

서울메이트

자바 기반 웹 서비스 개발 프로젝트

seoulmate팀 한민주, 김정훈, 정주은, 송원기, 송하은



목차

01	프로젝트 개요	프로젝트명과 개발 목적, 대상 소개
02	팀 소개	seoulmate팀 소개
03	개발 및 배포 환경	개발 및 배포에 사용된 프로그램 소개
04	WBS	프로젝트 진행 업무 분류 체계
05	프로젝트 구조	프로젝트 구조
06	사이트 구성	시스템 구조, 사이트 맵 구조 및 상세, 프로세스 정의
07	ERD	데이터 테이블 생성
08	REST API	프로젝트 진행 시 필요한 API 생성
09	핵심 기능	개발 과정에서 중요했던 핵심 기능 설명
10	배포	배포 과정 설명
11	트러블 슈팅	개발 과정에서 발생된 오류 문제 해결 과정
12	추후 작업	프로젝트 이후 추가 작업 설명
13	소감	프로젝트 종료 후 각자 소감

01

프로젝트 개요

프로젝트명과 개발 목적, 대상 소개



1. 프로젝트 개요

- 프로젝트 명

우리나라 민족의 넘치는 흥을 내포하고
'서울'의 발음과도 유사한 '소울'을 먼저 지정
'소울'과 '서울'을 모두 뜻하면서 사이트의 목적이 드러나는
프로젝트 명을 논의한 끝에,
함께하면 더 좋다! 라는 의미의 메이트가 들어가서 '서울메이트'로 결정

- 개발 목적

서울의 인기있는 명소들을 관광하고자 하는 사람들을 위해 가장 대표적인
명소를 안내하고, 투어버스 예약으로 관광객들의 편의성을 더함

- 대상

서울 관광이 처음이거나 대한민국 여행을 목적으로 온 외국인부터
투어버스 이용을 위해 예약하고자 하는 모든 관광객

- 주요 기능

서울 주요 도시의 대표적인 명소들 소개하고 안내,
예약과 문의 페이지를 통해 원활한 투어버스 이용



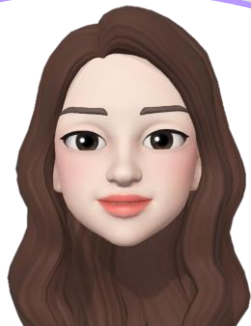
02

팀 소개

팀 소개



2. 팀 소개



한민주 : 팀장, FE, BE 담당

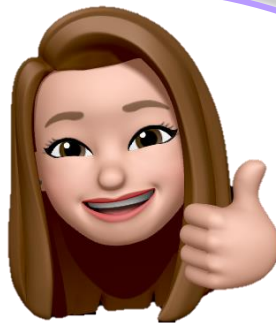
- 화면설계서 및 wbs 작성
- 발표자료 작성 (핵심기능, 팀 소개) 및 최종 수정

FE

- addForm 예약 페이지 html, css 작성
- 예약 내역에서 문의하기, 예약취소 수정
- 문의 내역 문의글 수정

BE

- controller 수정 및 예약내역, 문의내역 검색 기능 추가
- 문의 내역 문의글 내용 보기 기능 추가
- 전체 테스트
- DB, Rest API 설계



정주은 : 발표, FE, BE 담당

- 화면설계서 및 발표자료 작성 (트러블슈팅)

FE

- 로고 & 파비콘 제작
- index, 예약내역, 문의내역 페이지 html, css 작성
- 투어버스 예약, 문의하기 페이지 datepicker와 bootstrap 적용하여 페이지 업데이트

BE

- jdbcTemplate 이용하여 repository 작성
- controller 최종 수정 (예약하기, 예약 등록, 예약내역에서 문의하기 이동, 예약 취소)
- 예약 취소, 예약 인원수 계산 test 진행
- DB, Rest API 설계

2. 팀 소개



송원기 : BE 담당

- 화면설계서 및 발표자료 작성 (핵심기능)

BE

- repository를 이용하여 service 작성
- repository에 예약 취소 기능 추가
- repository, service 초기 테스트 (예약 저장 및 내역, 문의 저장 및 내역)
- DB, Rest API 설계



송하은 : FE 담당

- 화면설계서 및 발표자료 작성 (목차, 프로젝트 개요, 개발환경, WBS)

FE

- index 페이지 배치 정렬, 내비게이션바 작업 및 디자인 수정
- addForm 문의 페이지 html, css 작성

BE

- domain 작성



김정훈 : FE, BE 담당

- 화면설계서 및 발표자료 작성 (사이트 구성, ERD, REST API)

FE

- 명소 상세 페이지 html, css 작성

BE

- 초기 Controller 작성
- 예약 취소 기능 수정
- Rest API 설계

03

개발 및 배포 환경

개발 및 배포에 사용된 프로그램 소개



3. 개발 및 배포 환경

개발환경

- Windows 10
- Spring boot 2.7.9
- Java 11.0.16.1
- Gradle 7.6.1
- Lombok 1.18.26
- Thymeleaf 3.0.14
- Eclipse IDE
- H2 2.1.214
- Bootstrap 5.3
- jQuery 1.7.1

배포환경

- ubuntu
- AWS EC2
- AWS RDS
- MySQL



04

WBS

프로젝트 진행 업무 분류 체계



4. WBS

Project	서울관광지예약	한민주
Manager	한민주	김정훈
Write	한민주	송현기
Version	1.0	송하은
Issued date	2023-03-29	정주은

Work Break-down Structure						3월							4월							4월							4월							
Procedures	Steps	Tasks	담당자	schedule		산출물/비고	4주							1주							2주							3주						
				시작일	종료일		월	화	수	목	금	토	일	월	화	수	목	금	토	일	월	화	수	목	금	토	일	월	화	수	목	금	토	일
1.0.0. 착수 및 프로젝트 관리							27	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1.1.0. 화면설계서 작성																																		
		1.1.1. 개정이력, 서비스개요, 요구사항정의, 시스템구조 작성	현민주	3월 29일	3월 29일	화면설계서																												
		1.1.2. 와이어프레임, 페르소나 작성	송하은	3월 29일	3월 29일	화면설계서																												
		1.1.3. 와이어프레임, 페르소나, 디자인시스템 작성	정주은	3월 29일	3월 29일	화면설계서																												
		1.1.3. 사이트맵구조 및 상세, 프로세스 작성	송원기	3월 29일	3월 29일	화면설계서																												
1.2.0 화면설계서 수정																																		
		1.2.1. 사이트맵구조, 프로세스 정의 수정, 화면설계 추가	송원기	3월 30일	3월 30일	화면설계서																												
		1.2.2. 사이트맵상세 수정, 프로토타입 제작 구분 추가	김정훈	3월 30일	3월 30일	화면설계서																												
		1.2.3. 화면설계 추가	송하은	3월 30일	3월 30일	화면설계서																												
		1.2.4. 화면설계 추가	정주은	3월 30일	3월 30일	화면설계서																												
		1.2.5. 화면설계서 수정, 화면설계 추가	현민주	3월 30일	3월 30일	화면설계서																												
		1.2.5. 프로토타입 추가	송하은	3월 31일	3월 31일	화면설계서																												
		1.2.5. 프로토타입 추가, REST API 추가	정주은	3월 31일	3월 31일	화면설계서																												
		1.2.5. 화면설계 수정	김정훈	3월 31일	3월 31일	화면설계서																												
		1.2.5. 화면설계 수정	송원기	3월 31일	3월 31일	화면설계서																												
		1.2.5. 화면설계서 수정	현민주	3월 31일	3월 31일	화면설계서																												
2.0.0. 프론트 개발																																		
2.1.0. 관리자 페이지 프론트 개발, 일반관리																																		
		2.1.1. 홈	송하은	3월 31일	4월 3일	index.html																												
		2.1.2. 명소상세페이지	김정훈	3월 31일	4월 4일	gangnam.html, gwanghwamun.html, myeongdong.html, jamsil.html, hongdae.html																												
		2.1.3. 예약폼	현민주	3월 31일	4월 4일	addForm.html (book)																												
		2.1.3. 예약내역, 문의내역	정주은	3월 31일	4월 4일	list.html (book, inquiry)																												
		2.1.3. 문의폼	송원기	3월 31일	4월 4일	addForm.html (inquiry)																												
3.0.0. 백엔드 개발																																		
3.1.0. 개발 설계부분																																		
		3.1.1. domain	송하은	3월 31일	4월 3일	Seoul.java																												
		3.1.2. controller	김정훈	4월 3일	4월 4일	SeoulController.java																												
		3.1.3. controller	현민주	4월 3일	4월 4일	SeoulController.java																												
		3.1.4. repository	정주은	3월 31일	4월 3일	SeoulRepository.java																												
		3.1.5. service	송원기	4월 3일	4월 4일	SeoulService.java																												
4.0.0. 테스트 및 완료																																		
4.1.0. 유닛테스트																																		
		4.1.2. controller	김정훈	4월 3일	4월 4일	SeoulController.java																												
		4.1.3. controller	현민주	4월 3일	4월 4일	SeoulController.java																												
		4.1.4. repository	정주은	3월 31일	4월 3일	SeoulRepository.java																												
		4.1.5. service	송원기	4월 3일	4월 4일	SeoulService.java																												
		4.2.0. 전체테스트	현민주	4월 4일	4월 5일	SeoulService.java																												
		4.3.0. 배포	현민주	4월 10일	4월 23일	SeoulService.java																												

05

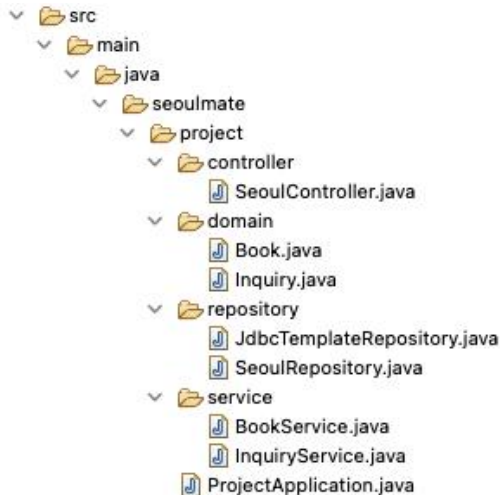
프로젝트 구조

프로젝트 구조



5. 프로젝트 구조(java)

- java



controller 프로젝트 전체에 사용되는 SeoulController를 포함

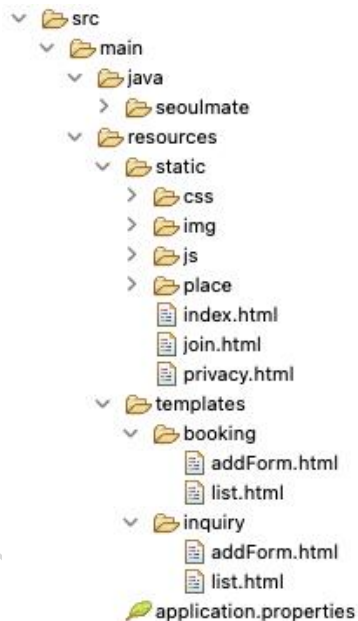
domain 예약, 문의에서 사용되는 클래스를 포함

repository 데이터베이스에 데이터를 전송해주는 JDBC repository와 인터페이스를 선언한 SeoulRepository를 포함

service 컨트롤러와 레퍼지토리를 연결해주는 BookService와 InquiryService를 포함

5. 프로젝트 구조(view)

- view



static

정적 자원인 CSS, 이미지와 index 페이지 등을 포함

templates

컨트롤러를 통해 전달되는 동적 자원을 포함

booking과 inquiry로 폴더를 나누어 관련된 페이지를 묶어 관리

5. 프로젝트 구조(test)

- test

```
src
├── main
└── test
    ├── java
    │   ├── seoulmate
    │   │   ├── project
    │   │   │   ├── repository
    │   │   │   │   ├── JdbcTemplateRepositoryTest.java
    │   │   │   └── service
    │   │   │       ├── BookServiceTest.java
    │   │   │       ├── InquiryServiceTest.java
    │   │   │       └── ProjectApplicationTests.java
```

repository jdbcTemplateRepository 테스트 파일을 포함

service BookService와 InquiryService 테스트 파일을 포함

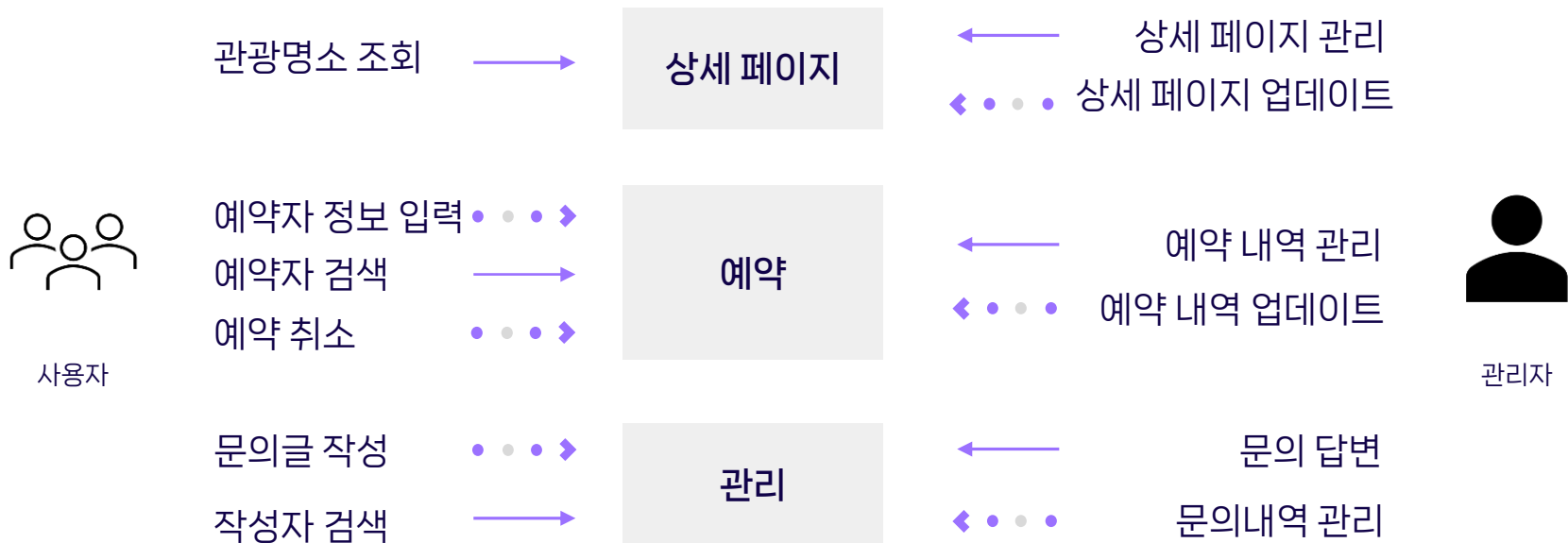
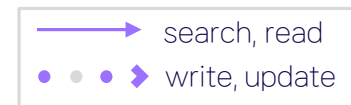
06

사이트 구성

시스템 구조, 사이트 맵 구조 및 상세, 프로세스 정의



6-1. 시스템 구조



6-2. 사이트 맵 구조

관광명소

강남

잠실

광화문

홍대

명동

예약

예약하기

예약내역

예약검색

예약취소

문의

문의하기

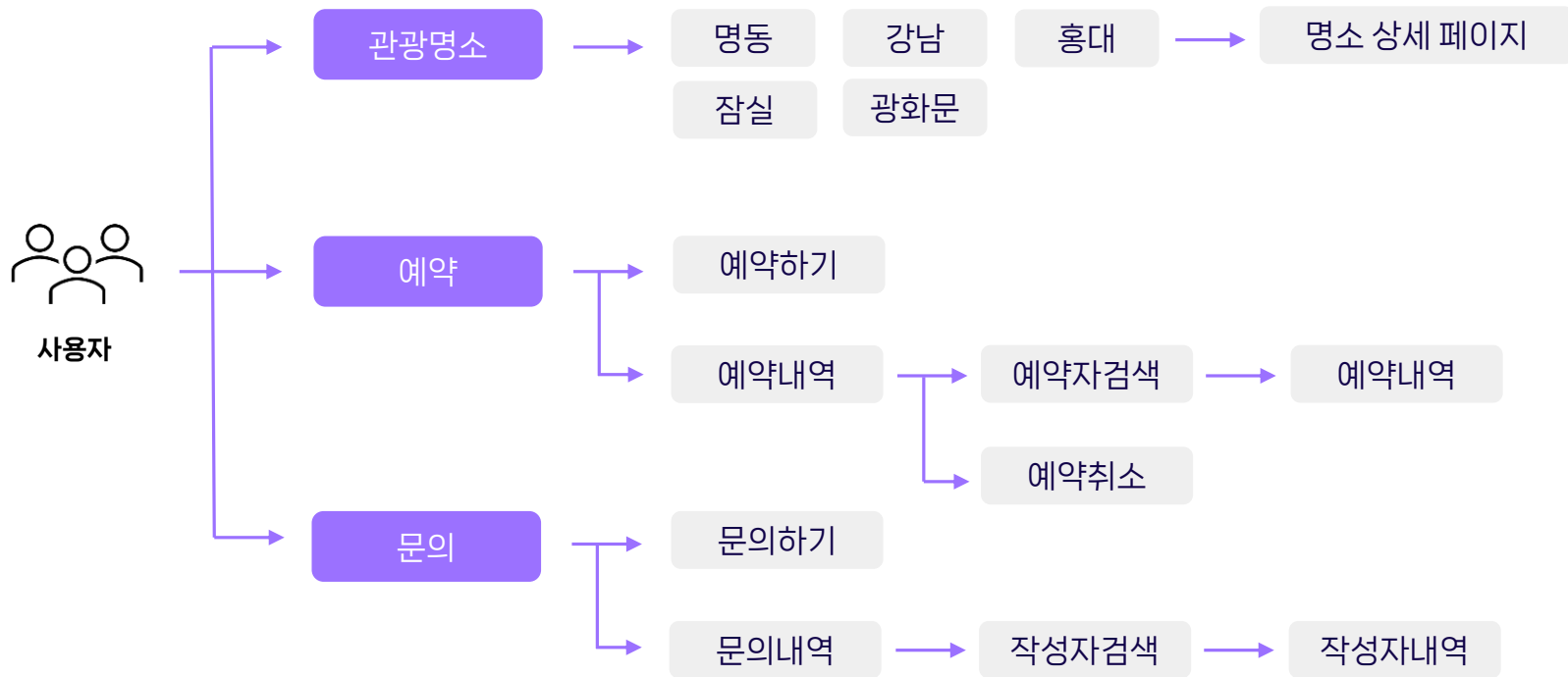
문의내역

문의검색

6-3. 사이트 맵 구조 상세

메뉴 1	메뉴 2		Descriptiton
관광명소	강남	잠실	명소 상세 페이지
	광화문	홍대	
	명동		
예약	예약하기		
	예약 내역		예약자 검색 예약자 내역 예약 취소
문의	문의하기		
	문의 내역		작성자 검색 작성자 내역

6-4. 프로세스 정의



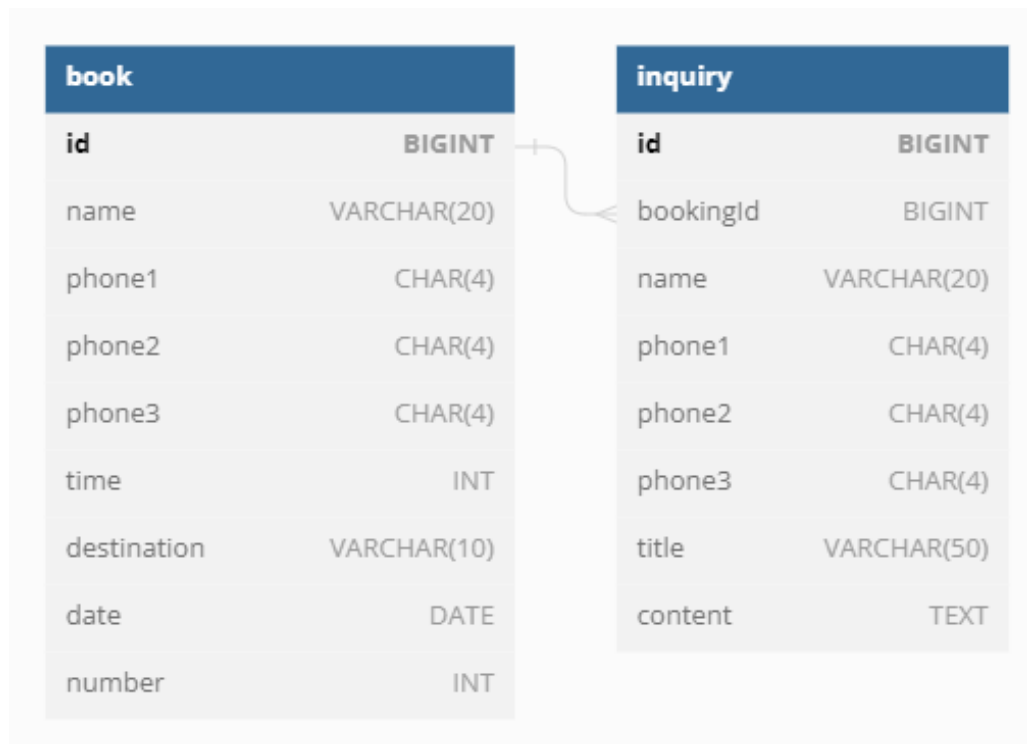
07

ERD

데이터 테이블 생성



7. ERD



08

REST API

프로젝트 진행 시 필요한 API 생성



8. REST API

HTTP Method	URI	기능	물리 VIEW
GET	/booking/add	예약하기	/booking/addForm.html
POST	/booking/add	예약 등록	Redirect:/booking/list.html
GET	/booking/list	예약 내역	/booking/list.html
POST	/booking/delete	예약 내역 삭제	Redirect:/booking/list.html
GET	/inquiry/add/{bookId}	예약내역에서 문의하기	/inquiry/addForm.html
GET	/inquiry/add	문의하기	/inquiry/addForm.html
POST	/inquiry/add	문의 등록	Redirect:/inquiry/list.html
GET	/inquiry/list	문의 내역	/inquiry/list.html

09

핵심 기능

개발 과정에서 중요했던 핵심 기능 설명



9-1. 핵심 기능(예약) – DB

Column Name	DataType	
id	BIGINT	PK, AUTO INCREMENT, NOT NULL
name	VARCHAR(20)	NOT NULL
phone1	CHAR(4)	NOT NULL
phone2	CHAR(4)	NOT NULL
phone3	CHAR(4)	NOT NULL
time	INT	NOT NULL
destination	VARCHAR(10)	NOT NULL
date	DATE	NOT NULL
number	INT	NOT NULL

phone 1,2,3은 다른 칼럼과는 다르게 고정된 값이므로 고정 길이 문자열 타입인 CHAR로 설정

모든 칼럼은 예약 시 필요한 정보이므로 반드시 작성하게 NOT NULL로 설정

id : 예약 번호

name : 예약자 이름

phone1,2,3 : 전화번호

time : 예약 시간

destination : 목적지

date : 날짜

number : 인원수

9-1. 핵심 기능(예약) – Repository_1

```
import java.util.List;
public interface SeoulRepository {

    Inquiry saveInquiry(Inquiry inquiry);
    List<Inquiry> findByNameInquiry(String name);
    List<Inquiry> findAllInquiry();

    Book saveBook(Book book);
    int getBookingCount(Book book);
    Optional<Book> findByIdBook(Long id);
    List<Book> findByNameBook(String name);
    List<Book> findAllBook();
    void deleteBook(Long id);
    void clearBeforeDate();

}
```

SeoulRepository를 인터페이스로 구현하여 필요한 메소드들을 SeoulRepository를 상속 받은 JdbcTemplateRepository에서 오버라이딩 할 수 있음

오버라이딩(Overriding)은 상위 클래스에서 정의한 메소드를 하위 클래스에서 재정의하는 것을 의미

```
@Repository
public class JdbcTemplateRepository implements SeoulRepository {

    private final JdbcTemplate jdbcTemplate;
```

9-1. 핵심 기능(예약) – Repository_2

```
Map<String, Object> parameters = new HashMap<>();
```

```
@Override
public Book saveBook(Book book) {
    SimpleJdbcInsert jdbcInsert = new SimpleJdbcInsert(jdbcTemplate).withTableName("book")
        .usingGeneratedKeyColumns("id");
    Map<String, Object> parameters = new HashMap<>();
    parameters.put("name", book.getName());
    parameters.put("phone1", book.getPhone1());
    parameters.put("phone2", book.getPhone2());
    parameters.put("phone3", book.getPhone3());
    parameters.put("time", book.getTime());
    parameters.put("destination", book.getDestination());
    parameters.put("date", book.getDate());
    parameters.put("number", book.getNumber());
    Number key = jdbcInsert.executeAndReturnKey(new MapSqlParameterSource(parameters));
    book.setId(key.longValue());
    return book;
}
```

HashMap은 Key-value 쌍을 해시코드로 매핑하여 저장해 key를 이용한 검색 및 삽입 성능이 뛰어나므로 HashMap 클래스를 이용하여 객체를 생성

Map<String, Object>을 사용함으로써, 삽입할 데이터의 정보를 각 필드별로 하나씩 전달하는 것보다 하나의 map에 한 번에 모든 데이터를 저장하고 전달

다양한 데이터 타입을 하나의 매개변수로 처리하기 위해 Object 타입을 값으로 사용

9-1. 핵심 기능(예약) – Service

```
@Service
@RequiredArgsConstructor
public class BookService {
    @Autowired
    private final JdbcTemplateRepository repository;

    public int getBookingCount(Book book) {
        return repository.getBookingCount(book);
    }

    public Book saveBook(Book book) {
        return repository.saveBook(book);
    }

    public List<Book> findByNameBook(String name) {
        return repository.findByNameBook(name);
    }
    public Optional<Book> findByIdBook(Long id){
        return repository.findByIdBook(id);
    }

    public List<Book> findAllBook() {
        return repository.findAllBook();
    }
    public void deleteBook(Long id) {
        repository.deleteBook(id);
    }

    public void clearBeforeDate() {
        repository.clearBeforeDate();
    }
}
```

@RequiredArgsConstructor는 Lombok 라이브러리에서 제공하는 어노테이션으로 초기화되지 않은 final 필드에 대해 생성자를 자동으로 생성해줌



어노테이션을 통해 repository 객체에 생성자 주입이 이루어짐

생성자 주입은 객체 생성 시점에 의존성을 외부에서 주입 받아 객체 내부에서 사용할 수 있음

9-1. 핵심 기능(예약) – Controller

```
@Controller
@RequiredArgsConstructor
public class SeoulController {

    private final BookService bookService;
    private final InquiryService inquiryService;

    // 문의하기(예약내역->문의하기) 1
    @GetMapping("/inquiry/add/{bookId}") 2
    public String inquiryAddFormFromBookList(@PathVariable Long bookId, Model model) {
        Book book = bookService.findByIdBook(bookId).get();
        model.addAttribute("book", book);
        return "inquiry/addForm";
    }
}
```

1. @GetMapping는 Spring MVC에서 GET 요청 처리하는 핸들러 메서드를 나타내는 어노테이션

데이터를 가져와 조회하는 부분이므로 RESTful API에서 GET 요청함

예약 id 값을 URL 경로에 포함하여 location.href 속성으로 전달하면 이동한 URL은 @GetMapping("/inquiry/add/{bookId}") 에서 처리됨

2. PathVariable 어노테이션을 이용하여 URL 경로에서 bookId 값을 추출

Model model은 Spring MVC의 모델 인터페이스인 Model 객체를 매개변수로 받아 데이터를 뷰로 전달

th:onclick="'|location.href='@{/inquiry/add/}\${book.id}'|"

9-1. 핵심 기능(예약) – View

```
<td><button type="button"  
  th:onclick="@{/inquiry/add/}${book.id}"  
  class="btn btn-outline-primary btn-sm">문의</button></td>
```

th:onclick은 버튼 클릭 이벤트에 대한 동작을 정의하는 Thymeleaf 속성

@{/inquiry/add/}는 Thymeleaf의 URL 표현식이며 \${book.id}는 Thymeleaf의 변수 표현식으로, book 객체의 id 속성 값을 삽입함

버튼 클릭 시 동적 생성된 URL(/inquiry/add/{bookId})로 이동

투어버스 예약 > 예약내역

예약 내역

성함을 입력하세요. 

예약번호	성함	연락처	예약날짜	출발시간	목적지	인원수	예약취소	문의하기
3	이미자	1122	2023-04-20	14	영동	3	<button>취소</button>	<button>문의</button>
33	홍	1122	2023-04-07	11	강남	10	<button>취소</button>	<button>문의</button>
35	박보검	1122	2023-04-07	11	강남	1	<button>취소</button>	<button>문의</button>

9-2. 핵심 기능(문의) – DB

Column Name	DataType	
id	BIGINT	PK, AUTO INCREMENT, NOT NULL
bookingId	BIGINT	FK
name	VARCHAR(20)	NOT NULL
phone1	CHAR(4)	NOT NULL
phone2	CHAR(4)	NOT NULL
phone3	CHAR(4)	NOT NULL
title	VARCHAR(50)	NOT NULL
content	TEXT	NOT NULL

bookingId는 foreign key, 예약을 하지 않은 사람이 문의할 수 있으므로 NULL이 가능하게 설정

id와 bookingId가 대량의 데이터가 생기게 되면 INT 타입에서 BIGINT 타입으로 변경하기에 어려움이 있어 추후 변경을 생각하여 BIGINT로 설정

id : 문의 번호

bookingId : 예약 번호

name : 예약자 이름

phone1,2,3 : 전화번호

title : 문의 제목

content : 문의 내용

9-2. 핵심 기능(문의) – Repository_1

```
import java.util.List;
public interface SeoulRepository {

    Inquiry saveInquiry(Inquiry inquiry);
    List<Inquiry> findByNameInquiry(String name);
    List<Inquiry> findAllInquiry();

    Book saveBook(Book book);
    int getBookingCount(Book book);
    Optional<Book> findByIdBook(Long id);
    List<Book> findByNameBook(String name);
    List<Book> findAllBook();
    void deleteBook(Long id);
    void clearBeforeDate();

}
```

`Inquiry saveInquiry(Inquiry inquiry)`는 `Inquiry` 객체를 매개변수로 받아들이고, `Inquiry` 객체를 반환

`List<Inquiry> findByNameInquiry(String name)`는 이름에 해당하는 `Inquiry` 객체들을 검색하여 `List` 형태로 반환

`List<Inquiry> findAllInquiry()`는 저장된 모든 `Inquiry` 객체들을 검색하여 `List` 형태로 반환

이때 `List` 형태인 이유는 여러 개의 객체를 한 번에 처리할 수 있어 메소드의 유연성과 확장성을 높이기 위해 사용

9-2. 핵심 기능(문의) – Repository_2

```
Number key = jdbcInsert.executeAndReturnKey(new MapSqlParameterSource(parameters));
```

```
@Override
public Inquiry saveInquiry(Inquiry inquiry) {

    SimpleJdbcInsert jdbcInsert = new SimpleJdbcInsert(jdbcTemplate)
        .withTableName("inquiry").usingGeneratedKeyColumns("id");
    Map<String, Object> parameters = new HashMap<>();
    parameters.put("bookingId", inquiry.getBookingId());
    parameters.put("name", inquiry.getName());
    parameters.put("phone1", inquiry.getPhone1());
    parameters.put("phone2", inquiry.getPhone2());
    parameters.put("phone3", inquiry.getPhone3());
    parameters.put("title", inquiry.getTitle());
    parameters.put("content", inquiry.getContent());
    Number key = jdbcInsert.executeAndReturnKey(new MapSqlParameterSource(parameters));
    inquiry.setId(key.longValue());
    return inquiry;
}
```

MapSqlParameterSource는 매개변수 목록을 Map 형태로 저장

Map 형태로 저장하는 이유는 테이블의 열과 값 매핑 표현과 키와 값에 다양한 타입의 데이터를 저장할 수 있기 때문

executeAndReturnKey 메소드는 데이터베이스에 데이터를 삽입하고, 자동 생성된 primary key 값을 반환

이를 통해 DB에 데이터 삽입이 아래의 그림과 같이 이루어짐

SELECT * FROM INQUIRY;

ID	BOOKINGID	NAME	PHONE1	PHONE2	PHONE3	TITLE	CONTENT
1	33	홍길동	010	2222	2222	문의합니다	문의

9-2. 핵심 기능(문의) – Service

```
@Service
@RequiredArgsConstructor
public class InquiryService {
    @Autowired
    private final JdbcTemplateRepository repository;

    public Inquiry saveInquiry(Inquiry inquiry) {
        return repository.saveInquiry(inquiry);
    }

    public List<Inquiry> findByNameInquiry(String name) {
        return repository.findByNameInquiry(name);
    }

    public List<Inquiry> findAllInquiry() {
        return repository.findAllInquiry();
    }
}
```

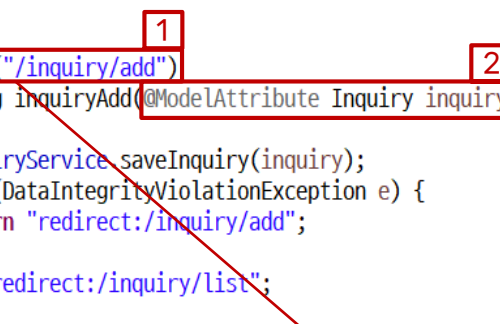


@Service을 사용하여 클래스를 서비스로 등록하면, 스프링은 해당 클래스의 인스턴스를 빈(Been)으로 생성하고 관리하며 의존성 주입 등의 기능을 사용할 수 있게 해 줌

9-2. 핵심 기능(문의) – Controller

```
// 문의하기
@GetMapping("/inquiry/add")
public String inquiryAddForm(Model model) {
    Book book = new Book();
    model.addAttribute("book", book);
    return "inquiry/addForm";
}

// 문의 등록
@PostMapping("/inquiry/add")
public String inquiryAdd(@ModelAttribute Inquiry inquiry) {
    try {
        inquiryService.saveInquiry(inquiry);
    } catch (DataIntegrityViolationException e) {
        return "redirect:/inquiry/add";
    }
    return "redirect:/inquiry/list";
}
```



```
<input type="submit" th:onclick="|location.href='@{/inquiry/add} '| "
      value="취소" class="btn btn-secondary col-2 py-3" />
<input type="submit" value="접수" class="btn btn-primary col-2 py-2" />
```

1. 취소 버튼은 클릭하면 /inquiry/add 경로로 이동하도록 되어 있고, 접수 버튼은 클릭하면 현재 위치한 URL을 POST 방식으로 서버로 전송

@PostMapping("/inquiry/add")는 /inquiry/add 경로로 들어오는 POST 요청을 처리하도록 설정

2. @ModelAttribute는 요청 파라미터(request parameter)를 해당 모델 객체로 바인딩하는 역할

@ModelAttribute Inquiry inquiry는 해당 요청 매개변수로 Inquiry 타입의 객체를 사용하고, 이 객체에 클라이언트의 요청 파라미터를 바인딩하여 전달

9-2. 핵심 기능(문의) – View

1. readonly는 사용자가 해당 필드를 클릭하거나 수정할 수 없으며 서버에서 전달한 값 확인만 가능한 속성인데 예약번호는 사용자가 작성하는 부분이 아니므로 이 속성을 사용함

th:value="{book.id}" 는 서버에서 전달 받은 book 객체의 id 필드 값을 출력하는 속성

2. 연락처에 숫자만 입력 받기 위해 입력된 값에 대해 유효성을 검사함

입력된 값에서 숫자와 점(.)을 제외한 모든 문자를 찾아낸 후 그 문자들을 빈 문자열로 대체하여 문자를 제거한 후 소수점(.)이 두 개 이상 입력될 경우, 두 번째 소수점 이후의 값을 모두 삭제

문의 접수

투어버스 관련 궁금하신 점을 남겨주시면 빠른시간 안에 답변해 드리겠습니다.

예약번호

이름

연락처 - -

제작을 입력해주세요

문의사항을 입력해주세요

위스 접수

```
<div class="row my-3">
  <div class="row col">
    <label for="bookingId" class="col-2">예약번호</label> <input
      type="text" name="bookingId" id="bookingId"
      class="form-control box" readonly th:value="{book.id}" />
    </div>
    <div class="row col">
      <label for="name" class="col-2">이름</label> <input type="text"
        id="name" name="name" class="form-control box" placeholder="홍길동"
        th:value="{book.name}" required />
      </div>
    </div>
    <div class="row phoneNumber my-3">
      <label for="phone1" class="col-2">연락처</label> <input type="tel"
        name="phone1" id="phone1" value="010" class="form-control phone"
        required
        oninput="this.value = this.value.replace(/[^\d-]/g, '').replace(/(\d*)\./g, '$1');"/>
      <span class="col-1"></span> <input type="tel" name="phone2"
        id="phone2" placeholder="0000" class="form-control phone"
        th:value="{book.phone2}" required
        oninput="this.value = this.value.replace(/[^\d-]/g, '').replace(/(\d*)\./g, '$1');"/>
      <span class="col-1"></span> <input type="tel" name="phone3"
        id="phone3" placeholder="0000" class="form-control phone"
        th:value="{book.phone3}" required
        oninput="this.value = this.value.replace(/[^\d-]/g, '').replace(/(\d*)\./g, '$1');"/>
    </div>
  </div>
```

10

배포

배포 과정 설명



10-1. 배포 AWS 인스턴스 생성

1. 인스턴스 생성 과정

- Ubuntu를 사용하여 제작

2. 키페어는 RSA는 비대칭키 암호화 방식 pem은 RSA 키페어를 저장하는 데 사용되는 파일 형식

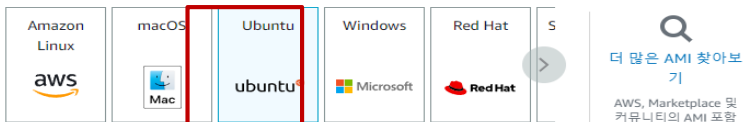
▼ 애플리케이션 및 OS 이미지(Amazon Machine Image) 정보

AMI는 인스턴스를 시작하는 데 필요한 소프트웨어 구성(운영 체제, 애플리케이션 서버 및 애플리케이션)이 포함된 템플릿입니다. 아래에서 찾고 있는 항목이 보이지 않으면 AMI를 검색하거나 찾아보십시오.

🔍 수천 개의 애플리케이션 및 OS 이미지를 포함하는 전체 카탈로그 검색

최근 사용

Quick Start



Amazon Machine Image(AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type
ami-04cebc8d6c4f297a3 (64비트(x86)) / ami-0084ff4520f7fd92a (64비트(Arm))
가상화: hvm ENA 활성화됨: true 루트 디바이스 유형: ebs

프리 티어 사용 가능

설명

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-03-25

아키텍처

AMI ID

64비트(x86)

ami-04cebc8d6c4f297a3

확인된 공급 업체

키 페어 생성

키 페어를 사용하면 인스턴스에 안전하게 연결할 수 있습니다.

아래에 키 페어의 이름을 입력합니다. 메시지가 표시되면 프라이빗 키를 사용자 컴퓨터의 안전하고 액세스 가능한 위치에 저장합니다. 나중에 인스턴스에 연결할 때 필요합니다. [자세히 알아보기](#)

키 페어 이름

seoulmate1

이름에는 최대 255개의 ASCII 문자를 포함할 수 있습니다. 앞 또는 뒤에 공백을 포함할 수 없습니다.

키 페어 유형

☒ RSA

RSA 암호화된 프라이빗 및 퍼블릭 키 페어

☐ ED25519

ED25519 암호화된 프라이빗 및 퍼블릭 키 페어(Windows 인스턴스에는 지원되지 않음)

프라이빗 키 파일 형식

☒ .pem

OpenSSH와 함께 사용

☐ .ppk

PuTTY와 함께 사용

취소

키 페어 생성

10-1. 배포 AWS 인스턴스 생성

1. 네트워크 설정

- 테스트 용임으로 HTTPS 트래픽 허용을 설정

2. Default 보안을 모든 트래픽으로 설정함

▼ 네트워크 설정 정보

편집

네트워크 정보

vpc-0f498227abaeff87c

서브넷 정보

기본 설정 없음(가용 영역의 기본 서브넷)

퍼블릭 IP 자동 할당 정보

활성화

방화벽(보안 그룹) 정보

보안 그룹은 인스턴스에 대한 트래픽을 제어하는 방화벽 규칙 세트입니다. 특정 트래픽이 인스턴스에 도달하도록 허용하는 규칙을 추가합니다.

☒ 보안 그룹 생성

☐ 기존 보안 그룹 선택

다음 규칙을 사용하여 'launch-wizard-2'(이)라는 새 보안 그룹을 생성합니다.

☒에서 SSH 트래픽 허용
인스턴스 연결에 도움

허치 무관
0.0.0.0/0

☒인터넷에서 HTTPS 트래픽 허용
예를 들어 웹 서버를 생성할 때 엔드포인트를 지정하려면

☒인터넷에서 HTTP 트래픽 허용
예를 들어 웹 서버를 생성할 때 엔드포인트를 지정하려면

△ 소스가 0.0.0.0/0인 규칙은 모든 IP 주소에서 인스턴스에 액세스하도록 허용합니다. 알려진 IP 주소의 액세스만 허용하도록 보안 그룹을 설정하는 것이 좋습니다.

아웃바운드 규칙 (1/1)

태그 관리

아웃바운드 규칙 편집

Q 보안 그룹 규칙 필터

< 1 >

<input checked="" type="checkbox"/>	Name	보안 그룹 규칙 ID	IP 버전	유형	프로토콜	포트 범위	대상	설명
<input checked="" type="checkbox"/>	-	sgr-00f02d9432dfdf9c9	IPv6	모든 트래픽	전체	전체	:/0	-

인바운드 규칙 (1/1)

태그 관리

인바운드 규칙 편집

Q 보안 그룹 규칙 필터

< 1 >

<input checked="" type="checkbox"/>	Name	보안 그룹 규칙 ID	IP 버전	유형	프로토콜	포트 범위	소스	설명
<input checked="" type="checkbox"/>	-	sgr-0c0de6e98713521...	IPv4	모든 트래픽	전체	전체	0.0.0.0/0	-

10-2. 배포 AWS RDS my SQL 설정

1. 인스턴스에 연결

- sudo apt update
- jdk 11 버전

인스턴스에 연결 정보

다음 옵션 중 하나를 사용하여 인스턴스 i-06dc968a2eb3d41cb (seoulmate)에 연결

EC2 인스턴스 연결 Session Manager SSH 클라이언트 EC2 직렬 콘솔

인스턴스 ID

i-06dc968a2eb3d41cb (seoulmate)

1. SSH 클라이언트를 엽니다.
2. 프라이빗 키 파일을 찾습니다. 이 인스턴스를 시작하는 데 사용되는 키는 seoulmate.pem입니다.
3. 필요한 경우 이 명령을 실행하여 키를 공개적으로 볼 수 없도록 합니다.
`chmod 400 seoulmate.pem`
4. 퍼블릭 DNS를(를) 사용하여 인스턴스에 연결:
`ec2-3-39-25-9.ap-northeast-2.compute.amazonaws.com`

예:

`ssh -i "seoulmate.pem" ubuntu@ec2-3-39-25-9.ap-northeast-2.compute.amazonaws.com`

참고: 대부분의 경우 지정된 사용자 이름은 정확합니다. 하지만 AMI 사용 지침을 읽고 AMI 소유자가 기본 AMI 사용자 이름을 변경했는지 확인하십시오.

```
Usage of /: 20.1% of 7.57GB   Users logged in: 0
Memory usage: 20%          IPv4 address for eth0: 172.31.37.100
Swap usage: 0%
```

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See <https://ubuntu.com/esm> or run: `sudo pro status`

The list of available updates is more than a week old.
To check for new updates run: `sudo apt update`

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

```
ubuntu@ip-172-31-37-100:~$ ls
ubuntu@ip-172-31-37-100:~$ ls
ubuntu@ip-172-31-37-100:~$ ls
ubuntu@ip-172-31-37-100:~$ sudo apt update
```

```
sudo apt install openjdk-19-jre-headless # version 19.0.2-7-0ubuntu3~22.04
sudo apt install openjdk-19-jre-headless # version 19.0.2+7-0ubuntu3~22.04
ubuntu@ip-172-31-37-100:~$ ls
ubuntu@ip-172-31-37-100:~$ sudo apt-get install openjdk-11-jdk
```

10-3. 배포 AWS RDS 데이터베이스 생성

1. RDS 설정

- My SQL로 설정, 버전 8.0.32
- DB 식별자는 database-1로 설정
- 관리자 admin
- 암호 green1111로 설정


데이터베이스 생성 방식 선택 정보


☒ 표준 생성
가용성, 보안, 백업 및 유지 관리에 대한 옵션을 포함하여 모든 구성 옵션을 설정합니다.


☐ 손쉬운 생성
권장 모범 사례 구성을 사용합니다. 일부 구성 옵션은 데이터베이스를 생성한 후 변경할 수 있습니다.


엔진 옵션


엔진 유형 정보


☐ Aurora (MySQL Compatible)



☐ Aurora (PostgreSQL Compatible)


☒ MySQL


☐ MariaDB


☐ PostgreSQL


☐ Oracle


☐ Microsoft SQL Server


엔진 버전

MySQL 8.0.32

설정

DB 클러스터 식별자 정보
DB 클러스터 이름을 입력합니다. 이름은 현재 AWS 리전에서 AWS 계정이 소유하는 모든 DB 클러스터에 대해 고유해야 합니다.

database-1

DB 클러스터 식별자는 대소문자를 구분하지 않지만 모두 소문자로 저장됩니다(예: "mydbcluster"). 제약 조건: 1~60자의 영숫자 또는 하이픈으로 구성되어야 합니다. 첫 번째 문자는 글자여야 합니다. 하이픈은 연속으로 2개를 포함할 수 없습니다. 끝에 하이픈이 올 수 없습니다.

▼ 자격 증명 설정

마스터 사용자 이름 정보
DB 클러스터의 마스터 사용자에 대한 로그인 ID를 입력하세요.

admin

1~16자의 영숫자. 첫 번째 문자는 글자여야 합니다.

☐ AWS Secrets Manager에서 마스터 보안 인증 정보 관리
Secrets Manager에서 마스터 사용자 보안 인증을 관리합니다. RDS는 사용자 대신 암호를 생성하고 수명 주기 동안 이를 관리할 수 있습니다.

☒ Secrets Manager에서 마스터 사용자 보안 인증 정보를 관리하는 경우 일부 RDS 기능은 지원되지 않습니다. 자세히 알아보기

☐ 암호 자동 생성
Amazon RDS에서 사용자를 대신하여 암호를 생성하거나 사용자가 직접 암호를 지정할 수 있습니다.

마스터 암호 정보

제약 조건: 8자 이상의 인쇄 가능한 ASCII 문자. 다음을 포함할 수 없습니다. /(슬래시), '(작은따옴표), '(큰따옴표) 및 @(앳 기호).

마스터 암호 확인 정보

10-4. 배포 AWS RDS 데이터베이스 생성

1. 연결

- default 그룹에 설정, 퍼블릭 액세스는 허용함
- 보안 그룹 이름 my db
- 베이스 포트 3306
- 클러스터로 설정

연결 정보

컴퓨팅 리소스

이 데이터베이스의 컴퓨팅 리소스에 대한 연결을 설정할지를 선택합니다. 연결을 설정하면 컴퓨팅 리소스가 이 데이터베이스에 연결할 수 있도록 연결 설정이 자동으로 변경됩니다.

☒ EC2 컴퓨팅 리소스에 연결 안 함
이 데이터베이스의 컴퓨팅 리소스에 대한 연결을 설정하지 않습니다. 나중에 컴퓨팅 리소스에 대한 연결을 수동으로 설정할 수 있습니다.

☐ EC2 컴퓨팅 리소스에 연결
이 데이터베이스의 EC2 컴퓨팅 리소스에 대한 연결을 설정합니다.

Virtual Private Cloud(VPC) 정보

VPC를 선택합니다. VPC는 이 DB 클라우드의 가상 네트워킹 환경을 정의합니다.

Default VPC (vpc-0f498227abaeff87c)
4 서브넷, 4 Availability Zones

해당 DB 서브넷 그룹이 있는 VPC만 나열됩니다.

☒ 데이터베이스를 생성한 후에는 VPC를 변경할 수 없습니다.

DB 서브넷 그룹 정보

DB 서브넷 그룹을 선택합니다. DB 서브넷 그룹은 선택한 VPC에서 DB 클러스터가 어떤 서브넷과 IP 범위를 사용할 수 있는지를 정의합니다.

default-vpc-0f498227abaeff87c
4 서브넷, 4 Availability Zones

퍼블릭 액세스 정보

☒ 예
RDS는 클러스터에 퍼블릭 IP 주소를 할당합니다. VPC 외부의 Amazon EC2 인스턴스 및 다른 리소스가 클러스터에 연결할 수 있습니다. VPC 내부의 리소스도 클러스터에 연결할 수 있습니다. 클러스터에 연결할 수 있는 리소스를 지정하는 VPC 보안 그룹을 하나 이상 선택합니다.

☐ 아니요
RDS는 클러스터에 퍼블릭 IP 주소를 할당하지 않습니다. VPC 내부의 Amazon EC2 인스턴스 및 다른 리소스만 클러스터에 연결할 수 있습니다. 클러스터에 연결할 수 있는 리소스를 지정하는 VPC 보안 그룹을 하나 이상 선택합니다.

VPC 보안 그룹(방화벽) 정보

데이터베이스에 대한 액세스를 허용할 VPC 보안 그룹을 하나 이상 선택합니다. 보안 그룹 규칙이 적절한 수신 트래픽을 허용하는지 확인합니다.

☐ 기존 항목 선택
기존 VPC 보안 그룹 선택

☒ 새로 생성
새 VPC 보안 그룹 생성

새 VPC 보안 그룹 이름

mydb

추가 구성

데이터베이스 포트 정보

데이터베이스가 애플리케이션 연결에 사용할 TCP/IP 포트입니다.

3306

데이터베이스 역할

☒ 클러스터
클러스터 내 모든 데이터베이스 인스턴스에 보안 그룹 적용
(리전 클러스터만 지원됨)

☐ 인스턴스
클러스터의 일부가 아닌 개별 데이터베이스 인스턴스에 보안 그룹 적용

RDS 데이터베이스

보안 그룹이 EC2 인스턴스에 추가됩니다. 보안 그룹은 또한 EC2 인스턴스가 데이터베이스 포트에 액세스할 수 있도록 허용하는 인바운드 규칙과 함께 클러스터에 추가됩니다.

RDS 데이터베이스 선택

RDS 데이터베이스 생성

10-5. 배포 AWS RDS 데이터베이스 생성

1. 프리티어 설정

- 테스트용 임의로 설정
- 자동 조정 활성화를 비활성화, 자동으로 스토리지를 생성하여, 저장용량을 만듦 (추가 요금 설정하지 않기 위해)

☐ 프로덕션
고가용성 및 빠르고 일관된 성능을 위해 기본값을 사용하세요.

☐ 개발/테스트
이 인스턴스는 프로덕션 환경 외부에서 개발 용도로 마련되었습니다.

☒ 프리 티어
RDS 프리 티어를 사용하여 새로운 애플리케이션을 개발하거나, 기존 애플리케이션을 테스트하거나 Amazon RDS에서 실수 경험을 쌓을 수 있습니다. [정보](#)

가용성 및 내구성

배포 옵션 정보
아래의 배포 옵션을 위해서 선택한 연진에서 지원하는 배포 옵션으로 제한됩니다.

- 다중 AZ DB 클러스터 - 신규
기본 DB 인스턴스와 읽기 가능한 예비 DB 인스턴스 2개가 있는 DB 클러스터를 생성합니다. 각 DB 인스턴스는 서로 다른 가용 영역(AZ)에 있습니다. 고가용성, 데이터 이중화를 제공하고 읽기 워크로드를 처리하기 위한 용량을 늘립니다.
- 다중 AZ DB 인스턴스(다중 AZ DB 클러스터 스냅샷에는 지원되지 않음)
다른 AZ에 기본 DB 인스턴스와 예비 DB 인스턴스를 생성합니다. 고가용성 및 데이터 이중화를 제공하지만 예비 DB 인스턴스는 읽기 워크로드에 대한 연장을 지원하지 않습니다.
- 단일 DB 인스턴스(다중 AZ DB 클러스터 스냅샷에는 지원되지 않음)
예비 DB 인스턴스가 없는 단일 DB 인스턴스를 생성합니다.

스토리지

스토리지 유형 [정보](#)

범용 SSD(gp2)
볼륨 크기에 따라 기준 성능 결정

할당된 스토리지 [정보](#)

20 GiB

최소값은 20GiB이고, 최대값은 6,144GiB입니다.

스토리지 자동 조정 [정보](#)

애플리케이션의 필요에 따라 데이터베이스 스토리지의 동적 조정 지원을 제공합니다.

☐ 스토리지 자동 조정 활성화
이 기능을 활성화하면 지정한 임계값 초과 후 스토리지를 늘릴 수 있습니다.

10-6. 배포 AWS RDS 데이터베이스 생성

1. 초기 데이터베이스 이름
 - mydb 설정
 - 자동 백업 비활성화
2. My SQL 커넥션 활성화

▼ 추가 구성

데이터베이스 옵션, 암호화 커짐, 백업 꺼짐, 역추적 꺼짐, 유지 관리, CloudWatch Logs, 삭제 방지 꺼짐.

데이터베이스 옵션

초기 데이터베이스 이름 [정보](#)

mydb

데이터베이스 이름을 지정하지 않으면 Amazon RDS에서 데이터베이스를 생성하지 않습니다.

DB 파라미터 그룹 [정보](#)

default.mysql8.0

옵션 그룹 [정보](#)

default:mysql-8-0

백업

- ☐ 자동 백업을 활성화합니다.
데이터베이스의 특정 시점 스냅샷을 생성합니다.

Setup New Connection

Connection Name: mydb Type a name for the connection

Connection Method: Standard (TCP/IP) Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname: emv.ap-northeast-2.rds.amazonaws.com Port: 3306 Name or IP address of the server host - and TCP/IP port.

Username: admin Name of the user to connect with.

Password: The user's password. Will be requested later if it's not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

10-7. 배포 AWS RDS my SQL 설정

1. Query 문 작성

- id = 자동 증가 select @@autocommit; set autocommit=1; 설정
- Book table(예약), inquiry table(문의)
- 아래와 같이 작성

Table: book

Columns:

id	bigint AI PK
name	varchar(20)
phone1	char(4)
phone2	char(4)
phone3	char(4)
time	int
destination	varchar(10)
date	date
number	int

Table: inquiry

Columns:

id	bigint AI PK
bookingId	bigint
name	varchar(20)
phone1	char(4)
phone2	char(4)
phone3	char(4)
title	varchar(50)
content	text

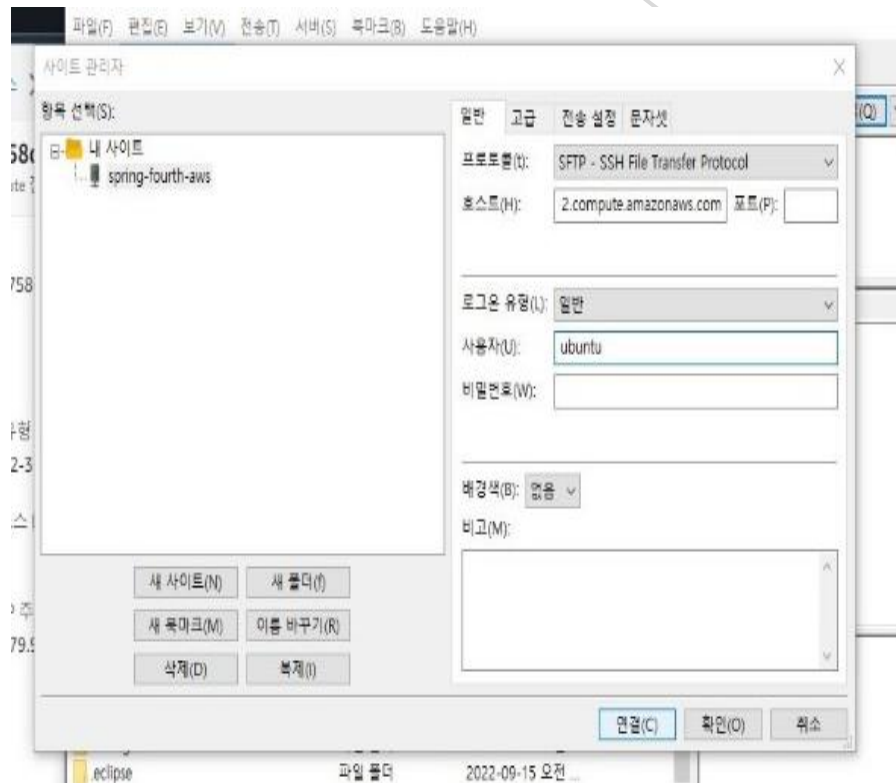
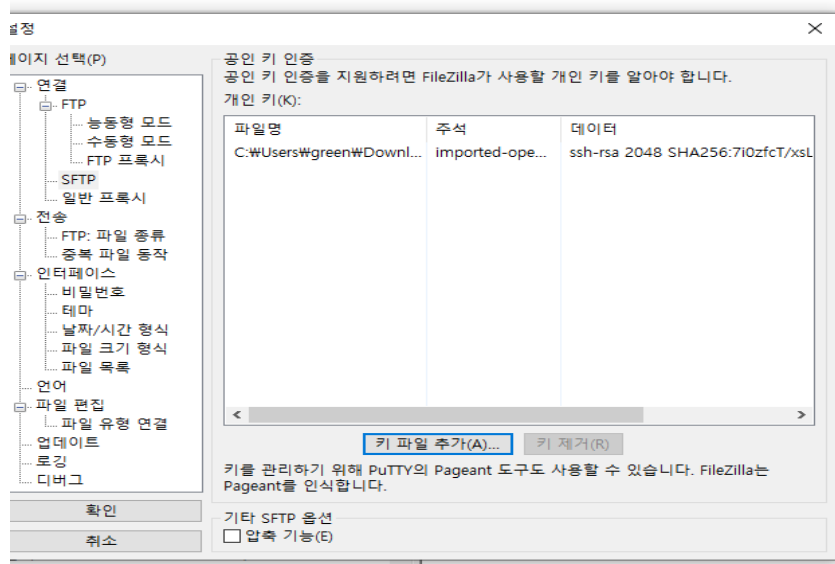
```
1 • use mydb;
2 • create table book (
3   id bigint auto_increment,
4   name varchar(20),
5   phone1 char(4),
6   phone2 char(4),
7   phone3 char(4),
8   time int,
9   destination varchar(10),
10  date date,
11  number int,
12  primary key(id)
13 );
14 • create table inquiry (
15   id bigint auto_increment,
16   bookingId bigint,
17   name varchar(20),
18   phone1 char(4),
19   phone2 char(4),
20   phone3 char(4),
21   title varchar(50),
22   content text,
23   primary key(id),
24   foreign key(bookingId) references book(id)
25   on delete cascade
26 );
27
28 • select @@autocommit;
29 • set autocommit=1;
```

10-8. 배포 fileZila

1. filezila

- 공인키 등록, 인스턴스 서버 연결

2. 폴더 생성




10-9. 배포 fileZila


1. Jar 파일 생성

2. 폴더

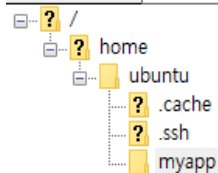
- java jar 파일을 업로드


```
C:\Users\green\Downloads\seoulmate-tour-final\seoulmate-tour>gradlew.bat build
```

 seoulmate-tour-0.0.1-SNAPSHOT 2023-05-06 오후 11:43

 seoulmate-tour-0.0.1-SNAPSHOT-plain 2023-05-06 오후 11:43

리모트 사이트: /home/ubuntu/myapp



파일명	크기	파일 유형	최종 수정	권한	소유자/그룹
...					
 seoulmate-tour-0.0.1...	32,542,497	Executable...	2023-05-06 ...	-rw-rw-r--	ubuntu ut

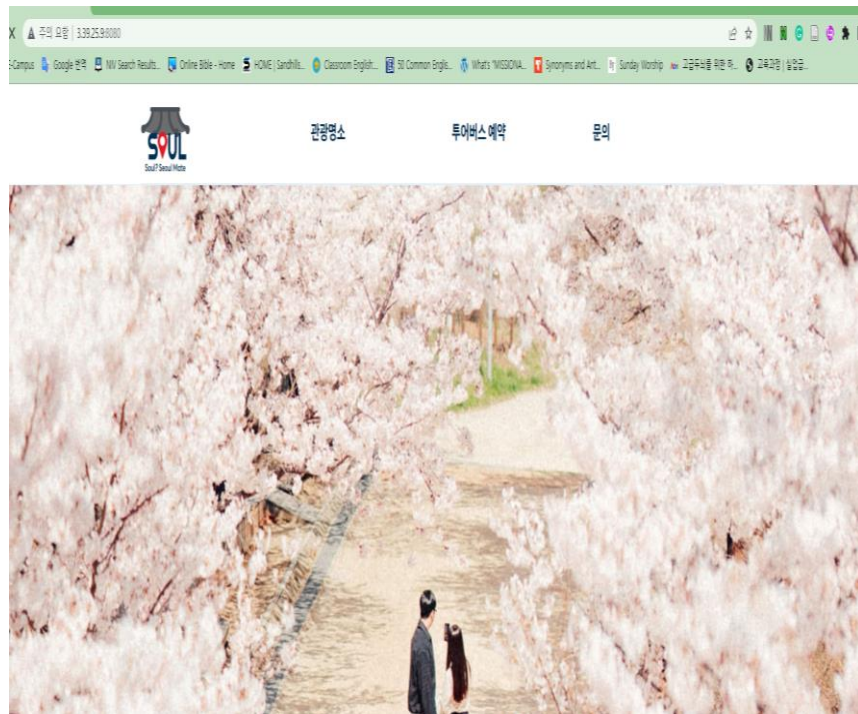
10-10. 배포 fileZila

1. 실행 권한 추가

2. 백그라운드 실행

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-37-100:~$ ls
myapp
ubuntu@ip-172-31-37-100:~$ cd myapp
ubuntu@ip-172-31-37-100:~/myapp$ la -al
total 31788
drwxrwxr-x 2 ubuntu ubuntu 4096 May 6 14:30 .
drwxr-x--- 5 ubuntu ubuntu 4096 May 6 14:30 ..
-rw-rw-r-- 1 ubuntu ubuntu 32542497 May 6 14:30 seoulmate-tour-0.0.1-SNAPSHOT.jar
ubuntu@ip-172-31-37-100:~/myapp$ sudo chmod +x seoulmate-tour-0.0.1-SNAPSHOT.jar
```

```
ubuntu@ip-172-31-37-100:~$ cd myapp
ubuntu@ip-172-31-37-100:~/myapp$ java -jar seoulmate-tour-0.0.1-SNAPSHOT.jar &
```



11

트러블 슈팅

개발 과정에서 발생한 오류 문제 해결 과정



11. 트러블 슈팅



예약 시 날짜를 선택하지 않을 경우 에러 발생함

예약 내역에서 문의 이동시 예약 내역의 저장 된 정보를 불러오지 못함

예약내역에서 문의 이동하여 문의 시 저장된 후 예약 번호가 보이지 않음

버스 탑승 기준 인원을 초과할 경우 예약을 불가능하게 처리할 수 없음

11. 트러블 슈팅 (1)

- 예약 시 날짜를 선택하지 않을 경우 에러 발생함

```
<div class="col-5 px-2 py-3 my-5">
  <div class="row">
    <label for="date">예약 날짜</label>
    <div id="datepicker" class="form-control">
      <input type="hidden" id="my_hidden_input" name="date" required />
    </div>
  </div>
</div>
```



```
<div class="col-5 px-2 py-3 my-5">
  <div class="row">
    <label for="date">예약 날짜</label>
    <div id="datepicker" class="form-control">
      <input type="date" id="my_hidden_input" name="date" class="blind" required />
    </div>
  </div>
</div>
```

Datepicker에서 value 값을 받지 못해 오류가 발생했고, 해결하기 위해 input 태그의 type을 "hidden"에서 "date"로 수정하고, 화면에는 보이지 않도록 CSS 클래스인 "blind"를 적용하여 사용자가 반드시 날짜를 선택해야 페이지가 표시되도록 변경

11. 트러블 슈팅 (2)

- 예약 내역에서 문의 이동시 예약 내역의 입력 정보를 불러오지 못함

```
Book saveBook(Book book);  
List<Book> findByNameBook(String name);  
List<Book> findAllBook();
```



```
Book saveBook(Book book);  
List<Book> findByNameBook(String name);  
List<Book> findAllBook();  
Optional<Book> findByIdBook(Long id);
```

예약 번호를 기반으로 한 세부사항을 가져올 수 있는 메소드가 없어서 세부사항을 불러오지 못해 repository에 객체를 한 개만 반환하는 메소드를 추가

이 메소드는 객체의 고유한 값인 id를 사용하여 해당 객체의 예약 내역 세부사항을 가져올 수 있도록 하여 사용자는 예약 번호를 입력하여 문의를 작성할 수 있게 변경됨

11. 트러블 슈팅 (3)

- 예약내역에서 문의 이동하여 문의 시 저장된 후 예약 번호가 보이지 않음

```
<div class="row col">
  <label for="bookingId" class="col-2">예약번호</label>
  <input type="text" name="bookingId" id="bookingId"
    class="form-control box" disabled th:value="${book.id}">
</div>
```

예약번호 65

이름 이미지

연락처 010 - 1111 - 5555

제목을 입력해주세요

문의번호	예약번호	성함	연락처	제목	내용
1	33	홍길동	2222	문의합니다	내용보기
2		이미지	5555	문의	내용보기

```
<div class="row col">
  <label for="bookingId" class="col-2">예약번호</label>
  <input type="text" name="bookingId" id="bookingId"
    class="form-control box" readonly th:value="${book.id}">
</div>
```

예약번호 65

이름 이미지

연락처 010 - 1111 - 5555

제목을 입력해주세요

문의번호	예약번호	성함	연락처	제목	내용
1	33	홍길동	2222	문의합니다	내용보기
2		이미지	5555	문의	내용보기
3	65	이미지	5555	문의/2	내용보기

disabled속성은 요소가 폼 데이터로 전송되지 않아서 문의에서 예약 번호가 보이지 않았기 때문에 readonly로 변경하여 예약 번호가 보이도록 수정

11. 트러블 슈팅 (4)

- 버스 탑승 기준 인원을 초과할 경우 예약을 불가능하게 처리할 수 없음

```
Book saveBook(Book book);  
List<Book> findByNameBook(String name);  
List<Book> findAllBook();  
Optional<Book> findByIdBook(Long id);  
void deleteBook(Long id);  
void clearBeforeDate();
```



```
Book saveBook(Book book);  
List<Book> findByNameBook(String name);  
List<Book> findAllBook();  
Optional<Book> findByIdBook(Long id);  
int getBookingCount(Book book);  
void deleteBook(Long id);  
void clearBeforeDate();
```

인원 초과로 예약이 불가능한 경우를 처리할 수 있는 메소드가 없어서 인원이 초과돼도 예약이 진행됐기 때문에 getBookingCount라는 새로운 메소드를 추가한 뒤 호출
이 메소드는 목적지, 시간, 날짜를 기준으로 예약된 인원 수를 조회하여 반환

```
@Override  
public int getBookingCount(Book book) {  
    String destination = book.getDestination();  
    int time = book.getTime();  
    String date = book.getDate();  
    String query = "select sum(number) from book where destination like ? and time=? and date like ?";  
    int bookingCount = jdbcTemplate.query(query, new ResultSetExtractor<Integer>() {  
        @Override  
        public Integer extractData(ResultSet rs) throws SQLException, DataAccessException {  
            if (rs.next()) {  
                return rs.getInt(1);  
            } else {  
                return null;  
            }  
        }  
    }, destination, time, date);  
    log.info("bookingCount >>" + bookingCount);  
    return bookingCount;  
}
```

11. 트러블 슈팅 (5)

● 배포 과정에서 pem 경로 잘못 설정으로 인한 오류 발생

```
Microsoft Windows [Version 10.0.22621.1555]  
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\hi-sinchon>D:
```

```
D:\>cd D:\webservice\web\workspace\seoulmate-tour
```

```
D:\webservice\web\workspace\seoulmate-tour>cd myapp/  
지정된 경로를 찾을 수 없습니다.
```

```
D:\webservice\web\workspace\seoulmate-tour>|
```



```
C:\Users\hi-sinchon>D:
```

```
D:\>cd webservice
```

```
D:\webservice>ssh -i "seoulmate-tour.pem" ubuntu@ec2-3-139-90-52.us-east-2.compute.amazonaws.com  
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-1031-aws x86_64)
```

```
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage
```

```
System information as of Mon Apr 24 01:16:32 UTC 2023
```

System load:	0.0	Processes:	98
Usage of /:	40.8% of 7.57GB	Users logged in:	1
Memory usage:	47%	IPv4 address for eth0:	172.31.43.53
Swap usage:	0%		

```
* Ubuntu Pro delivers the most comprehensive open source security and  
compliance features.
```

```
https://ubuntu.com/aws/pro
```

```
* Introducing Expanded Security Maintenance for Applications.  
Receive updates to over 25,000 software packages with your  
Ubuntu Pro subscription. Free for personal use.
```

문제가 발생한 경로로 이동하려고 할 때 경로를 찾을 수 없는 오류가 발생했고, 해당 경로가 존재하지 않는 것으로 추정되어 경로를 정확하게 입력했는지 확인 후 정확한 경로를 입력하여 오류 해결

11. 트러블 슈팅 (6)

- 배포 과정에서 서버가 중복되어 켜져 있음으로 인한 오류 발생

```
*****
APPLICATION FAILED TO START
*****

Description:
Web server failed to start. Port 8080 was already in use.

Action:
Identify and stop the process that's listening on port 8080 or configure this application to listen on another port.
```

배포 과정에서 서버가 중복되어
켜져 있어 포트 8080이 이미
존재한다는 오류가 발생하므로
켜져 있는 서버를 확인한 뒤
불필요하게 켜져 있는 서버를
중단하여 오류 해결



```
root      4532    1267    0 14:32 pts/0    00:00:00 sudo nohup java -jar seoulmate-tour-0.0.1-SNAPSHOT.jar
root      4533    4532    0 14:32 pts/1    00:00:00 sudo nohup java -jar seoulmate-tour-0.0.1-SNAPSHOT.jar
root      4534    4533    7 14:32 pts/1    00:00:09 java -jar seoulmate-tour-0.0.1-SNAPSHOT.jar
ubuntu    4603    1267    0 14:34 pts/0    00:00:00 ps -ef
ubuntu@ip-172-31-37-100: ~/myapp$ sudo kill -9 4532
```

12

추후 작업

프로젝트 이후 추가 작업 설명



12. 추후 작업

01 로그인

개인 정보 보호 및 고객의 예약, 문의 내역 관리를 위한 로그인 시스템 도입

02 정원 수

정원 수 초과 시 날짜 선택 불가능 기능 도입

03 결제 수단

온라인 결제를 위한 결제 시스템 도입



13

소감

프로젝트 종료 후 각자 소감



13. 소감

한민주

이번 프로젝트는 기획부터 배포까지 처음 해보는 것이라 많은 어려움이 있었지만, 팀원들과 함께 문제를 해결하면서 끈질기게 노력한 덕분에 극복할 수 있었습니다. 특히, 프론트와 백엔드 모두를 직접 경험하면서 더욱 의미 있는 시간이었습니다. 이번 프로젝트에서 팀장이라는 역할을 맡아 책임감과 부담감이 컸지만 팀원들과 같이 문제를 해결하고 서로 의견을 조율하는 과정을 통해 팀 프로젝트를 하는 것은 소통과 협업이 필수라는 것을 깨닫는 시간이었습니다. 이번 프로젝트를 통해 많은 것을 배우고 성장할 수 있었으며, 향후 프로젝트에서도 이번 경험이 큰 도움이 될 것이라고 생각합니다.

송하은

처음으로 프로그램 개발 수업을 통해 팀 프로젝트 작업을 진행하면서 새로운 경험을 하게 되어 많은 것을 깨닫는 시간이었습니다. 프론트에 관심이 많아 주로 프론트 작업을 진행하면서 웹 브라우저로 보여지는 것 뿐만 아니라 기능이 수행되기 위해서는 백엔드 쪽에서도 많은 개발과 작업이 이루어지고 있다는 것을 알게 되었습니다. 프론트와 백엔드가 서로 밀접하게 연관되어 있어 작업을 할 때 개별적으로 진행해 결과물을 내기 보다는 소통을 통해 프로그램 작업이 이루어져야 한다는 것을 배우는 시간이었습니다. 팀 프로젝트가 다양한 생각을 가진 사람들이 모여 하나의 결과물을 내야 하기 때문에 쉽지 않았지만 그만큼 더욱 소통과 협력의 중요성을 느끼는 시간이었습니다.

김정훈

이번 프로젝트는 예약 기능, 문의 기능, 그리고 관광지 페이지를 개발하는 것이 주요 목표였습니다. 팀원들과의 협업은 매우 중요했고, 각자의 역할과 책임을 명확히 정의하고 작업 일정을 체계적으로 관리했습니다. 다양한 기술과 도구를 사용하여 프로젝트를 진행했고, 이를 통해 개발 역량과 협업 능력을 향상시킬 수 있었습니다. 팀원들과의 협업을 통해 소통과 문제 해결 능력을 향상시켰습니다. 이번 프로젝트는 저에게 개발자로서의 성장을 위한 좋은 경험이었습니다. 팀원들과 함께 성장하고 가치 있는 웹사이트를 개발할 수 있어서 감사합니다.

13. 소감

송원기

2주간의 첫 팀 프로젝트 경험을 통해, 프로젝트의 디테일 부분들을 조율하면서 함께 일하는 다양한 사람들과 모인 장소에서 작업하였습니다.

프론트와 백엔드를 연관시키며 클라이언트에게 보여지고 배포될지를 공부하면서 보완해 나갔습니다. 서로 의견을 조율하는 부분에서 어려움도 있었지만, 이를 통해 스프링부트와 타임리프 등 다양한 기능을 활용할 수 있는 기회를 얻었습니다. 앞으로의 프로젝트에서는 작은 부분부터 체계적으로 조율하며 진행하고자 하며, 이번 프로젝트 경험을 통해 여러가지 개발 환경을 배울 수 있어 즐거웠습니다.

정주은

실제로 데이터베이스와 연동되는 사이트를 만들고 기능을 추가하는 것이 재미있었고 무심코 이용하던 사이트들이 어떻게 작동하고 어떻게 만들어졌는지 조금이나마 이번 팀 프로젝트를 통해 경험할 수 있어서 재밌었습니다.





THANKS!

seoulmate팀

포트폴리오 소개를 마치겠습니다.

감사합니다.