

22/04/21 ~ 22/05/12

숙박플렛
폼

Spring
GOING

01

개발 환경

02

사용 기술-백엔드

03

사용 기술-프론트

04

협업

05

핵심 키워드

06

설계

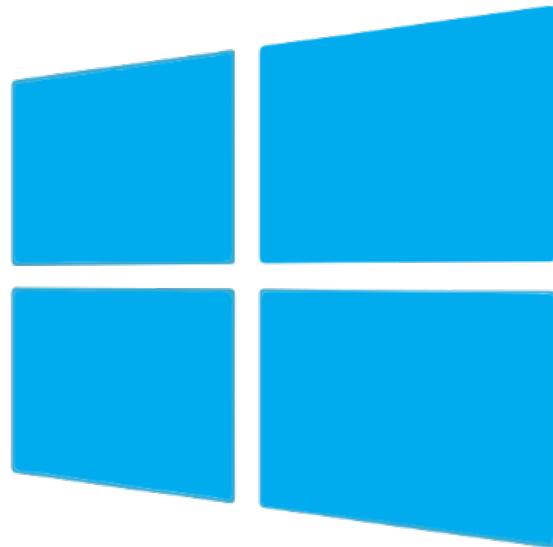
07

핵심 기능

01

. 개발 환경

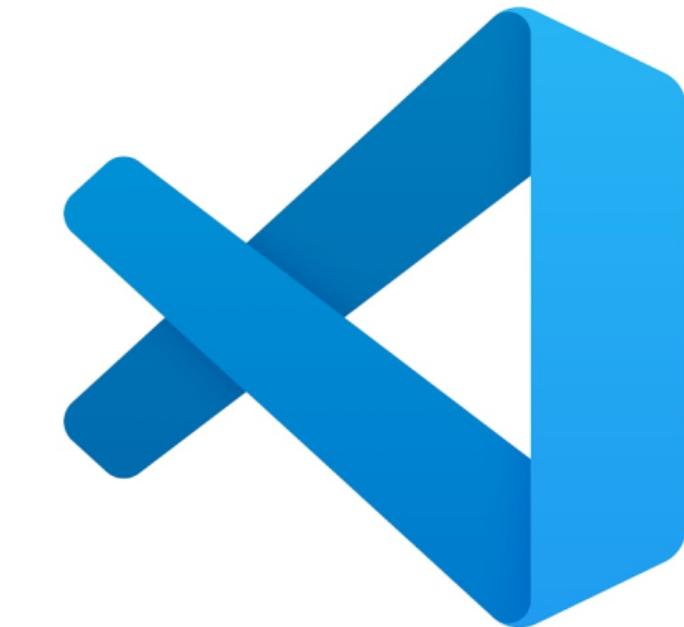
1. 개발환경



Window



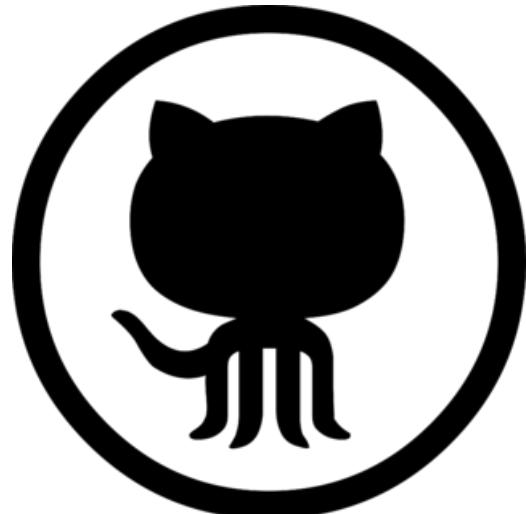
MAC OS



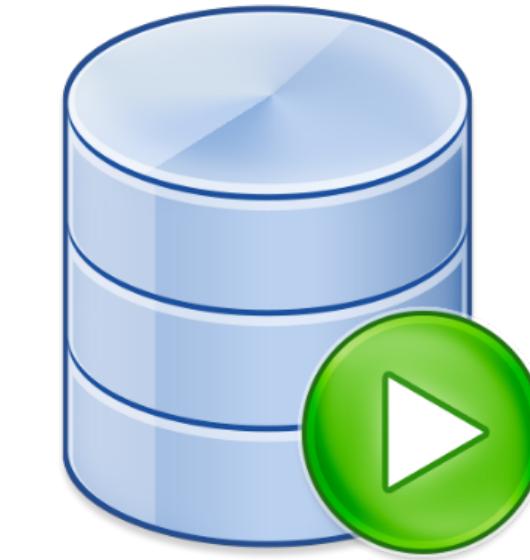
VSCode



Eclipse



GitHub



SQL Developer

02. 사용 기술-백엔드

프레임워크 & 라이브러리

-
- Java 11
 - Spring MVC
 - Spring AOP
 - Spring Security
 - JUnit4
 - MyBatis

Database

-
- Oracle
 - Oracle Cloud

WAS

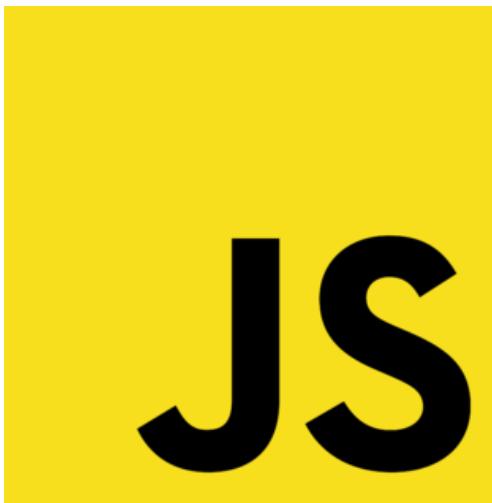
-
- Apache Tomcat

03. 사용 기술-프론트

HTML



CSS



 **jQuery**



HTML

CSS

JavaScript

jQuery

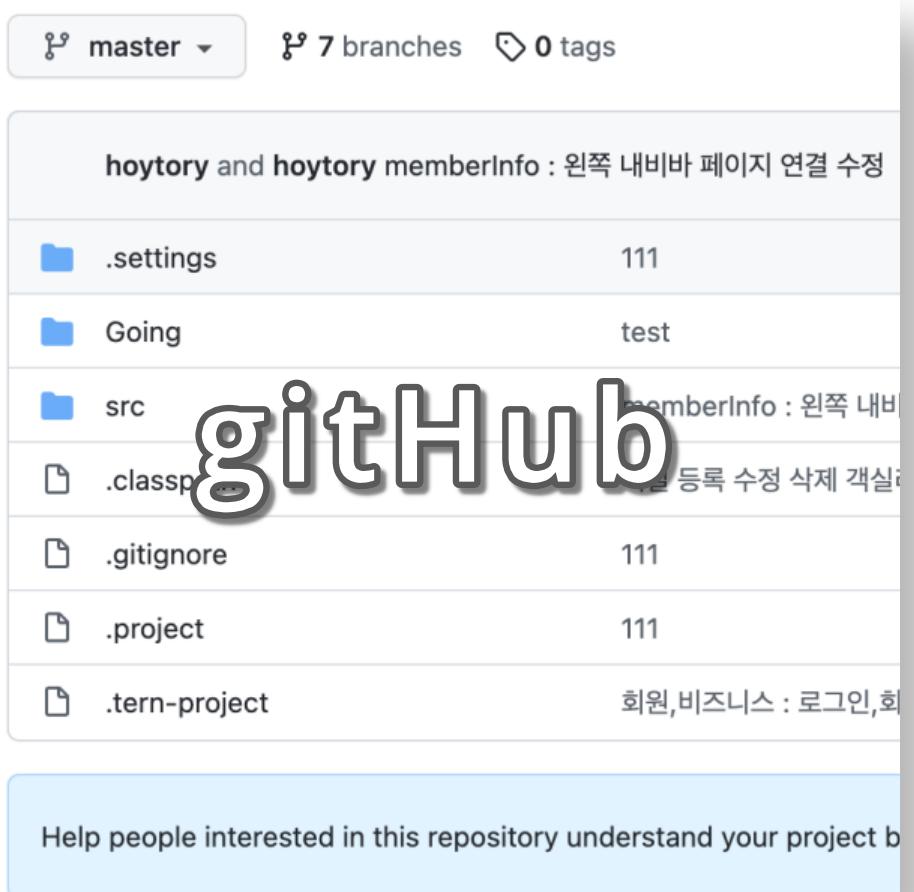
Bootstrap5

04.

협업

코드 관리

공용 Repository 생성 후
Colaborators 기능으로 초대,
함께 코드 관리



일정 관리

개인, 공동의 목표를 설정 후 관리



설계

실시간 참여가 가능한
디자인 커뮤니케이션 툴을 이용해
추상적이었던 프로젝트를
시각적 자료로 제작

프로토타입

➡ Database ERD
Whimsical
☰ 핵심 로직

flowchart

05.

핵심 키워드

1

전반적인 프로젝트 제작 과정 경험

직접 설계부터 완성까지 만들어보며 부족하고 보완해야 할 점을 찾아볼 수 있는 계기가 되었습니다.
잘못된 설계가 추후 큰 파장을 불러오는 것을 느끼고 프로젝트 제작 절차에 대한 중요성을 느꼈습니다.

2

Spring을 학습하기 전 MVC 패턴에 대한 깊은 공부

MVC 패턴인 Model2를 기반으로 프로젝트를 구성해나가며 Model, View, Controller 세 구조가
어떻게 작동하고 상호작용하는지 직접적인 경험을 통해 배울 수 있었습니다.

3

다양한 기능 사용으로 폭넓은 경험 쌓기

많은 기능을 구현하고 싶었기에 그만큼 다양한 기능들을 배우며 사용함으로써
프레임워크 / 라이브러리 / API 사용에 대한 노하우를 습득할 수 있었습니다.

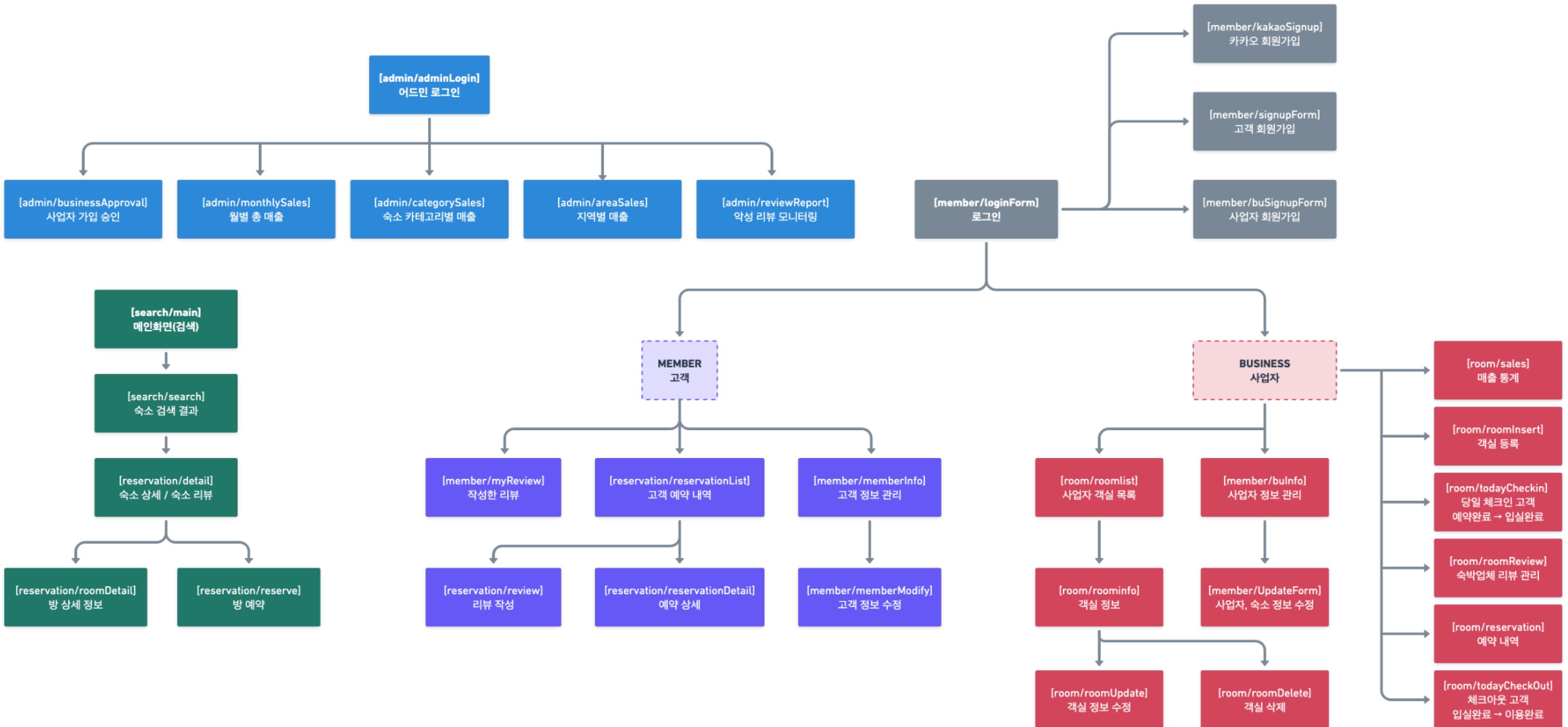
4

다양한 경우의 수가 존재하는 예약 서비스

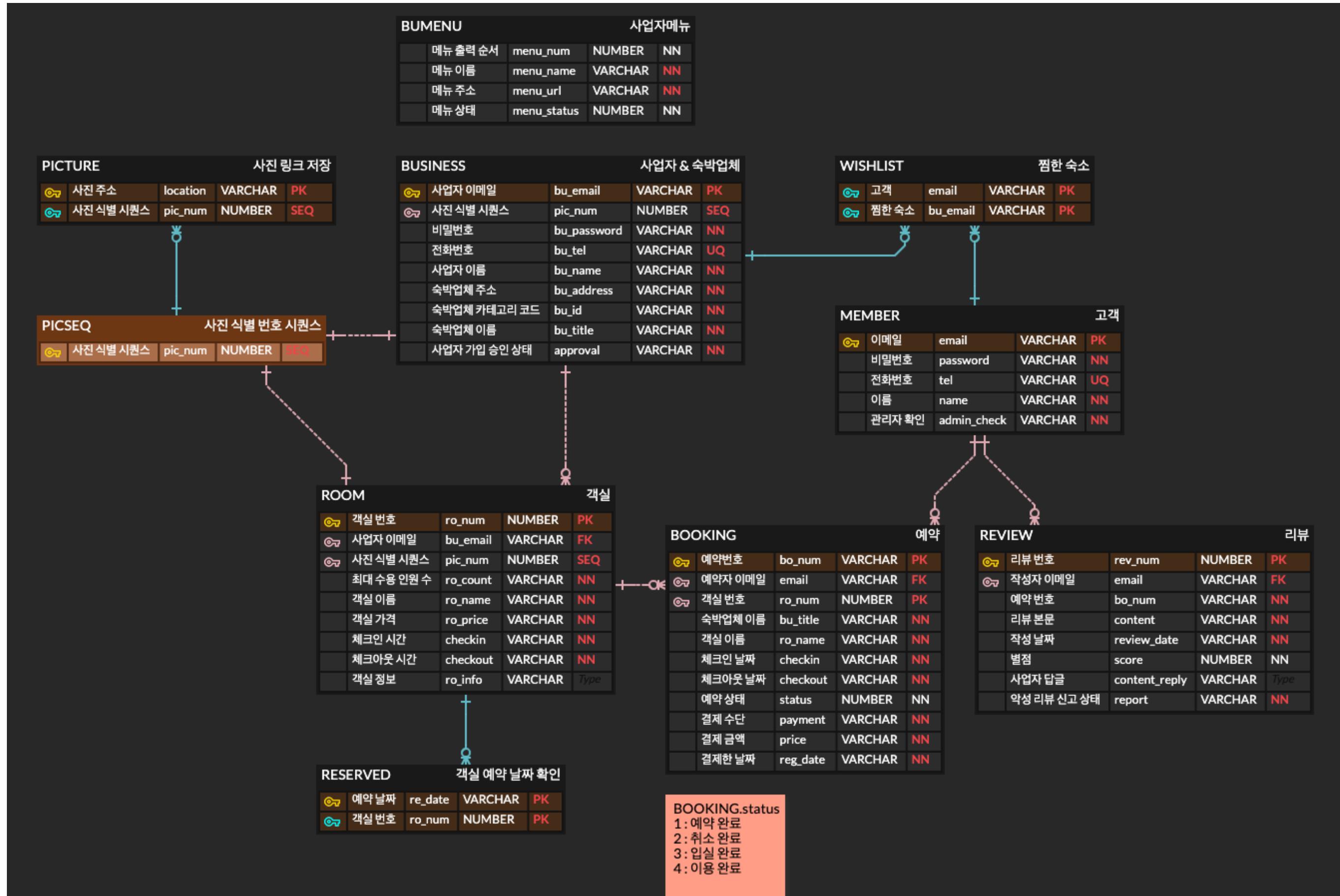
고객 / 사업자로 나뉘는 플랫폼 형태의 서비스를 흉내내기 위해
부족하지만 많은 경우의 수에 대응하며 제작했습니다.

06.

설계

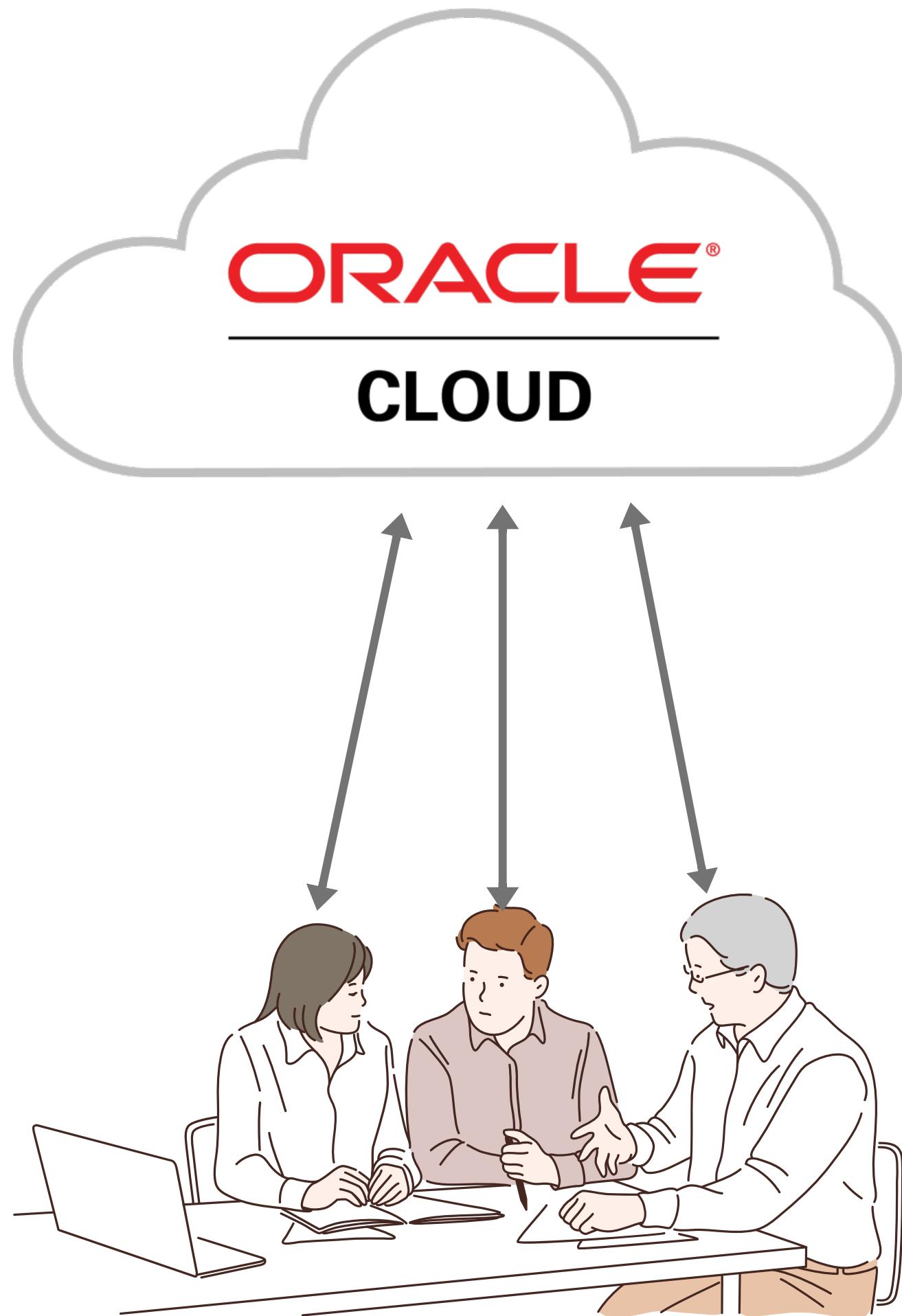


데 이 터 베 이 스 E R D



07.

핵심 기능



- 모든 조원이 항상 동일한 데이터베이스 사용
- 수정 즉시 다른 조원에게 적용되어 일관적인 데이터 저장
- 개인 개발 환경에 구애받지 않음



AOP를 이용한 Logging

로그

서비스의 메소드를 편하게 추적할 수 있도록 AOP를 이용해 서비스 패키지에 포인트 컷을 지정, 실행메소드와 소요 시간을 로그로 띄울 수 있도록 했습니다

```
[MemberService.getMemberOne(..)] : 0.496초
[SearchService.hot10BusinessList()] : 0.64초
[ReserveService.getReviewAvgCountBusiness(..)] : 0.548초
[ReserveService.getPicList(..)] : 0.534초
[ReserveService.getOverlapRoomList(..)] : 0.497초
[ReviewService.businessReviewList(..)] : 0.518초
[SearchService.hot10BusinessList()] : 0.65초
[ReserveService.getReviewAvgCountBusiness(..)] : 0.498초
[ReserveService.getPicList(..)] : 0.453초
[ReserveService.getOverlapRoomList(..)] : 0.525초
```

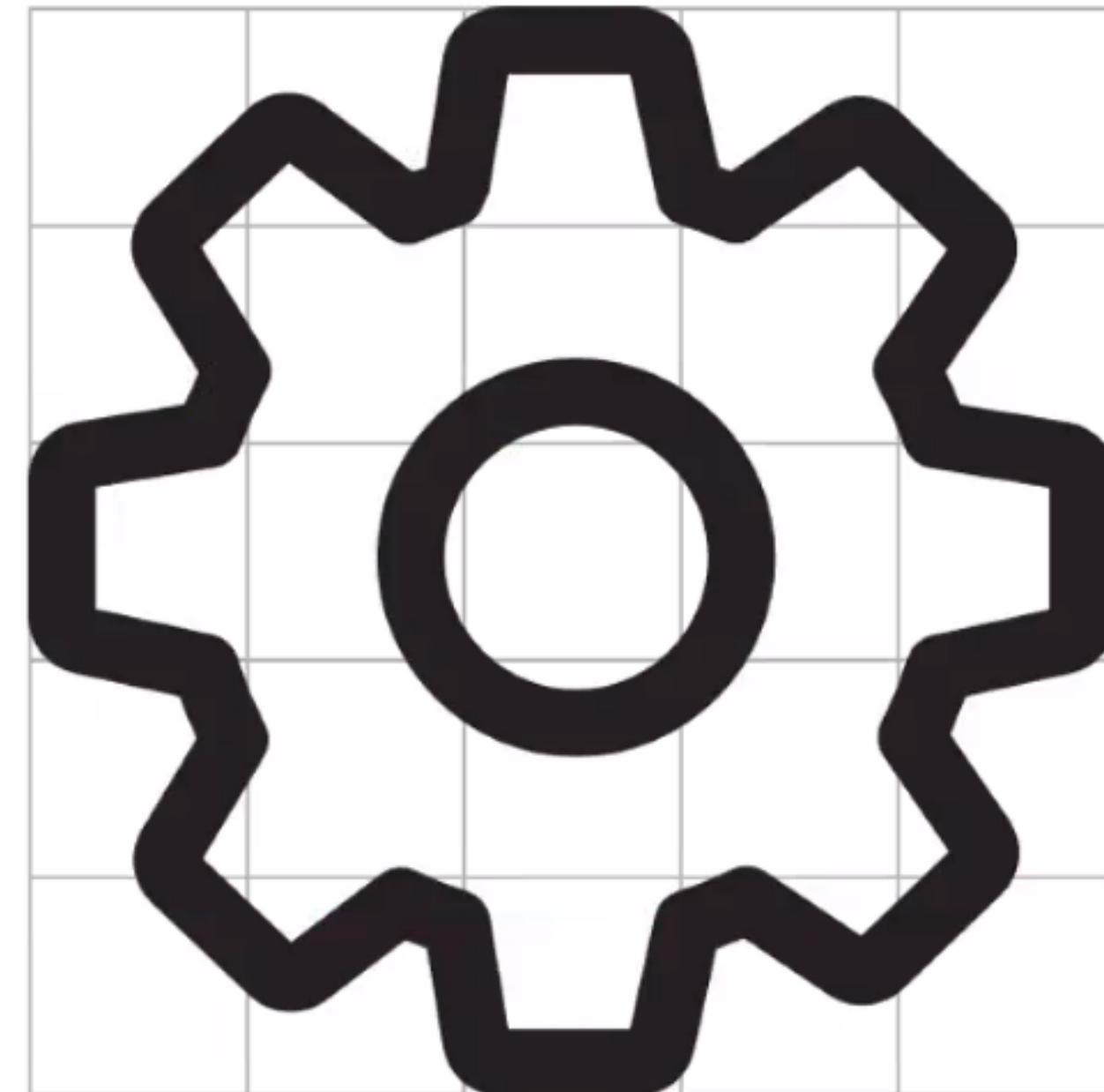
```
@Component
@Aspect
public class Logging {
    @Around("execution(* service.*.*(..))")
    public Object logger(ProceedingJoinPoint pjp) throws Throwable {
        Stopwatch sw = new Stopwatch();
        Object obj = null;
        sw.start();
```

> Refactoring

리팩토링

부족한 첫 설계에 기능들을 쌓아가다 보니 코드 품질이 계속해서 나빠지는 것을 느끼고 지속해서 리팩토링을 했습니다.

최대한 중복되는 코드를 줄이고 **SOLID**와 스프링의 사용 이유에 대해 공부하고 집중할 수 있었습니다.



자주 사용하는 코드 분리

DATAPARSE

```
1 package util;  
2  
3 import java.text.ParseException;  
4  
5  
6 public class DateParse {  
7     private static SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");  
8     private static SimpleDateFormat strFormat = new SimpleDateFormat("yyyyMMdd");  
9     private static DateTimeFormatter localDateFormat = DateTimeFormatter.ofPattern("yyyyMMdd");  
10  
11     // 외부에서 객체 생성 금지  
12     private DateParse() {}  
13  
14     // yyyy-MM-dd -> yyyyMMdd => DB에 넣기 위해 사용  
15     public static String dateToStr(String date) {  
16         return date.replaceAll("-", "");  
17     }  
18  
19     // yyyyMMdd -> yyyy-MM-dd => input date value로 넣기 위해 사용  
20     public static String strToDate(String str) {  
21         return strFormat.format(LocalDateTime.parse(str, localDateFormat));  
22     }  
23  
24 }
```



분리

예약 서비스 특성상 Date Type 데이터를 가공해서 사용할 일이 많아 중복되는 코드를 분리했습니다.



객체 전역화

거의 모든 곳에서 사용되는 객체이고 클래스 내부에 의존성이 없기 때문에 static 키워드를 이용해 더 쉽게 접근할 수 있도록 했습니다.



기본 생성자 private

불필요한 객체 생성을 막기 위해 기본 생성자를 private으로 선언했습니다.

SqlSessionFactory -> SqlSessionTemplate

SPRING JDBC

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="configLocation" value="classpath:mybatis/mybatis-config.xml"/>
    <property name="dataSource" ref="dataSource"/>
</bean>

<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
    <constructor-arg ref="sqlSessionFactory"/>
</bean>

<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver"/>
    <property name="url" value="jdbc:oracle:thin:@going_medium?TNS_ADMIN=/Users/lasbe/Wallet_going"/>
    <property name="username" value="_____"/>
    <property name="password" value="_____"/>
</bean>
```



스프링 트랜잭션

스프링 트랜잭션 설정에 따라 자동으로
커밋과 롤백을 수행합니다.
덕분에 세션을 열고 닫는 코드를 줄일 수
있었습니다.



Thread-Safe

SqlSessionTemplate의 빈은
Thread-Safe하게 세션 객체를 제공하기
때문에, 동일한 객체를 공유할 수 있습니다.

Field Injection -> Constructor Injection

DI

```
private final ReserveDao reserveDao;
private final BookingDao bookingDao;

@Autowired
public ReserveService(ReserveDao reserveDao, BookingDao bookingDao) {
    this.reserveDao = reserveDao;
    this.bookingDao = bookingDao;
}
```



Final 키워드 사용

의존성 주입 중 생성자 주입을 사용했을 때만
final 키워드를 사용할 수 있습니다.

이는 불필요한 변경의 가능성을 차단할 수 있고
실행 중 의존성에 대한 문제를 컴파일 시점
에 미리 방지할 수 있다는 장점이 있습니다



StackOverflow 방지

실제로 가능성은 낮지만 StackOverflow가
발생할 시 프로그램이 구동 되는 도중 에러가
발생하는 것이 아닌 객체의 생성 시점에 오류를
발견할 수 있습니다.

비즈니스 로직 분리

@SERVICE

```
@Service  
public class SearchService {  
  
    private final SearchDao searchDao;  
  
    @Autowired  
    public SearchService(SearchDao searchDao) {  
        this.searchDao=searchDao;  
    }  
  
    public List<Business> searchBusinessList(SearchDTO searchDTO) throws Exception {  
        return searchDao.searchBusinessList(searchDTO);  
    }  
  
    public List<Business> hot10BusinessList() throws Exception {  
        return searchDao.hot10BusinessList();  
    }  
}
```



Final 키워드 사용

Controller는 요청과 응답에 대한 코드에만 집중,
Service는 비즈니스 로직에 집중,
Repository는 DB에 접근하는 코드를 집중시켜
집중된 책임을 분산시킬 수 있었습니다.



여러개의 DAO를 묶은 트랜잭션

하나의 로직을 구동하기 위해 서비스에서
여러 DAO를 불러와 DB의 상태를 바꾸기 때문에
하나의 논리적 기능으로써 이용할 수 있습니다.

```

<mvc:interceptors>
    <mvc:interceptor>
        <mvc:mapping path="/reservation/reserve"/>
        <mvc:mapping path="/reservation/reservePro"/>
        <mvc:mapping path="/reservation/cancel"/>
        <mvc:mapping path="/reservation/review"/>
        <mvc:mapping path="/reservation/reviewPro"/>
        <bean class="interceptor.MemberLoginInterceptor" />
    </mvc:interceptor>
    <mvc:interceptor>
        <mvc:mapping path="/room/**"/>
        <bean class="interceptor.BusinessLoginInterceptor"/>
    </mvc:interceptor>
    <mvc:interceptor>
        <mvc:mapping path="/admin/monthlySales"/>
        <mvc:mapping path="/admin/areaSales"/>
        <mvc:mapping path="/admin/categorySales"/>
        <mvc:mapping path="/admin/businessApproval"/>
        <mvc:mapping path="/admin/reviewReport"/>
        <bean class="interceptor.AdminLoginInterceptor" />
    </mvc:interceptor>
</mvc:interceptors>

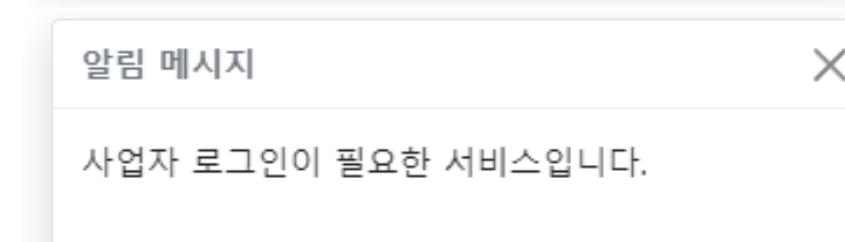
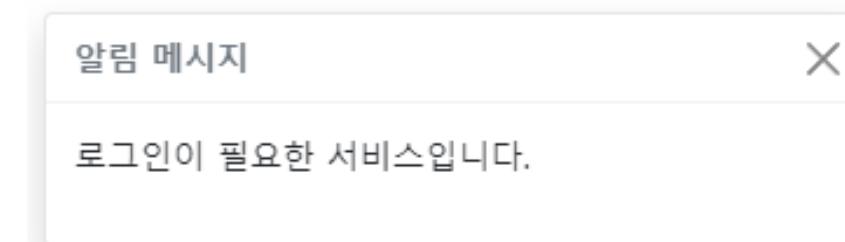
```

```

@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object h
    throws Exception {
    HttpSession session = request.getSession();
    String email = (String) session.getAttribute("admin");
    if(email == null) {
        String msg = URLEncoder.encode("관리자 로그인이 필요한 서비스입니다.", "UTF-8");
        response.sendRedirect(request.getContextPath() + "/admin/adminLogin?msg=" + msg);
        return false;
    }
    return true;
}

if(email == null) {
    String msg = URLEncoder.encode("로그인이 필요한 서비스입니다.", "UTF-8");
    response.sendRedirect(request.getContextPath() + "/member/loginForm?msg=" + msg);
    return false;
}
return true;
}

```



인터셉트를 이용해 로그인 하기전에
관리자, 고객, 사업자페이지로 이동하면
해당 페이지에 필요한 아이디로 로그인하라는
메세지를 출력하고 해당 로그인페이지로 이동합니다.

사업자 객실관리

객실 등록 및 관리

- 사업자가 객실을 등록하면 가격순으로 객실 확인
- 객실에 들어가면 해당 객실의 정보를 출력하며 정보수정과 객실삭제 가능



고잉트윈 (바다전망)

이용인원 : 1명

30,000원



고잉더블 (바다전망)

이용인원 : 3명

50,000원



고잉트원 (바다전망)

객실기본정보

가격	30000원
체크인	15:00
체크아웃	00:00

주변정보

승우역 차량 20분
을왕리해수욕장 차량 5분
한국도심공항 터미널 차량 18분

공지사항

정보수정

객실삭제

지도검색

지도를 이용한 검색

검색 결과를 지도 API에 표시해
숙소 위치를 직관적으로 확인할 수
있습니다. 또한 찾은 숙소로 바로
이동할 수 있습니다

GOING

지도검색 예약내역 로그인

상세조건

2022. 05. 12.

2022. 05. 13.

인원 수

2

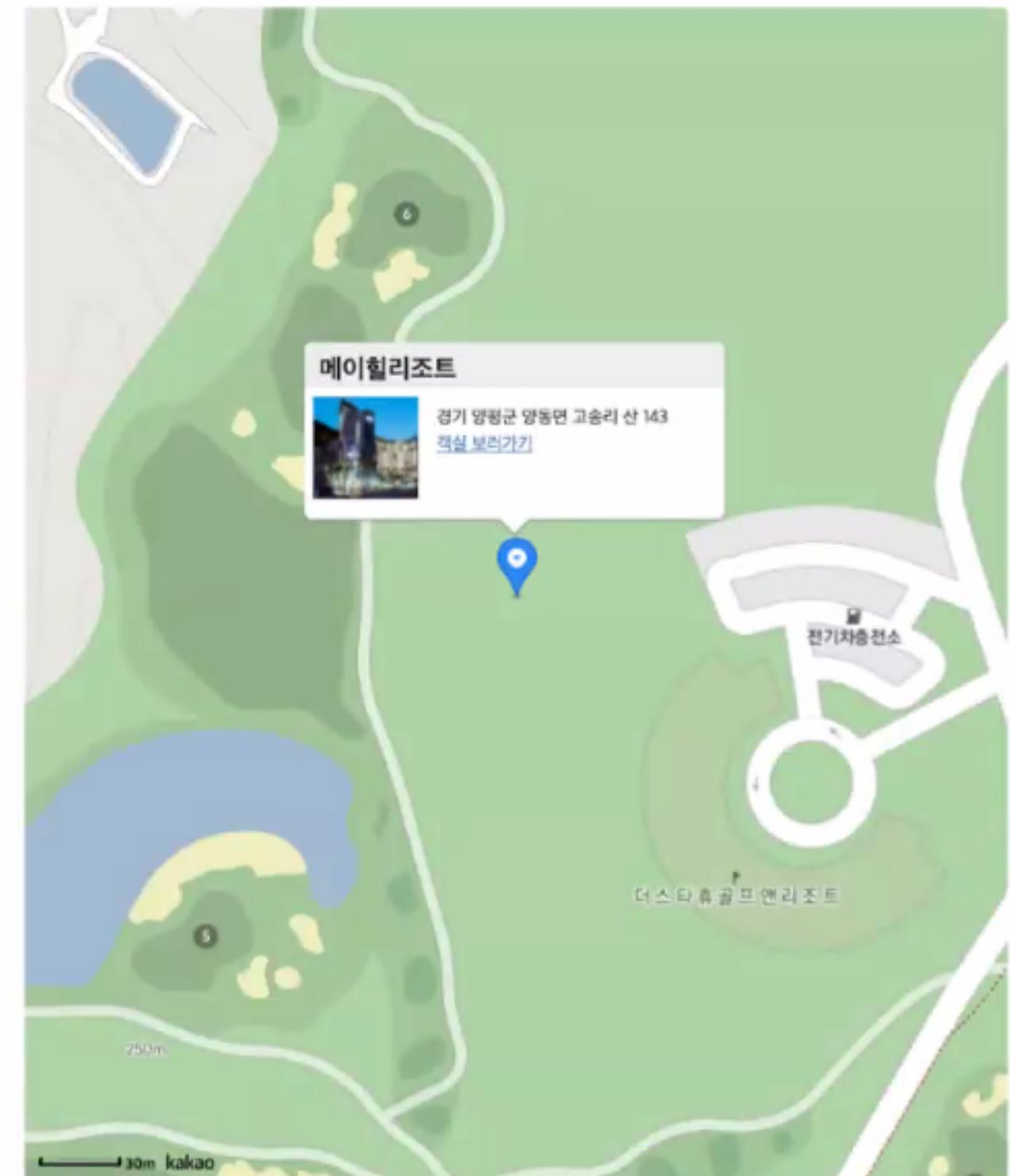
검색

메이힐

숙소 유형

- 호텔
- 모텔
- 펜션
- 리조트

적용



사업자 예약 관리

예약 내역

방이름

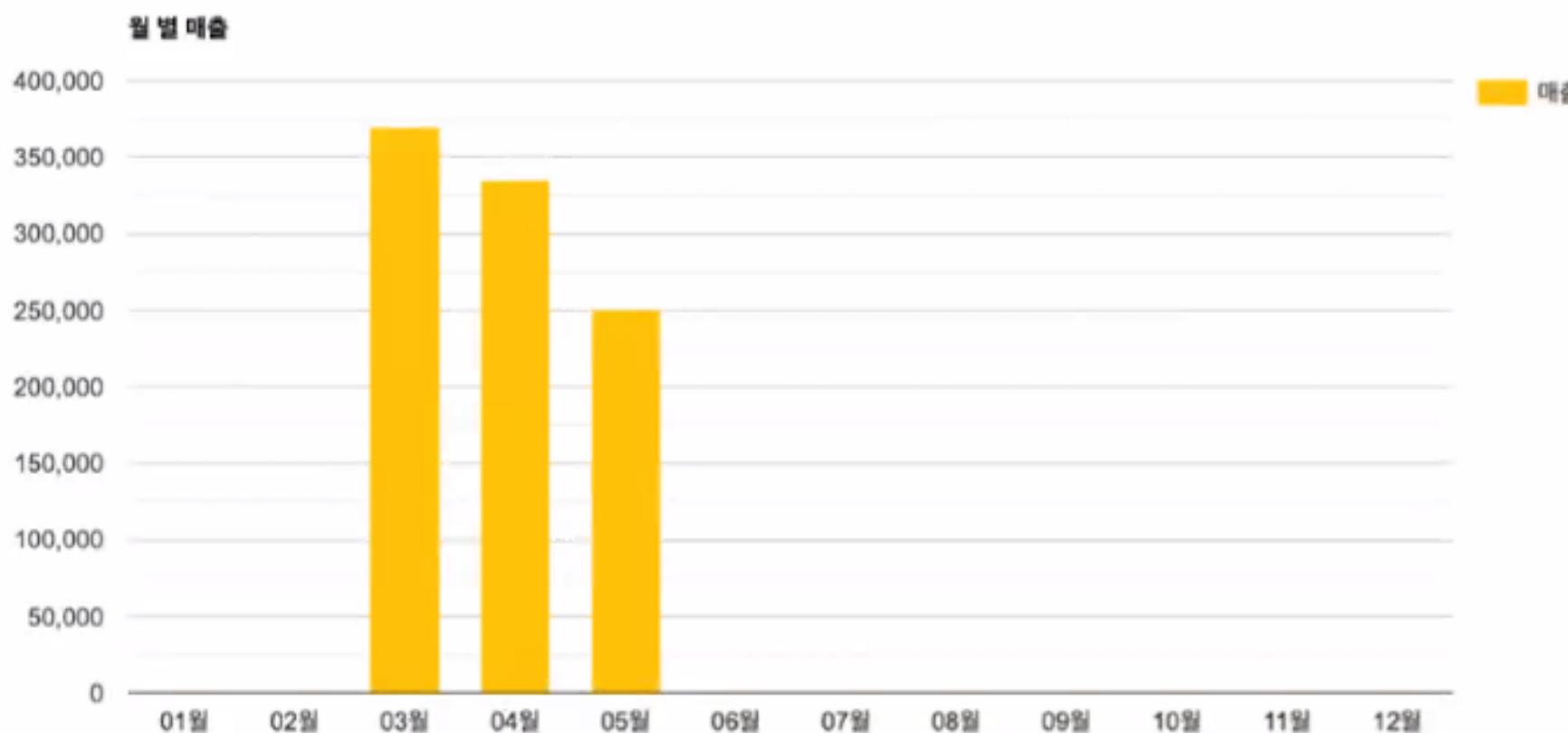
검색

고객이 예약을 완료했다면 해당 숙박업소는 모든 객실 예약에 관련된 정보를 확인 및 검색할 수 있습니다.

객실 이름	이용인원수	체크인	체크아웃	예약자이름	핸드폰번호	Email	예약상태
고잉 복층	2	20220509	20220510	조성필	01051324874	poi@naver.com	예약 완료
고잉트윈 (바다전망)	1	20220325	20220328	장성우	01027194917	2166757786	결제 취소
고잉 패밀리 스몰 (바다전망)	3	20220326	20220409	장성우	01027194917	2166757786	결제 취소
고잉 패밀리 라지 (바다전망)	4	20220503	20220504	장성우	01027194917	2166757786	결제 취소
고잉 복층	2	20220326	20220327	HYG	01077777777	2168574158	이용 완료
고잉 패밀리 라지 (바다전망)	4	20220330	20220401	조성필	01051324874	poi@naver.com	이용 완료
고잉더블 (바다전망)	3	20220331	20220401	장성우	01027194917	2166757786	이용 완료
고잉 복층	2	20220331	20220401	신발가게	01099999999	abc@naver.com	이용 완료
고잉 복층	2	20220401	20220402	조성필	01051324874	poi@naver.com	이용 완료
고잉더블 (바다전망)	3	20220406	20220409	장성우	01027194917	2166757786	이용 완료

Previous 1 2 Next

객실등록 객실정보 예약확인 매출 체크인 체크아웃 리뷰



사업자 매출 통계

GOOGLE CHART



예약을 통한 매출의 발생을
월별 통계로 확인할 수 있습니다.

입-퇴실 관리

객실등록 객실정보 예약확인 매출 체크인 체크아웃 리뷰

오늘 체크인목록

입실 전							입실완료								
번호	객실 이름	이용인원수	체크인	체크아웃	예약자이름	핸드폰번호	예약상태	번호	객실 이름	이용인원수	체크인	체크아웃	예약자이름	핸드폰번호	예약상태
1	고양 복층	2	20220509	20220510	조성필	01051324874	체크인								

사업자는 당일 체크인 고객이 입실했을 시
입실 전 -> 입실 완료로 상태를 변경 가능하고
입실한 고객이 퇴실했을 시
퇴실 전 -> 퇴실 완료로 상태를 변경해서
고객관리를 할 수 있습니다.

오늘 체크아웃 목록

퇴실 전							퇴실완료							
객실 이름	이용인원수	체크인	체크아웃	예약자이름	핸드폰번호	예약상태	번호	객실 이름	이용인원수	체크인	체크아웃	예약자이름	핸드폰번호	예약상태
고양 복층	2	20220509	20220510	조성필	01051324874	체크아웃								

체크아웃 날짜보다 일찍 퇴실하는 고객을 퇴실 완료로 변경하면 **퇴실 날짜 ~ 체크아웃 날짜** 기간은 다시 예약이 가능하도록 했습니다.

사업자 리뷰 관리

사업자가 운영하는 숙소의 리뷰는 한꺼번에 확인할 수 있습니다.
악의적으로 비방하는 댓글은 서비스 운영자에게 신고할 수 있고,
손님의 댓글에 답글을 달수도 있습니다.

The screenshot shows a web application for managing reviews. At the top, there's a navigation bar with icons for search, filter, and user profile. Below it is a table of reviews:

별점	리뷰 내용	작성일
★★★☆☆	2166757786 - 고양이호 (바다전망) 항상 여기만 옵니다	2022-06-11
★★★★★	2166757786 - 고양 스페셜 리뷰 온 대고 예약했는데 직원들이 불친절하네요는 다신 안합니다	2022-06-11
★★★★★	2166757786 - 고양 바다 너무 잘 놀다 갑니다	2022-06-11

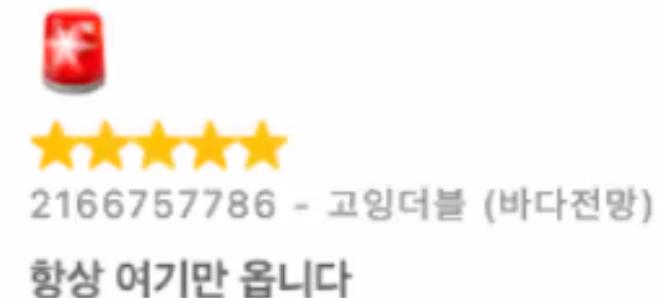
In the center, a modal dialog is open with the following content:

Review?rev_num=48&content_reply=오지마세요

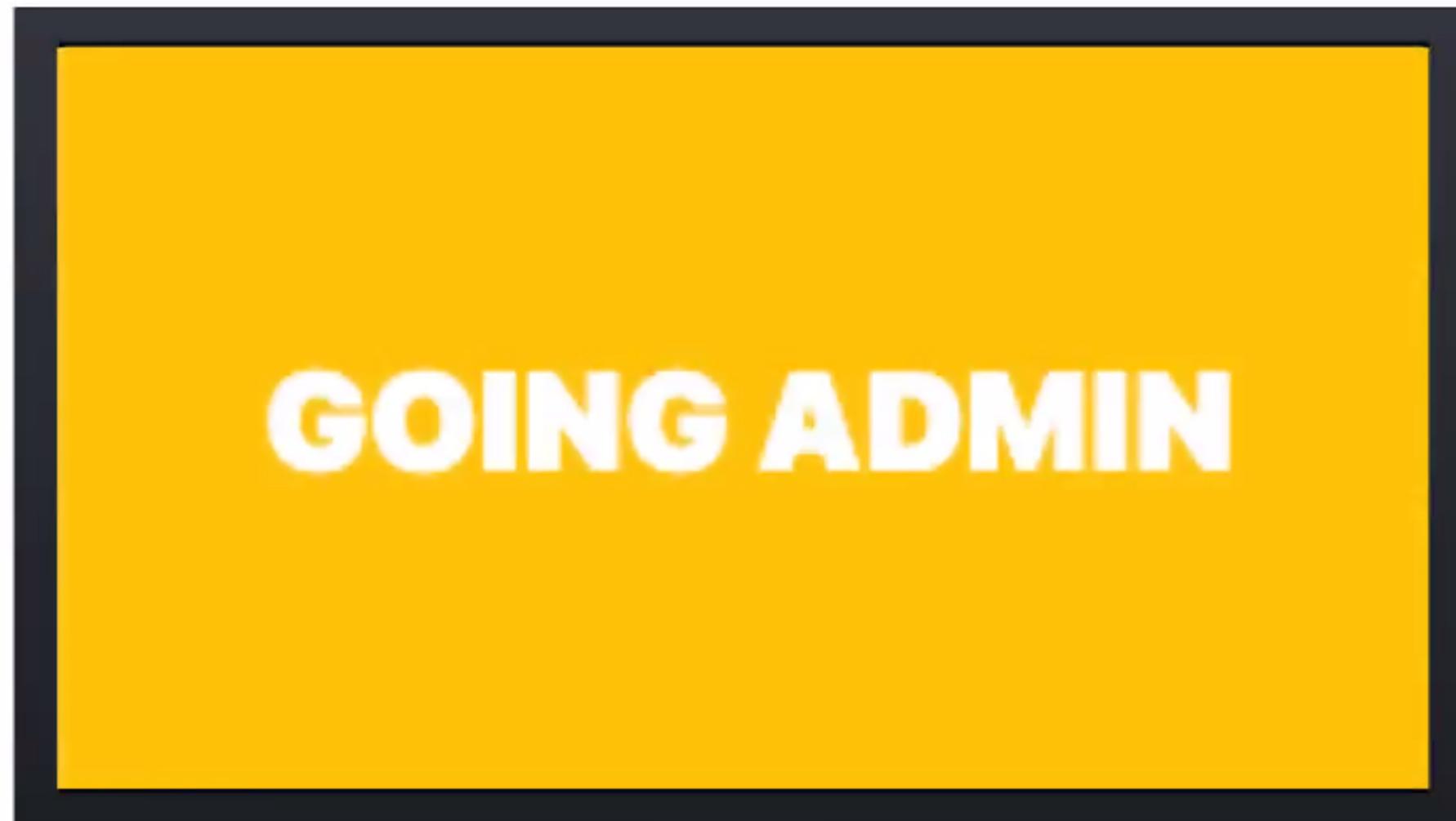
localhost:8080의 메시지

2166757786님의 리뷰를 신고하시겠습니까??

취소 **확인**



↳ 감사합니다



관리자 페이지

ADMIN 기능

- 로그인
- 월별 매출
- 지역별 매출
- 숙소 카테고리별 매출
- 사업자 가입 승인
- 리뷰 신고 관리

관리자 로그인

localhost:8080/goingSpring/admin/adminLogin

이메일

비밀번호

로그인

관리자 로그인창은 서비스에 직접적으로 노출되어 있지 않아 링크를 직접 입력해 이동해야 합니다.

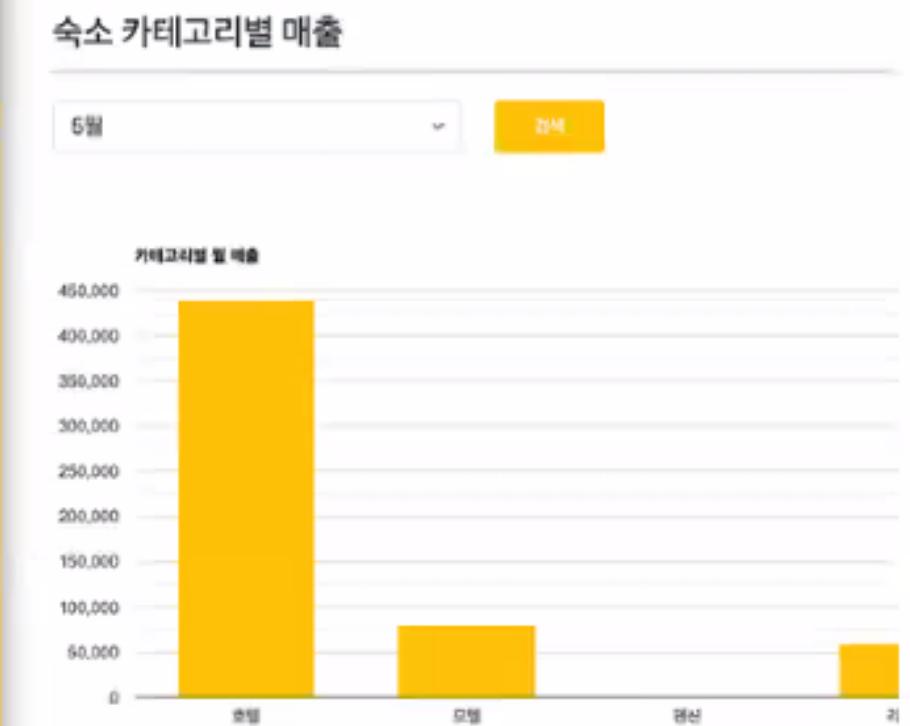
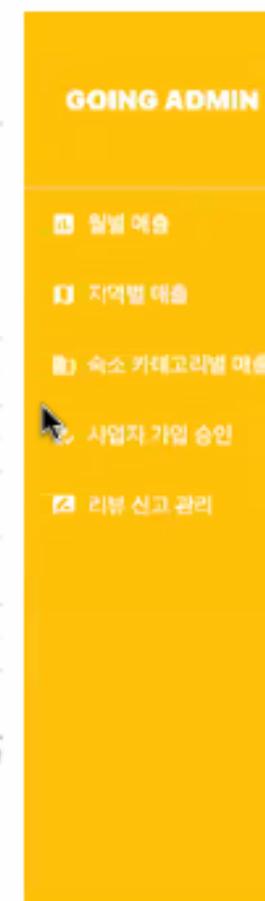
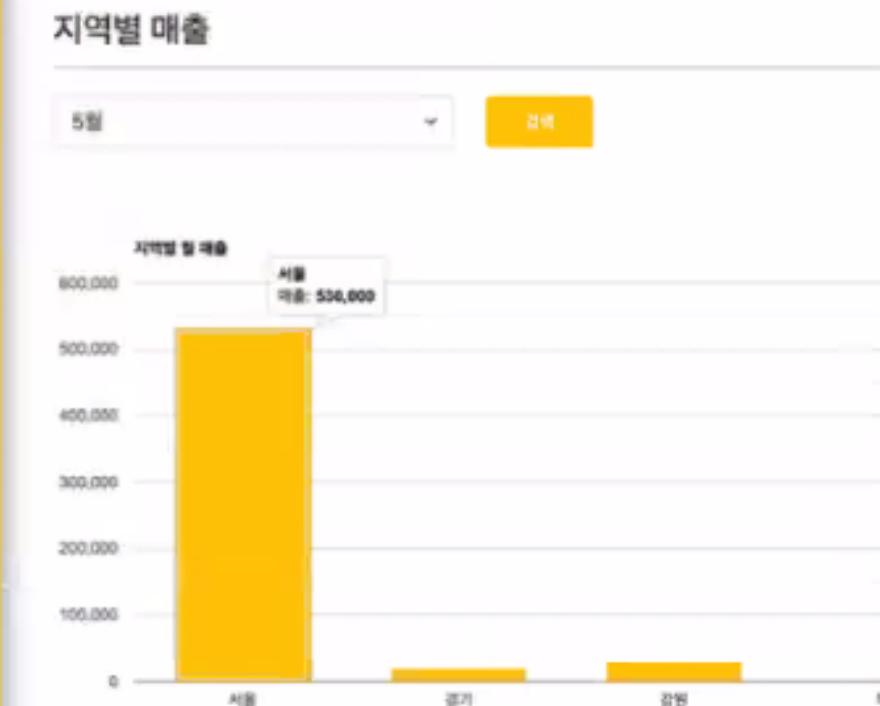
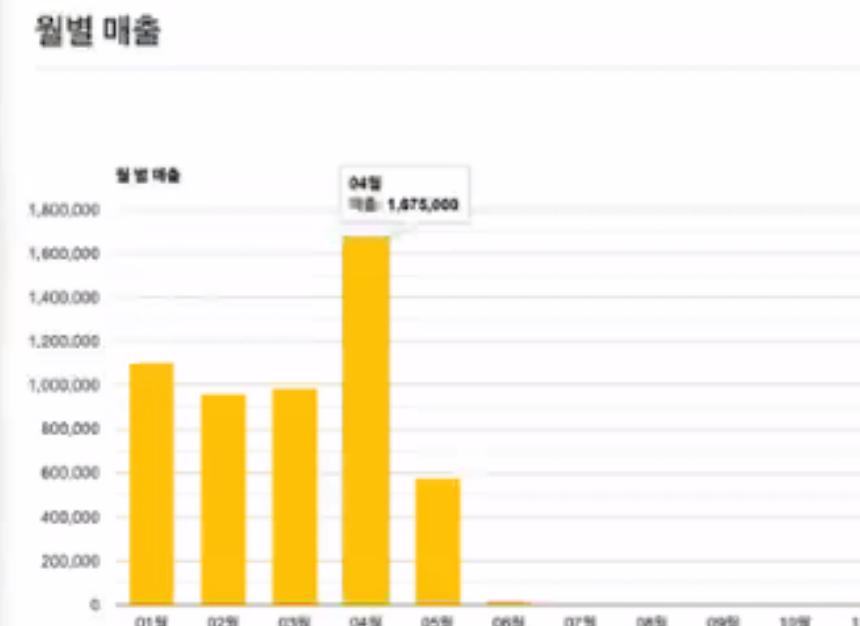
관리자 검증은 MEMBER 테이블의 admin_check 를 통해 이루어지며 이 데이터는 서비스 내에서 수정이 불가능하기 때문에 DBMS로 직접 해주어야 합니다.

MEMBER					고객
키	이메일	e-mail	VARCHAR	PK	
	비밀번호	password	VARCHAR	NN	
	전화번호	tel	VARCHAR	NN	
	이름	name	VARCHAR	NN	
	관리자 확인	admin_check	VARCHAR	NN	

다양한 매출 통계

월별 총합, 지역별, 카테고리별

관리자는 다양한 통계를 통해 서비스에 대한 전반적인 매출을 확인할 수 있습니다.



프로젝트 후기

팀 프로젝트는 처음이기도 하고 가이드라인이 없는 상태여서 도대체 어디서부터 시작해야될지 막막함을 느꼈습니다.

어떻게 역할 배분을 해야하는지, 어떻게 테이블을 구성하고 기능들을 연결해야할지 도저히 감도 잡히지 않았지만 팀원들과 같이 프로젝트에 필요한 기능들을 하나하나 Whinsical에 나열해가며 해당 기능에 필요한 테이블과 로직들을 논의하며 작업하는데 처음에 막막했지만 하나씩 완성되가는걸 보며 협업의 중요성에 대해 알게 되고 전공자인 팀장덕분에 설계에대해 많은걸 공부하고 대략적인 페이지 흐름도와 데이터베이스 관계도를 작성해나가며 추상적이었던 로직들이 점차 구체화 됨을 느끼며 설계 단계의 중요성을 느낄 수 있었습니다.

막막했던 초기부터 완성까지의 짧은 시간의 기억들을 되짚어보면 처음에는 수업에 설계에대한 내용이 없어 설계부터 로직짜는데 어떤식으로 해야될지 몰라 막막했지만 팀원들과 하나씩 해결해가면서 완성되는걸 보며 즐거움을 느끼고 스프링프로젝트로 넘어가면서 5명이었던 팀이 팀장과 저만 남아서 프로젝트구현하는데 힘들긴 했지만 결과에서 뿌듯함을 얻어갈 수 있는 너무나 소중한 기회였습니다.

**THANK
YOU**