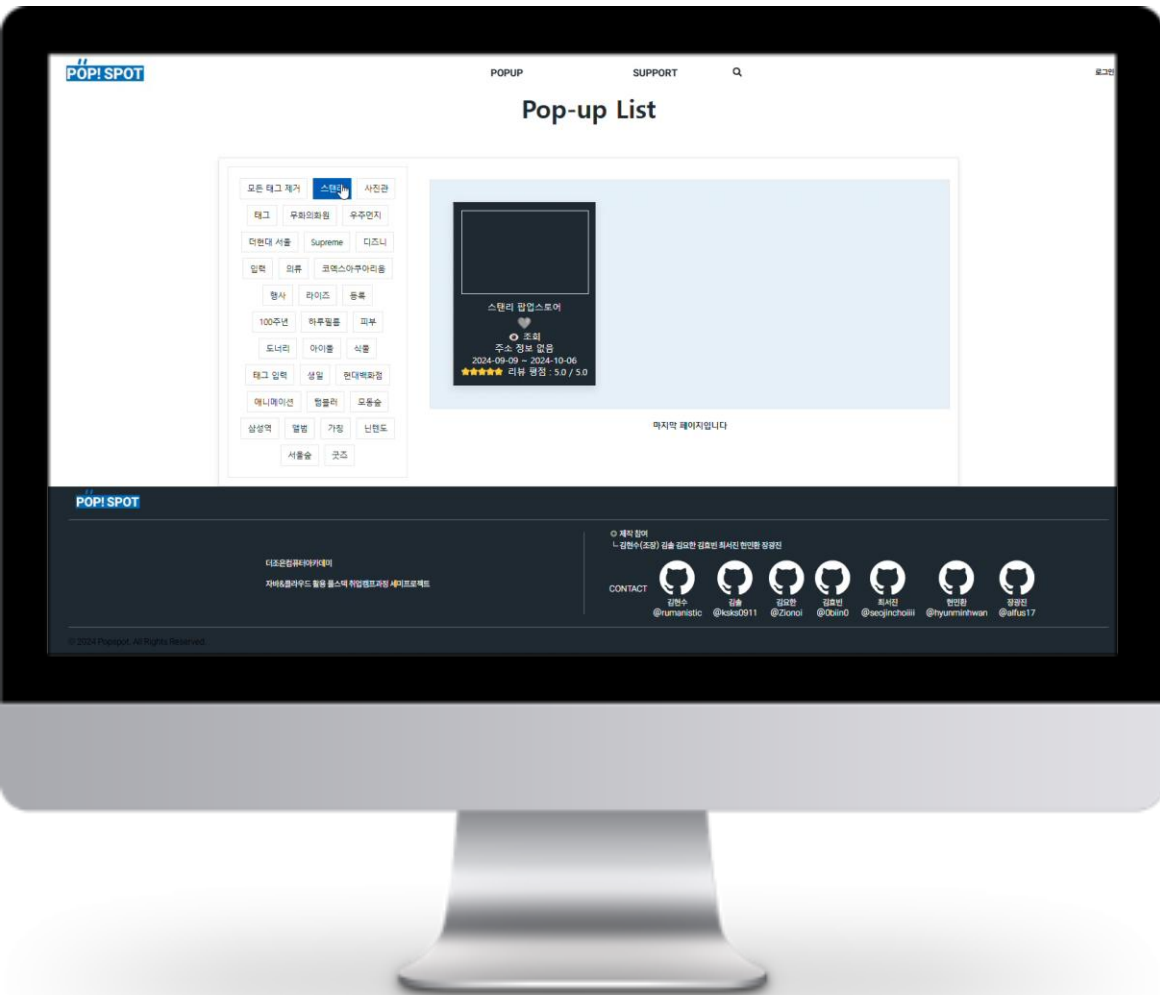
INQUIRY_NO	NUMBER	고객문의 번호, PK, FK
USER_ID	VARCHAR2(12)	아이디, PK, FK
ANSWER_NO	NUMBER	고객문의 답변 번호, PK
ANSWER	VARCHAR2	문의 답변
CREATED_DATE	DATE	등록 일시

FAQ

USER_ID	VARCHAR2(12)	아이디, PK, FK
FAQ_NO	NUMBER	FAQ 번호, PK
QUESTION	VARCHAR2(50)	질문
ANSWER	VARCHAR2(400)	답변
CREATED_DATE	DATE	생성일
MODIFIED_DATE	DATE	수정일
IS_DELETED	NUMBER(1)	삭제 여부
DELETED_DATE	DATE	삭제일



카테고리별 키워드 검색 기능



코드소개

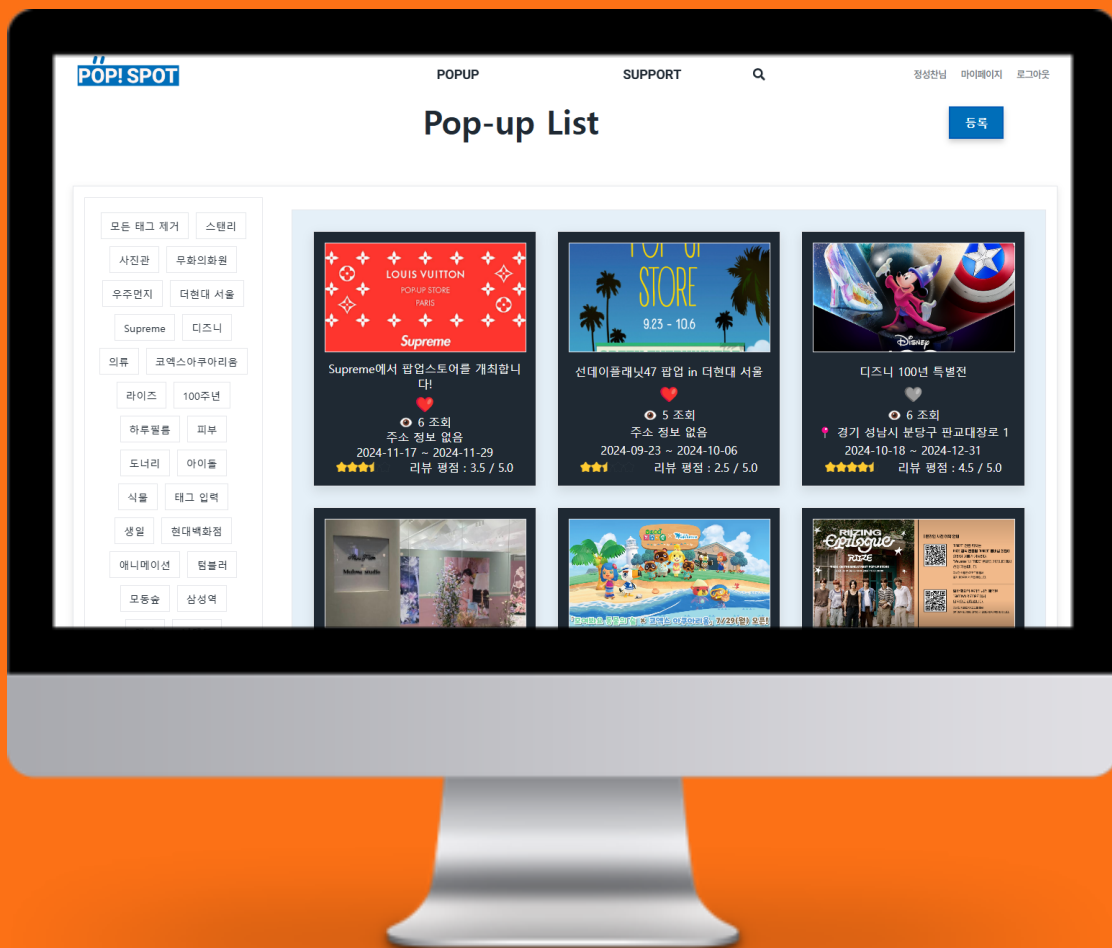
```
public ApiResponse searchListByKeyword(String keyword) {
    HashMap<String, Object> result = new HashMap<>();
    List<Event> eList = es.searchListByKeyword(keyword);
    if (eList.size() > 0) {
        Set<Long> eventNos = new HashSet<>(); // 검색된 이벤트의 eventNo만 추출
        for (Event e : eList) {
            eventNos.add(e.getEventNo());
        }
        // 검색된 이벤트의 eventNo를 기반으로 리뷰 포인트 가져오기
        List<ReviewPoint> rPoints = rs.getReviewPointAvg(eventNos);
        HashMap<Long, Double> rPoint = new HashMap<>();
        for (ReviewPoint rp : rPoints) {
            rPoint.put(rp.getEventNo(), rp.getReviewPointAvg());
        }
        result.put("eList", eList); // 검색된 이벤트 리스트
        result.put("rPoint", rPoint); // 해당 이벤트의 리뷰 포인트 매핑
        return ApiResponse.builder(true, SUCCESS, result);
    }
    // 검색 결과가 없는 경우에도 빈 eList와 rPoint 반환
    result.put("eList", new ArrayList<>()); // 빈 리스트
    result.put("rPoint", new HashMap<>()); // 빈 매핑 객체
    return ApiResponse.builder(true, NOT_FOUND, result);
}
```

사용자가 검색어 입력 또는 태그 선택시 해당 검색조건에 맞는 팝업스토어 리스트업

Header 컴포넌트에서 검색어를 받으면 해당 키워드로 테이블을 조회 후 검색한 팝업스토어들의 PK를 이용해 해당 스토어들의 리뷰평점을 DB에서 가져온 후 HashMap에 담아서 ApiResponse 객체를 이용해 반환후 EventListComponent로 navigate State로 데이터 전송 후 리스트업, 검색어 없이 태그만 선택시 해당 태그의 해당하는 업체만 쿼리해 리스트업

POPSPOT 기능별 구현 로직 & UI 흐름 설명

팝업 리스트 좋아요 기능



코드소개

```
function LikeCount ({no,userId}){
  const [likes,setLikes] = useState(false);
  useEffect(()=>{
    axios.get(`/api/event/like/${no}/${userId}`)
      .then(result => {setLikes(result.data)})
      .catch( err => console.error
        ("좋아요 상태를 불러오는 중 오류가 발생했습니다.",err));},[])

  return(
    <
      <span
        style={{
          cursor: 'pointer',
          fontSize: '24px',
          color: likes ? 'red' : 'gray', // 좋아요 여부에 따라 색상 변경
        }} >
        {likes ? '♥' : ''}
      </span>
    </
  )
}
```

서버에서 데이터를 다시 가져오지 않고, 현재 상태를 기반으로 계산하여
좋아요 수(likeNo)를 빠르게 업데이트하여 성능을 향상

주요 동작

좋아요 기능을 컴포넌트로 구현하여 재사용성과 유지보수성을 높임.

사용자가 좋아요 버튼을 클릭하면 userId와eventNo를 서버로 전송 하고 저장/삭제를 하고

좋아요 개수를 실시간으로 1증가/감소

좋아요 여부에 따라 하트 색상이 동적으로 변경되며, 비동기 데이터 처리를 통해 빠르고

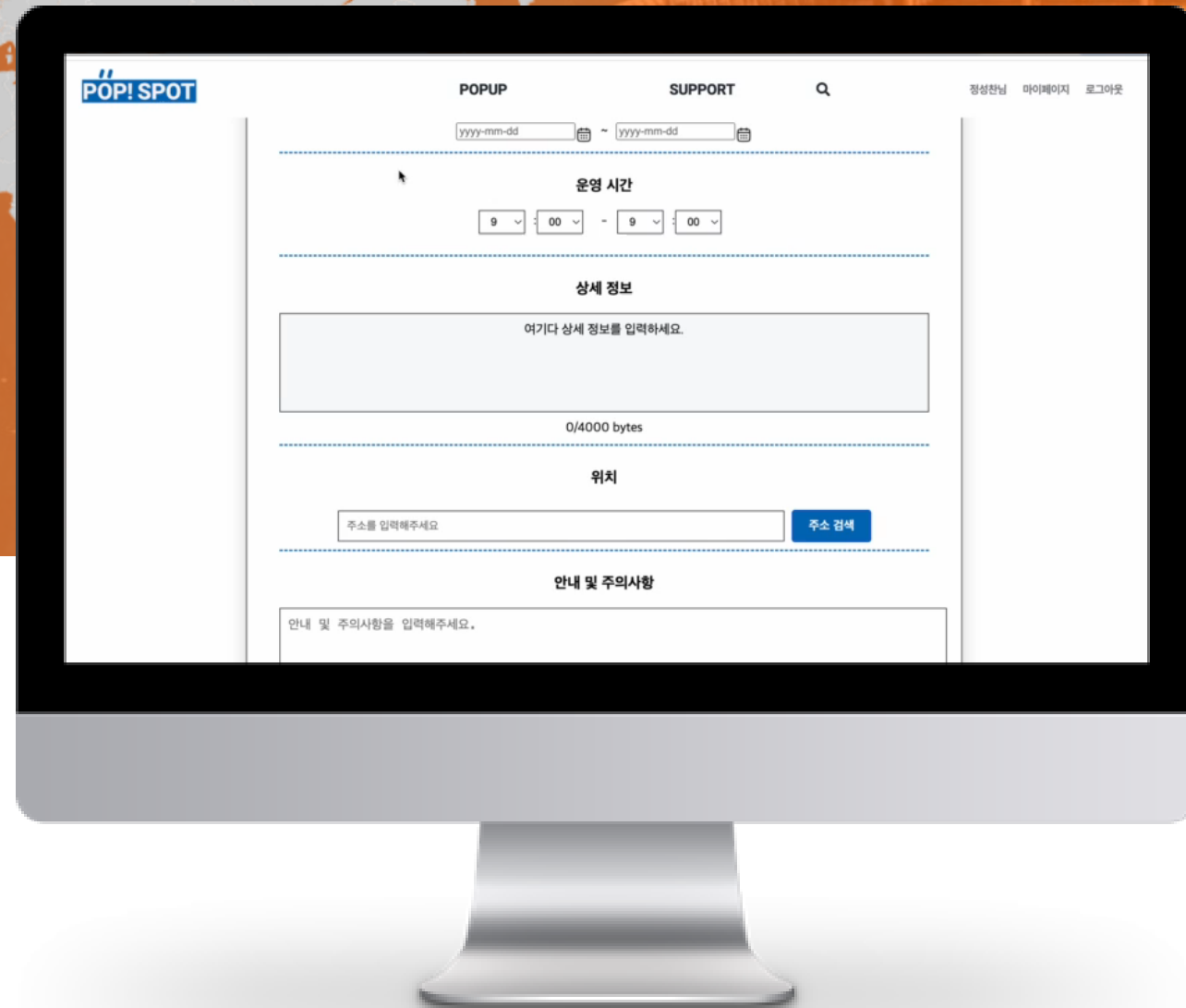
정확하게 상태를 업데이트

지도 API

검색 창을 열고, 사용자가 선택한 주소 데이터를 가져옴. 선택된 주소는 address와 eventData에 저장. 이후, Kakao지도 API의 Geocoder를 사용하여 주소를 위도(latitude)와 경도(longitude)로 변환. 변환된 좌표를 기준으로 지도의 중심을 업데이트하고, 해당 위치에 마커를 표시. 지도가 제대로 렌더링될 수 있도록 map.layout()을 호출하여 레이아웃을 재정렬

코드소개

```
const searchAddress = () => {  
  new window.daum.Postcode({  
    onComplete: function (data) {  
      const addr = data.address;  
      setAddress(addr);  
      setEventData({...eventData, address:addr});  
      const {map, marker} = window.kakaoMapData;  
      const geocoder = new window.kakao.maps.services.Geocoder();  
      geocoder.addressSearch(addr, function (results, status) {  
        if (status === window.kakao.maps.services.Status.OK) {  
          const result = results[0];  
          const coords = new window.kakao.maps.LatLng(result.y, result.x);  
          setEventData((prev) => ({  
            ...prev,  
            lat: result.y,  
            lon: result.x,  
          }));  
          document.getElementById('map').style.display = 'block';  
          map.layout();  
          map.setCenter(coords);  
          marker.setPosition(coords);  
        }  
      });  
    },  
  }).open();  
};
```



POPSPOT 기능별 구현 로직 & UI 흐름 설명

지도 API

Kakao지도 API를 활용해 특정 위치를 지도에 표시하는 React 컴포넌트
lat(위도), lon(경도), address를 props로 받아 지도를 설정하고, 클릭 시
Kakao지도 웹페이지에서 해당 위치를 검색 가능 지도 로직을 분리해 재사용성과
관리 편의성을 높이기 위해 컴포넌트로 따로 제작

코드소개

```
function KakaoMap({lat, lon, address}) {  
  useEffect(() => {  
    const container = document.getElementById('map');  
    const options = {  
      center: new window.kakao.maps.LatLng(lat, lon),  
      level: 3,  
    };  
    const map = new window.kakao.maps.Map(container, options);  
    const marker = new window.kakao.maps.Marker({  
      position: new window.kakao.maps.LatLng(lat, lon),  
    });  
    marker.setMap(map);  
    window.kakao.maps.event.addListener(map, "click", () => {  
      const kakaoLink = `https://map.kakao.com/link/search/${  
        encodeURIComponent(address)}`;  
      window.open(kakaoLink, "_blank");  
    });  
  }, []);  
  return (  
    <div id="map" style={{ width: '100%', height: '350px', marginTop: '10px', }} >  
      </div>  
  )  
}  
export default KakaoMap;
```



Redis 활용 인기 조회수 조회

```
// 컨트롤러
@GetMapping("views/{eventNo}/increment")
public ApiResponse incrementViewCount(@PathVariable("eventNo") String eventNo) {
    return redisService.incrementViewCount(eventNo);
}

//서비스
public ApiResponse incrementViewCount(String eventNo) {
    try {
        Double newCount = redisRepository.incrementViewCount(eventNo);
        int intCount = (newCount != null) ? newCount.intValue() : 0; // null인 경우 0으로 처리
        return ApiResponse.apiBuilder(true, "조회수 증가 성공", intCount);
    } catch (Exception e) {
        System.err.println("RedisService Error (incrementViewCount): " + e.getMessage());
        return ApiResponse.apiBuilder(false, "조회수 증가 실패");
    }
}

//repositoryImpl
private static final String REDIS_SORTED_SET_KEY = "view";
private final ZSetOperations<String, String> zSetOperations;
@Autowired
public RedisRepositoryImpl(RedisTemplate<String, String> redisTemplate) {
    this.zSetOperations = redisTemplate.opsForZSet();
}
```



Redis를 사용한 인기조회수들의 빠른 조회 및 정렬

Redis를 활용해 조회수를 실시간으로 업데이트 하며, Sorted Set을 사용하여 빠르고 효율적으로 인기항목을 정렬하고 관리



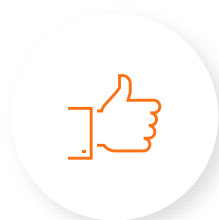
병목현상 방지

Oracle DB는 강력한 트랜잭션 관리와 영구 데이터 저장에 적합하지만, 실시간 데이터 처리 및 대량의 읽기/쓰기 작업에서는 병목현상을 방지하기 위해 Redis를 사용

POPSPOT 기능별 구현 로직 & UI 흐름 설명

응답 처리 방식 개선

View 단에서 HTTP 요청에 대한 응답을 처리할 때, 각 API마다
응답에 반환되는 데이터 타입이 달라 일관된 접근 방식으로
데이터를 처리하기 어려움



공통적인 응답 생성 로직

응답 데이터를 일관성 있게 관리하기 위해 ApiResponse DTO를 생성.
ApiResponse를 반환값으로 갖는 ResponseEntity 객체를 생성하기 위해
DTO Builder 클래스를 설계하고 공통적인 응답 생성 로직을
ResponseBuilder 클래스로 분리



API 응답 처리의 일관성 문제 해결

요청에 대한 응답을 View단에서 처리할 때,
각 API에 대해서 동일한 접근 방식을 사용하지 못하는 이슈가 발생하여
동일한 방법으로 모든 Http Response 데이터에 접근하기 위한 방법을 고민

```
@PostMapping("/check/{userId}")
public ResponseEntity<ApiResponse> checkUsername(@PathVariable(name="userId") String userId) {
    ApiResponse res = us.checkDuplicatedId(userId);
    return rb.buildNoContentResponse(res, HttpStatus.CONFLICT);
}

public ApiResponse checkDuplicatedId(String userId) {
    boolean isDuplicated =

    ...

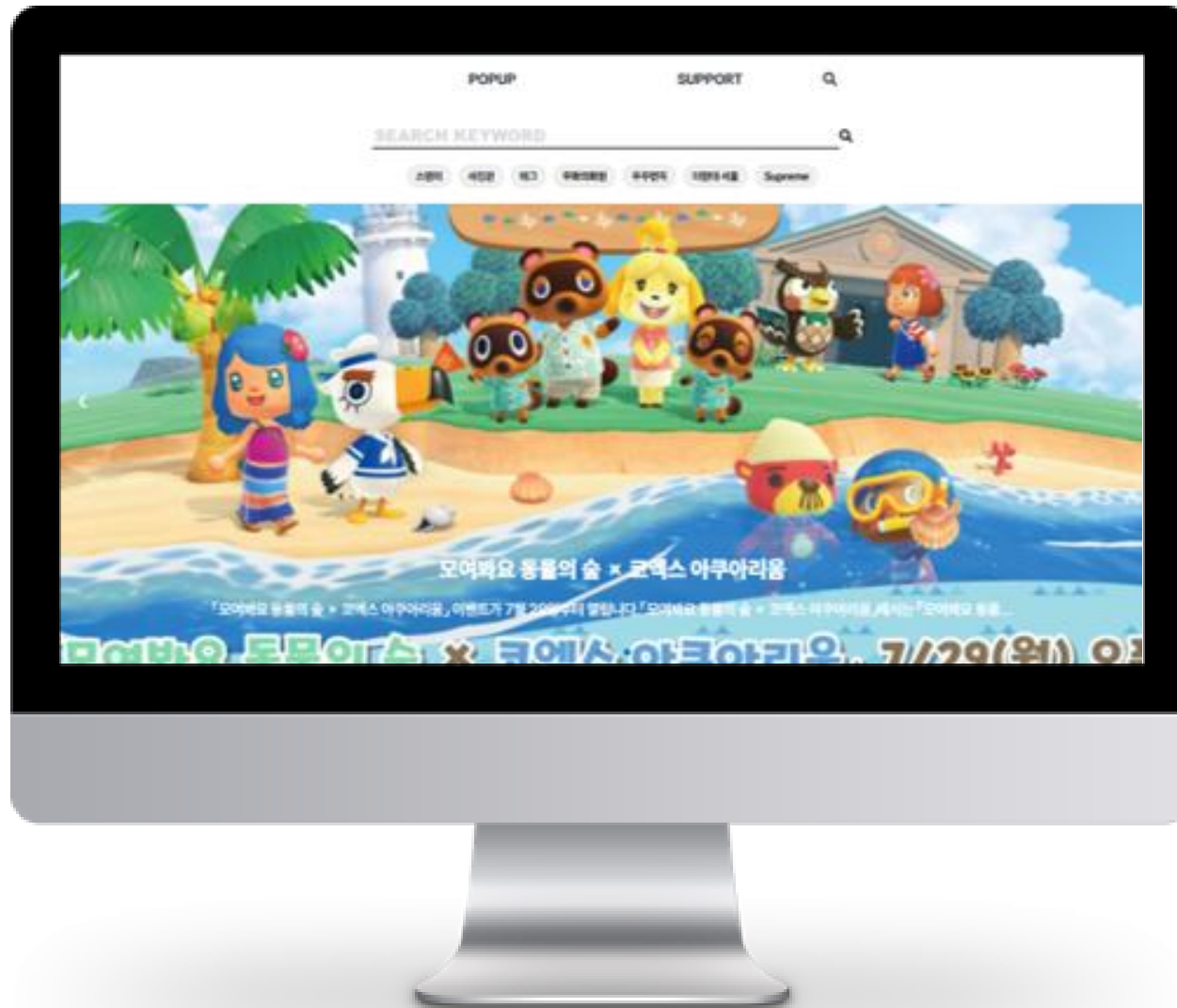
    ApiResponse.apiBuilder(false, "중복되는 아이디입니다.")ApiResponse.apiBuilder
    (true, "사용 가능한 아이디입니다.");
}


public static ApiResponse apiBuilder(boolean state, String message) {
    return ApiResponse.builder()
        .success(state)
        .message(message)
        .build();
}

public ResponseEntity<ApiResponse> buildNoContentResponse(ApiResponse res, HttpStatus
defaultStatus){
    HttpStatus stat = res.isSuccess() ? HttpStatus.NO_CONTENT : defaultStatus;
    return ResponseEntity.status(stat).body(res);
}
```

기능 시연

MEDIPACS



A person in a suit is shown from the chest down, holding a pen in their right hand and a notepad in their left hand. The background is a blurred office setting with a desk and a glass of water. The entire image is overlaid with a semi-transparent orange filter.

Q&A