OH! 조

팀원 소개

BE :

- **최혜미(팀장)**

  회원가입, 회원탈퇴, 이메일인증, 상세페이지 조회, 마이페이지 조회 및 수정, 사용자 리뷰 불러오기

- **이유상**

  spring security ,장바구니 및 결제 관련 crud, 예약 조회 관련, 프로젝트 배포

- **백인권**

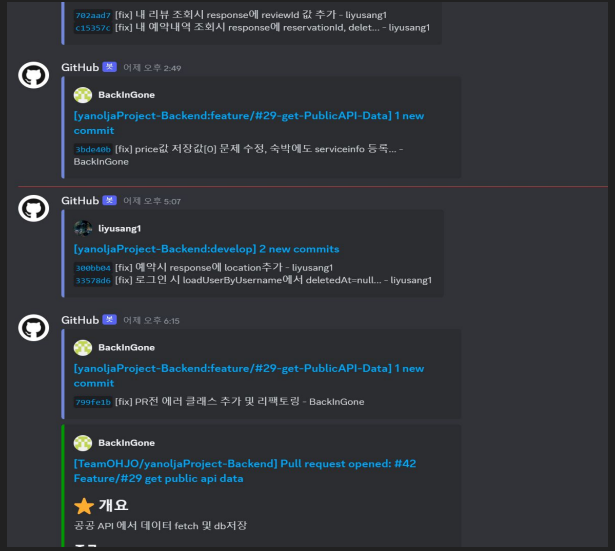
  메인페이지, 오픈 api

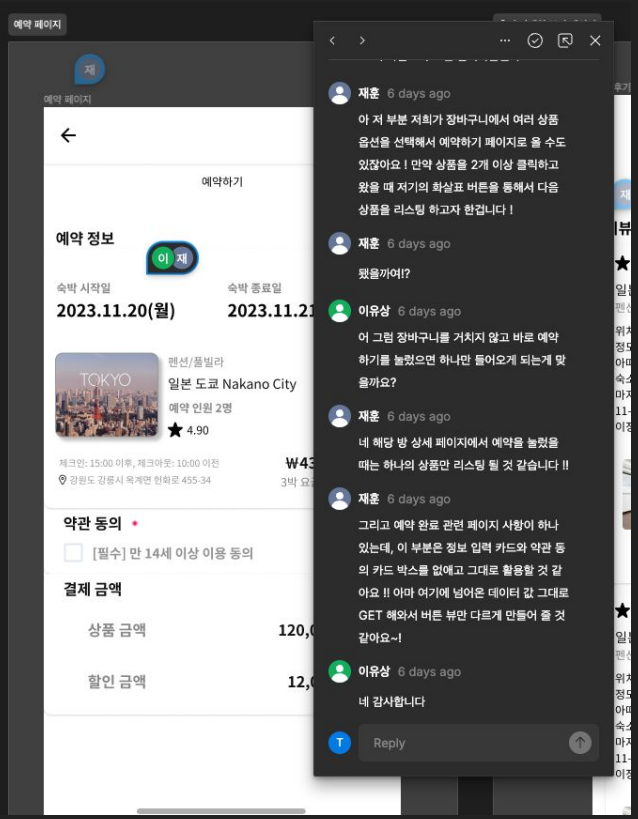
- **안수지**

  리뷰 crud , 좋아요 , 위시리스트

협업 과정

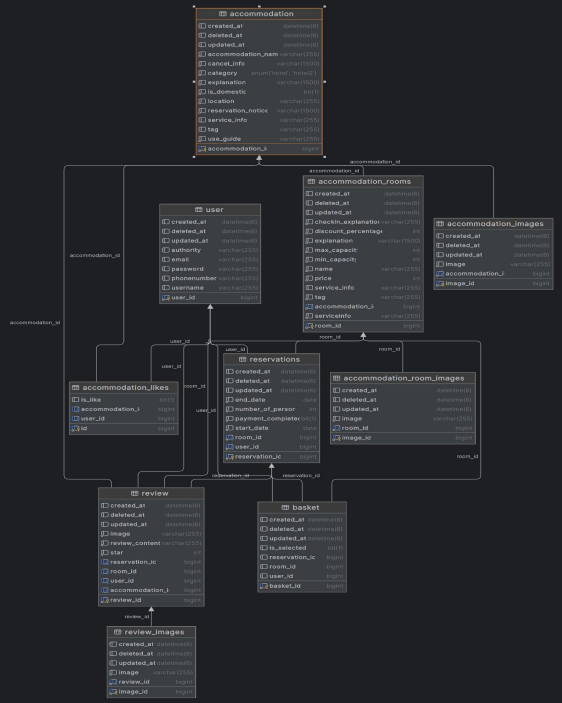
- githook 적용해 디스코드와 연동
- FE & BE 코드컨벤션 작성 및 공유
- 매일 5시 FE & BE 정기회의
- BE : 포스트맨 사용하여 api 공유
- FE : 피그마 사용하여 와이어프레임 공유



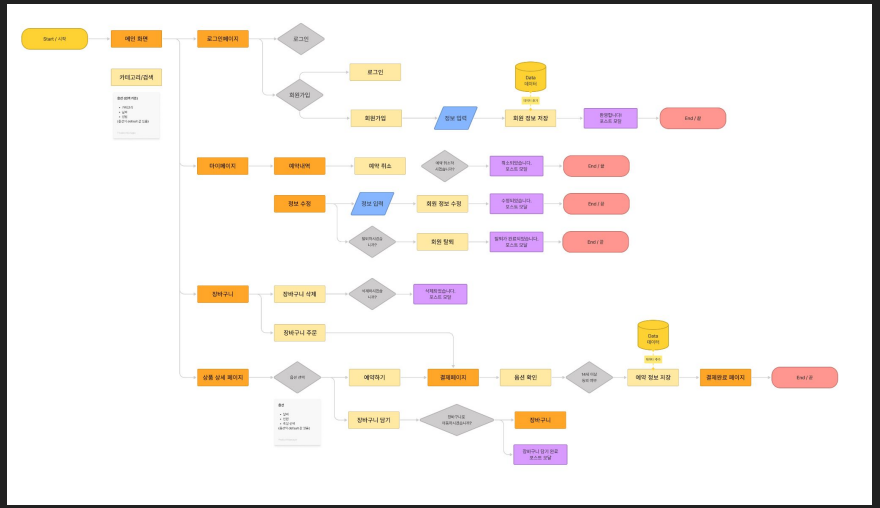


기획 과정

ERD 작성해 프론트와 공유



FLOWCHART 작성



FE & BE 함께 요구사항 작성

우선순위	1차 분류	2차 분류 - 페이지	서비스	필요기능	기능설명
2	회원가입	회원가입 (1/3)	약관 동의	약관동의	체크박스
2		회원가입 (2/3)	이메일 인증	이메일 인증	
1				이메일 중복 확인	
1		회원가입 (3/3)	정보 등록	정보 입력	이름, 전화번호, 비밀번호, 비밀번호 확인
				정보 유효성 검사	이름 5자 이내, 전화번호 형식, 비밀번호/ 비밀번호 확인 일치 여부, 비밀번호 형식 지정
1				PW암호화	passwordEncoder을 통한 비밀번호 해시화
1				회원 db 등록	
1	로그인		로그인	이메일,pw 확인	
				세션생성	
				인증 유효성 검사	
2	마이페이지	내 정보 수정	내 정보 수정	PW수정	
2				이름 수정	
2				회원 탈퇴	
1		예약내역	예약 내역	예약 히스토리 리스팅	
				예약 취소 기능	
2		내 후기 내역	내 후기 내역	후기 작성하기	
2				후기 히스토리 리스팅	
2		포인트 확인	포인트 확인	포인트 적립 히스토리 리스팅	
1	메인페이지		상품(숙박) 목록	일부 카테고리 별 상품 리스팅	
2				카테고리 선택	
1				품절상품 처리	
1				오픈api를 통한 검증	
1	검색			상품 필터링	날짜/숙박인원/지역/가격
1	장바구니		장바구니 목록	상품 리스팅	
				장바구니 삭제(체크박스)	
				상품 주문(체크박스)	
1	주문(예약)			예약등록	
			결제	만 14세 이상 이용 동의(체크박스)	약관동의
				결제	
				예약취소	
1	상품 상세			상품 옵션 선택	
2				후기 조회	
1				장바구니 담기	
				날짜, 숙박인원 선택	
2				찜(좋아요)	
1	주문(예약) 결과 확인			결제 완료 결과 출력	
1	상품 리스트	상품 리스트	상품(숙박)목록	카테고리 별 전체 상품 리스팅	
				카테고리 분리	
				상품 정렬	가격 높은 순 / 가격 낮은 순 / 거리 순

BE 기술 스택

📌기술스택 & 구현환경

- Java : java 17
- FrameWork : springboot 3.1.5 springsecurity spring data JPA spring web
- Build : Build Gradle
- VCS : VCS Git Github
- Database : Database GCP Cloud SQL
- DBMS : DBMS MySQL

🔗 배포환경 : 배포 환경 GCP VM ubuntu 20

📌컨벤션

- Code Convention IntelliJ Java Google Style
- GitFlow Workflow

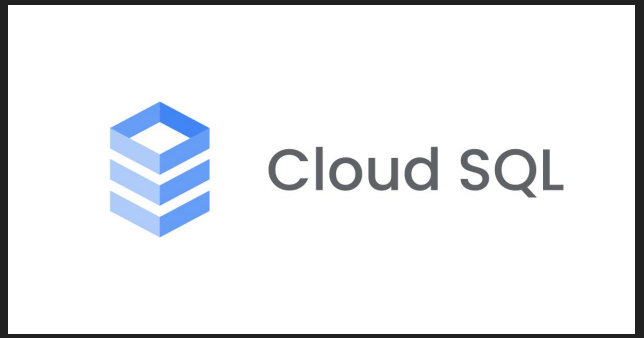
📌패키지 구조

```
com.example.yanolja
├── domain
│   ├── user
│   ├── accommodation
│   ├── review
│   ├── reservation
│   ├── accommodationLikes
│   ├── basket
│   ├── wishlist
│   ...
└── global
    ├── springsecurity
    ├── entity
    ├── config
    ├── exception
    ├── jwt
    └── util
```

배포 과정

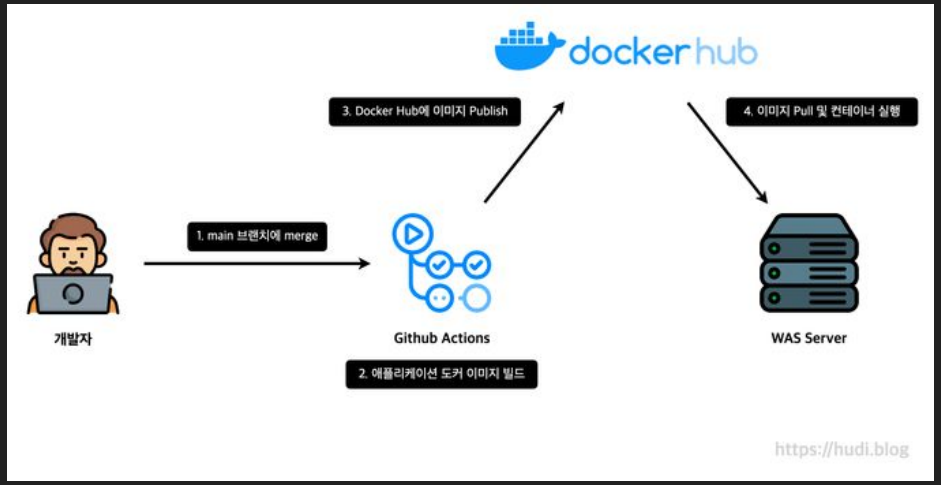
생성한 도커이미지를 도커허브에 올린 후
GCP VM에서 pull해서 작성한 docker-compose.yml 파일을 바탕으로 배포
DB는 GCP cloud SQL에서 MySQL dbms을 활용해서 관리 및 저장

도커 이미지로 배포하였기에 어디서나 동일한 환경의 프로젝트를 실행 할 수 있고
빠른 배포 및 확장이 매우 용이 함



CI/CD

GitHub Actions 워크플로우 작성을 통해 자동화




```yaml
name: Backend CD
on:
  push:
    branches: [ main ]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - name: 저장소 Checkout
        uses: actions/checkout@v3

      - name: 스프링부트 애플리케이션 빌드 # (1)
        run: ./gradlew build

      - name: 도커 이미지 빌드 # (2)
        run: docker build -t <docker_hub_username>/<image_name>

      - name: Docker Hub 로그인 # (3)
        uses: docker/login-action@v2
        with:
          username: ${{ secrets.DOCKERHUB_USERNAME }}
          password: ${{ secrets.DOCKERHUB_TOKEN }}

      - name: Docker Hub 퍼블리싱 # (4)
        run: docker push <docker_hub_username>/<image_name>

      - name: WAS 인스턴스 접속 및 애플리케이션 실행 # (5)
        uses: appleboy/ssh-action@v0.1.6
        with:
          host: ${{ secrets.WAS_HOST }}
          username: ${{ secrets.WAS_USERNAME }}
          password: ${{ secrets.WAS_PASSWORD }}
          port: ${{ secrets.WAS_SSH_PORT }}
          script: |
            docker stop $(docker ps -a -q)
            docker rm $(docker ps -a -q)
            docker pull <docker_hub_username>/<image_name>
            docker compose up ..
```

util 클래스 사용

1. 예외처리 util

```
// USER
USER_NOT_FOUND(HttpStatus.NOT_FOUND, message: "존재하지 않는 회원입니다."),
USER_ALREADY_REGISTERED(HttpStatus.BAD_REQUEST, message: "이미 가입된 회원입니다."),
INVALID_PHONENUMBER(HttpStatus.LENGTH_REQUIRED, message: "유효하지 않은 휴대폰 번호 형식"),
INVALID_EMAIL(HttpStatus.PRECONDITION_FAILED, message: "유효하지 않은 이메일 형식"),
INVALID_PASSWORD(HttpStatus.BAD_REQUEST, message: "잘못된 비밀번호입니다."),

//Reservation
RESERVATION_CONFLICT(HttpStatus.BAD_REQUEST, message: "예약하려는 시간대에 예약이 존재합니다."),
INVALID_CANCEL_RESERVATION_REQUEST(HttpStatus.BAD_REQUEST, message: "예약이 존재하지 않거나 이미 취소된
INVALID_RESERVATION_ID(HttpStatus.BAD_REQUEST, message: "존재하지 않는 reservationId입니다."),

//Basket
DUPLICATED_BASKET_CONTENT(HttpStatus.BAD_REQUEST, message: "이미 장바구니에 같은 상품이 있습니다."),

//Review
REVIEW_NOT_FOUND(HttpStatus.NOT_FOUND, message: "존재하지 않는 리뷰입니다."),
INVALID_BASKET_ID(HttpStatus.BAD_REQUEST, message: "존재하지 않는 장바구니입니다."),

//ACCOMODATIONLIKES
INVALID_ARGUMENT(HttpStatus.BAD_REQUEST, message: "잘못된 사용자 ID 또는 숙소 ID입니다."),

//Accomodation
ACCOMMODATION_NOT_FOUND(HttpStatus.NOT_FOUND, message: "숙소를 찾을 수 없습니다."),

//AccomodationRooms
INVALID_ACCOMMODATION_ID(HttpStatus.NOT_FOUND, message: "존재하지 않는 방입니다."),

//Permission
PERMISSION_DENIED(HttpStatus.BAD_REQUEST, message: "해당 작업을 수행 할 권한이 존재하지 않습니다."),

INTERNAL_SERVER_ERROR(HttpStatus.INTERNAL_SERVER_ERROR, message: "서버 내부 에러"),

//APIService
JSON_PARSING_ERROR(HttpStatus.BAD_REQUEST, message: "JSON 파싱 오류가 발생했습니다."),
API_CALL_FAILURE(HttpStatus.INTERNAL_SERVER_ERROR, message: "외부 API 호출에 실패했습니다."),
API_PROCESSING_ERROR(HttpStatus.INTERNAL_SERVER_ERROR, message: "API 처리 중 오류가 발생했습니다.");


private HttpStatus httpStatus;
private String message;

38 usages  hym
ErrorCode(HttpStatus httpStatus, String message) {
    this.httpStatus = httpStatus;
    this.message = message;
}
}
```

2. 시간처리 util

```
18 usages  9 inheritors  liyusang1 *
@Getter
@MappedSuperclass
@EntityListeners(AuditingEntityListener.class)
public abstract class BaseTimeEntity {

    @CreatedDate
    protected LocalDateTime createdAt;

    @LastModifiedDate
    protected LocalDateTime updatedAt;

    @Column(insertable = false)
    protected LocalDateTime deletedAt;

    2 overrides  liyusang1
    protected void delete(LocalDateTime currentTime) {
        if (deletedAt == null) {
            deletedAt = currentTime;
        }
    }

    no usages  liyusang1
    public boolean isDeleted() { return deletedAt != null; }

    2 usages  1 override  liyusang1
    protected void restore() { deletedAt = null; }
}
```

3. response 통일

```
hym +1
@Getter
public class ResponseDTO<T> {

    private final int code;
    private final String message;
    private final T data;

    no usages  hym +1
    @Builder
    private ResponseDTO(int code, String message, T data) {
        this.code = code;
        this.message = message;
        this.data = data;
    }

    hym
    public static ResponseDTO<Object> res(final HttpStatus httpStatus, final String message) {
        return ResponseDTO.builder()
                .code(httpStatus.value())
                .message(message)
                .build();
    }

    hym
    public static <T> ResponseDTO<T> res(final HttpStatus httpStatus, final T data) {
        return ResponseDTO.<T>builder()
                .code(httpStatus.value())
                .data(data)
                .build();
    }

    hym
    public static <T> ResponseDTO<T> res(final HttpStatus httpStatus,
        final String message,
        final T data) {
        return ResponseDTO.<T>builder()
                .code(httpStatus.value())
                .message(message)
                .data(data)
                .build();
    }
}
```

issue

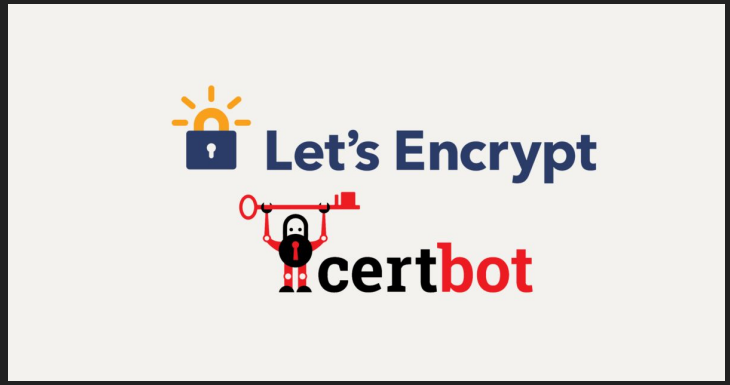
서로다른 도메인간 쿠키 저장 불가 이슈

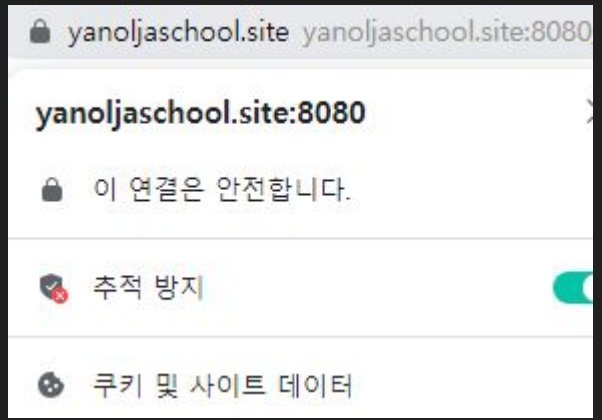
same site : none

set.domain()

set.cookie (secure) -> ssl 인증을 통한 https 필요

로그인시 토큰을 response하고, api 요청시 헤더를 통해 쿠키를 보내는 방식으로 변경





이 삽질을 통해 spring에서 ssl인증서를 등록해

https로 response를 보내는 방법을 알았고

서로 다른 도메인간 쿠키 공유에 있어서 여러가지 설정 값이 필요한것임을 배움

리팩토링

1. redis 적용

캐시 db (레디스)를 활용해
자주 사용되는 데이터들을 담아 조회 성능 개선

docker 이미지로 배포하고 있으므로
docker에 redis를 띄워서 해볼 예정

2. 소셜로그인

OAuth 2.0을 이용해 소셜 로그인 추가 구현해 보기

3. 기존 코드 리팩토링

테스트코드, 예외처리 추가 등 기존 코드 리팩토링 예정

4. query dsl

```
@Query("SELECT r FROM Reservations r " +
    "WHERE r.room.roomId = :roomId " +
    "AND (:startDate <= r.endDate AND :endDate>= r.startDate " +
    "OR :startDate >= r.startDate AND :endDate <= r.endDate) " +
    "AND r.paymentCompleted = true AND r.deletedAt is NULL")
Optional<Reservations> findConflictingReservations(@Param("roomId") Long roomId,
    @Param("startDate") LocalDate startDate,
    @Param("endDate") LocalDate endDate);
```

Querydsl을 사용하여 프로젝트에서 사용한 쿼리를 개선
Querydsl은 타입 안전한 쿼리를 생성하고,
코드 기반으로 쿼리를 작성할 수 있는 라이브러리

-> 주어진 JPQL 쿼리를 Querydsl을 사용하여 개선할 예정

프론트엔드 부분

팀원 소개

FE :

- **김특희**(팀장)

  : 메인페이지 구현, 내 정보 관리 구현

- **진정민**

  : 숙소/객실 상세 페이지, 내 리뷰 구현

- **김다빈**

  : 위시리스트, 장바구니, 예약내역 구현

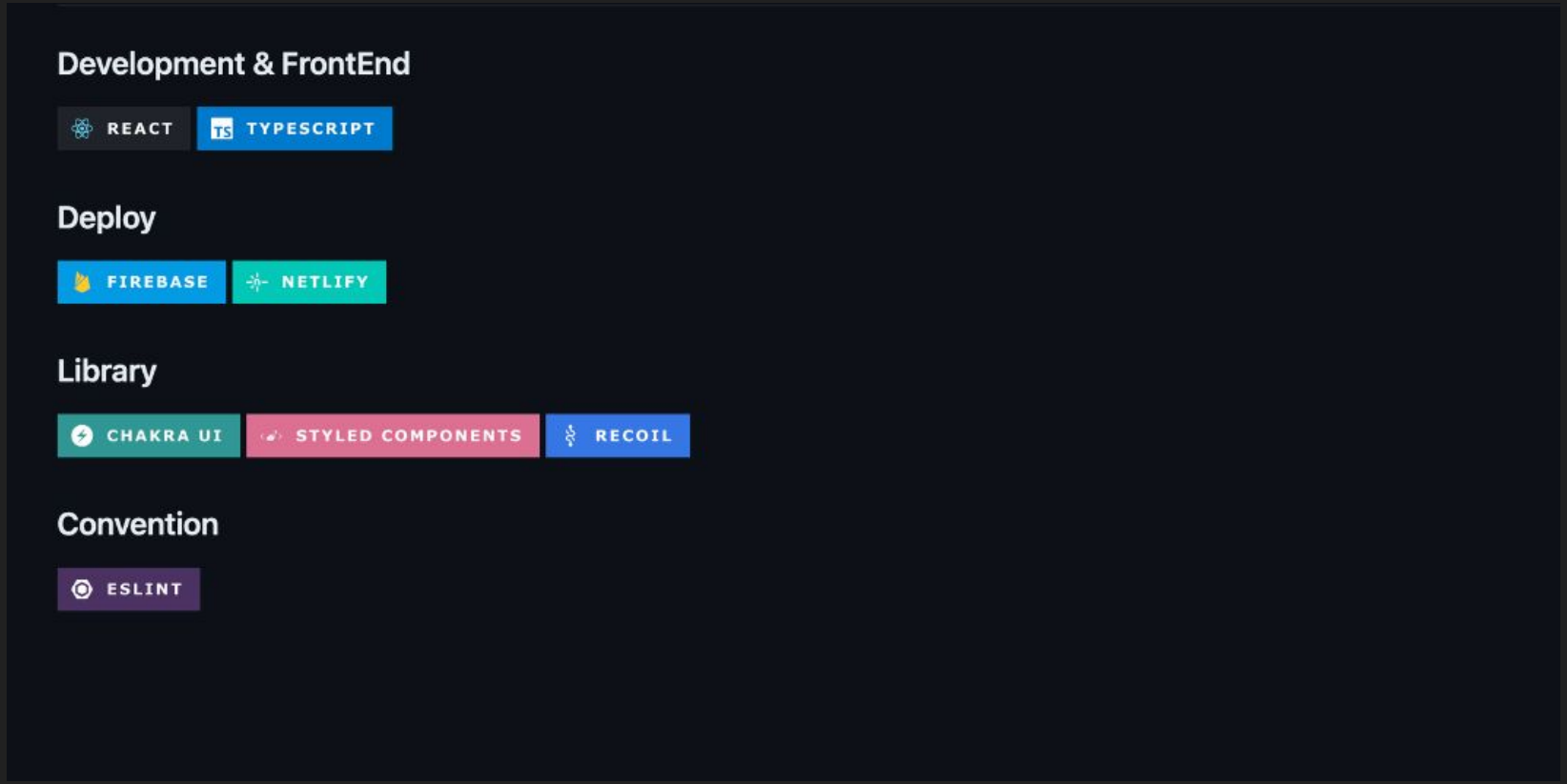
- **최재훈**

  : 디자인, 예약하기, 결제페이지 구현

- **신하연**

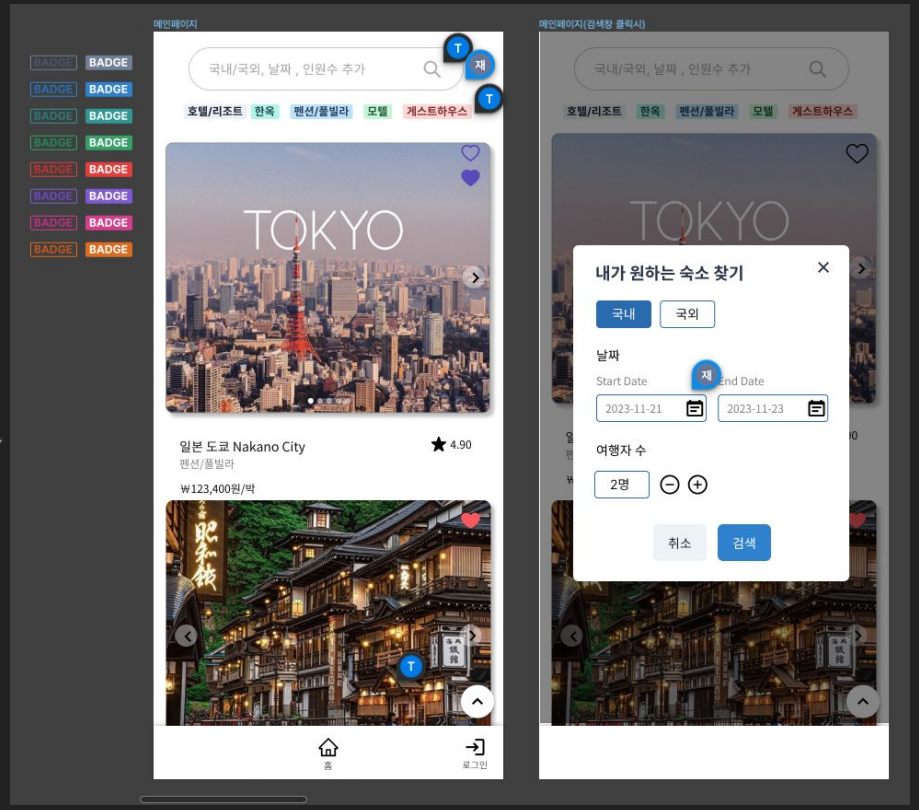
  : 로그인/회원가입 구현

프론트엔드 기술스택


Development & FrontEnd
REACT
TYPESCRIPT
Deploy
FIREBASE
NETLIFY
Library
CHAKRA UI
STYLED COMPONENTS
RECOIL
Convention
ESLINT

자랑거리 - 요구사항 분석 & 꼼꼼한 기획

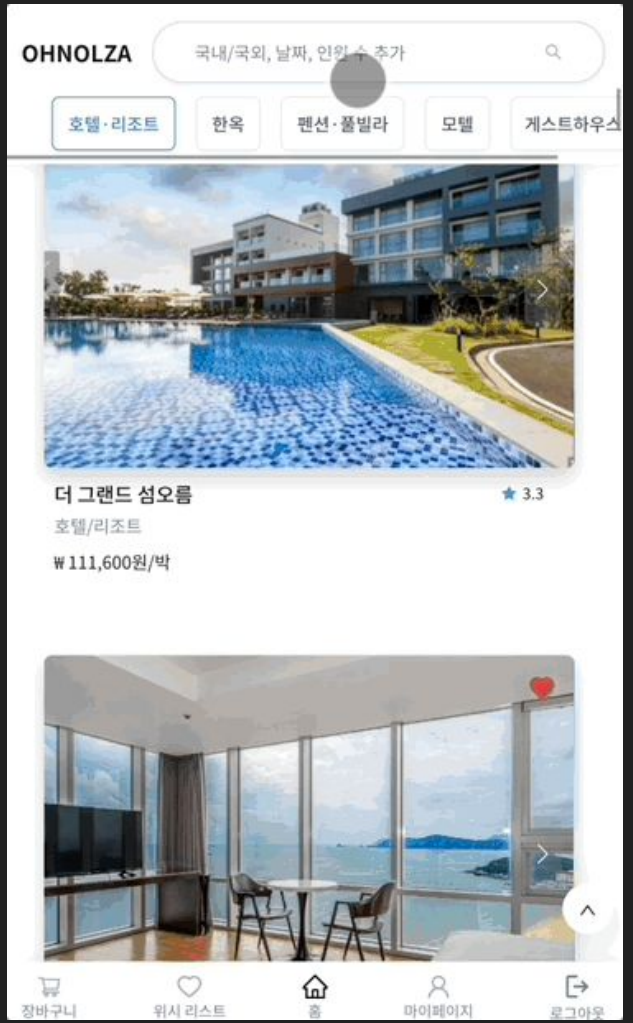
(와이어 프레임)



(디자인 시안)

다양한 옵션 필터링을 통한 숙소 리스팅 구현

: 국내/국외, 날짜 별,
인원 수를 통한 검색 필터링



좋아요, 마이페이지, 지도, 내 정보 관리