

숙소호스팅 및 예약 서비스 (Heunheuntrip)

BootStrap으로 기반으로 만들어진 반응형 웹사이트입니다. JAVA Spring을 통해 서버를 구동하며 JSON을 이용해 클라이언트 사이드 렌더링으로 처리하고 있습니다. AWS EC2, S3 서버로 분리 하였고 JAVA, JavaScript, MariaDB, Git, Gradle의 환경에서 개발 하였습니다.

수행기간 : 2019-04 ~ 2019-06

프로젝트 구분 : 팀 프로젝트(5명) / 기여도 매우 높음

상용화 : 베타서비스

전담 역할 : Back-end, Server, Database

보조 역할 : Front-end

사용기술



특징

- 알림** : User process에 따른 SMS, Email
- Logging** : Login 시도 시 일시, 요청시간, 아이디, 성공/실패 여부의 Log
- Server** : Aws EC2 2개 서버와 S3 Storage 서버 분리

프로젝트 Github

<https://github.com/sonbyungjun/heunheuntrip>

시연영상

<https://www.youtube.com/embed/hBiv0t-dOVM>

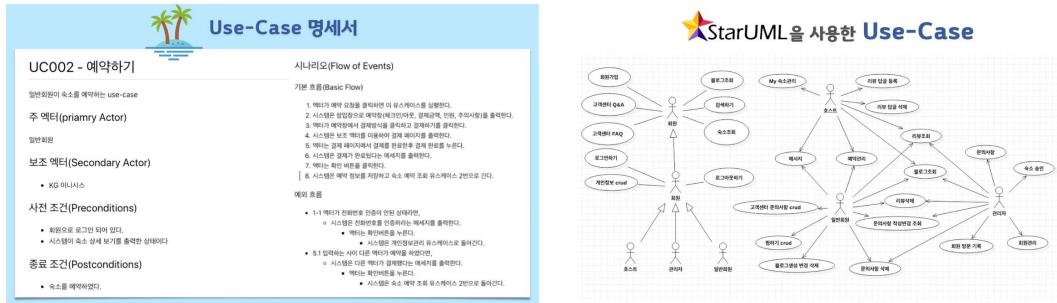
프로젝트 URL

<https://sbj.kr/heunheuntrip/html>

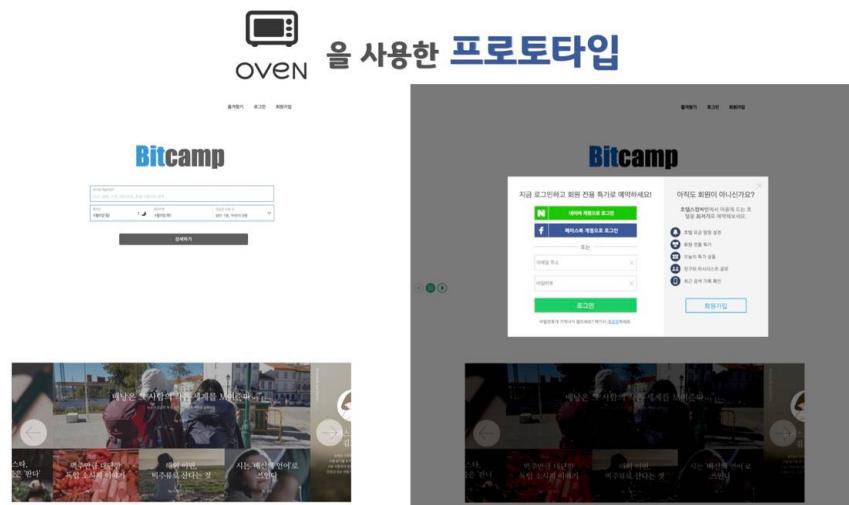
프로젝트 발표영상

<https://www.youtube.com/watch?v=G2yrWLZoANg>

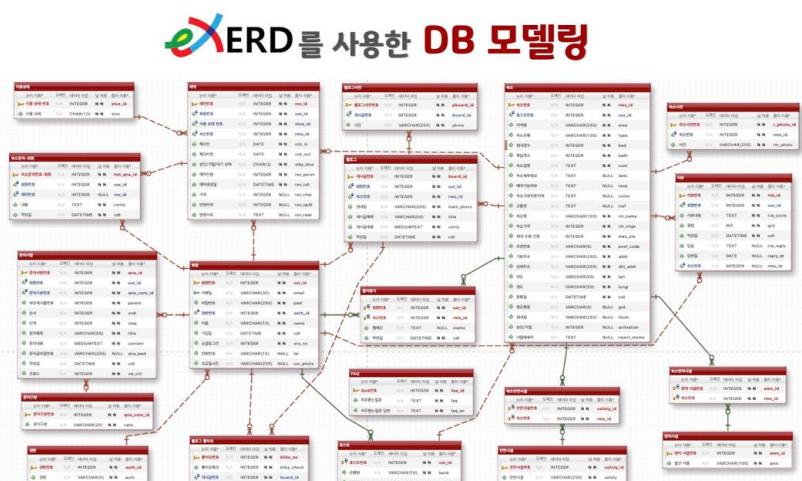
Use-Case



프로토타입



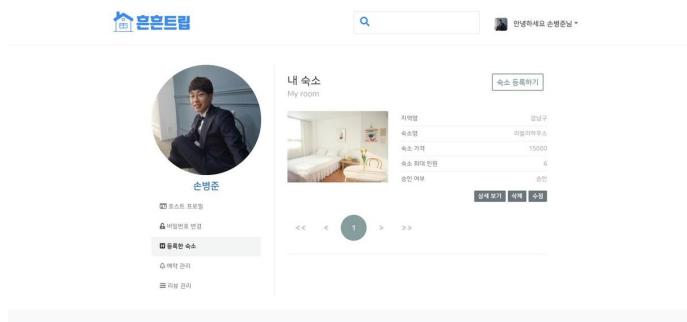
DB 모델링



맡은 기능

- 숙소 등록, 조회
- 숙소 검색
- 결제, 예약
- Log
- Aws Server 구축
- Spring Scheduled

1. 숙소 호스팅



사용자가 숙소를 등록할 수 있습니다.

관련 Tech Stack: Spring, JavaScript, Fullpage.js, Fileupload, Sweetalert2

역할: Front-end, Back-end

구현사항

1. fullpage.js 라이브러리를 가져와 단계별로 정보를 저장합니다.
2. 다음주소 api를 사용해 주소 정보를 조회 후 위도 경도를 가져오는 함수를 사용하여 DB에 저장
3. fileuploader 라이브러리를 사용하여 사진 등록
4. Ajax로 서버에서 JSON을 응답받고 Handlebars로 컴파일
5. 다음지도 api를 사용하여 지도와 맛집정보 명소정보를 표시
6. DateRangePicker를 사용하여 달력 구현

이슈

1. 서버에서 Ajax로 JSON 데이터를 가져와 html 태그를 만들어 추가하는데에는 javascript 코드로 생성하기 번거롭고 불편하다. handlebars를 사용해 JSON 데이터를 script 태그 안에 변수처럼 사용하여 컴파일 한 후 사용했습니다.
2. 다음주소 api를 사용하면 주소 정보를 가져오기 쉽다. 하지만 우리 프로젝트에서는 위도 경도가 필요 했다. 다음주소 api를 이용하여 가져온 주소로 다음지도 api의 위도 경도를 가져오는 함수를 호출하여 해결했다.
3. JSON 데이터를 클라이언트에서 서버로 보낼 때 java 라이브러리인 Jackson이 데이터 타입으로 자동 변환 시켜줘서 편리하다. 하지만 INT 형으로 변환할 때 문자타입이면 예외가 발생하는데 Controller 파라미터의 데이터 타입을 입력하고 그 뒤에 BindingResult 클래스를 쓰면 예외정보가 담기면서 무시된다. 예외를 출력하고 디버깅을 하여 해당 문제를 해결했다.

2. 숙소 검색

관련 Tech Stack: Spring, Mybatis, Ajax, JSON

역할: Front-end, Back-end

구현사항

- 지역명이나 역명으로 검색 시 다음지도api에 있는 함수를 호출하여 위도 경도를 찾고 그 기준으로 반경 5KM 에 있는 숙소를 찾는다.

이슈

- 검색어를 입력하고 검색을 누르면 일단 기준이 되는 위도 경도를 찾아야 했다. 다음주소 api를 써서 검색어에 대한 위치 정보가 나와서 편리하다. 하지만 위치 정보가 배열로 반환하기 때문에 기준을 찾기가 힘들었다. 배열에 있는 첫번째 정보가 검색한 결과에 위치가 가장 가깝다는것을 알게되어 그 위치의 위도 경도를 사용했다.

3. 결제 / 예약

관련 Tech Stack: Spring, JavaScript, I'mProt

역할: Front-end, Back-end

구현사항

- 클라이언트에서 숙소 번호와 체크인과 체크아웃 날짜를 서버에게 보내서 서버에서 생성한 주문번호와 금액을 클라이언트에게 보내줍니다.
- 클라이언트에서 결제정보로 I'mPort로 결제를 하게 되고 승인번호가 리턴되고 클라이언트에서 다시 서버로 승인번호를 보냅니다.
- 서버는 승인번호 받아서 I'mPort로 보내면 처음에 클라이언트가 보냈던 정보를 리턴하게 됩니다.

4. 리턴된 정보와 서버에서 생성한 정보를 비교하여 맞으면 DB에 정보가 저장되고 틀리면 결제취소가 이뤄집니다.
5. DB에 저장하는 과정에서 먼저 체크인 날짜를 가져와 해당 날짜가 예약되어있는지 비교한 다음에 추가가 됩니다.

이슈

1. DB에 예약정보를 저장하는 과정에서 체크인 날짜가 중복될 우려가 있었다. 쓰레드가 동시에 접근하여 Insert하게 되면 같은 날짜가 저장되게 된다. 여러가지 방법이 있었지만 synchronized를 사용하여 한 쓰레드만 접근하게 했다.

4. Log

번호	요청URL	생성일	아이피주소	이메일	여부	time
836	/json/auth/login	2019-06-29 01:49:58	123.142.176.234	a@naver.com	success	3
835	/json/auth/login	2019-06-29 00:31:55	123.142.176.234	byungjin2020@gmail.com	success	9
834	/json/auth/logout	2019-06-29 00:31:43	123.142.176.234	thien@naver.com	success	1
833	/json/auth/login	2019-06-29 00:31:31	123.142.176.234	thien@naver.com	success	3
832	/json/auth/logout	2019-06-29 00:31:25	123.142.176.234	a@naver.com	success	1
831	/json/auth/login	2019-06-29 00:14:46	123.142.176.234	a@naver.com	success	5
830	/json/auth/login	2019-06-28 23:59:57	123.142.176.234	kim@naver.com	success	17
829	/json/auth/login	2019-06-28 23:59:49	182.212.163.186	saint2020@naver.com	success	3
828	/json/auth/logout	2019-06-28 23:59:16	182.212.163.186	a@naver.com	success	1
827	/json/auth/login	2019-06-28 09:19:09	182.212.163.186	a@naver.com	success	6
826	/json/auth/logout	2019-06-28 09:18:36	182.212.163.186	a@naver.com	success	32
825	/json/auth/login	2019-06-28 09:18:15	182.212.163.186	a@naver.com	success	4
824	/json/auth/login	2019-06-28 03:09:38	123.142.176.234	a@naver.com	success	11
823	/json/auth/login	2019-06-28 03:08:21	127.0.0.1	thien@naver.com	success	14
822	/json/auth/login	2019-06-27 02:57:21	127.0.0.1	a@naver.com	success	14
821	/json/auth/logout	2019-06-27 02:57:11	127.0.0.1	saint2020@naver.com	success	1
820	/json/auth/login	2019-06-27 02:46:25	127.0.0.1	thien145@naver.com	success	83
819	/json/auth/logout	2019-06-27 02:45:44	127.0.0.1	byungjin2020@gmail.com	success	1
818	/json/auth/login	2019-06-27 02:45:36	127.0.0.1	saint2020@naver.com	success	12
817	/json/auth/logout	2019-06-27 02:45:29	127.0.0.1	a@naver.com	success	1
816	/json/auth/login	2019-06-27 02:14:48	127.0.0.1	a@naver.com	success	13

관련 Tech Stack: Spring

역할: Back-end

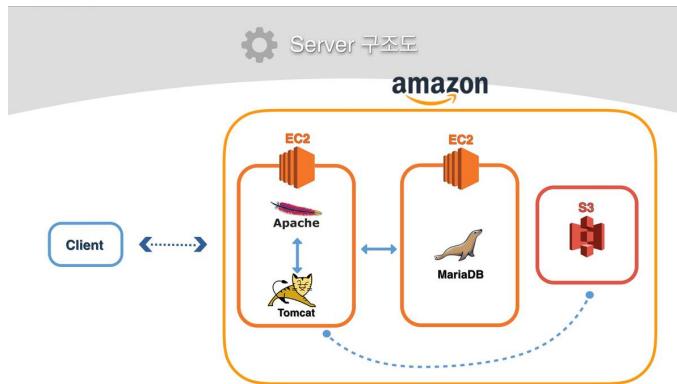
구현사항

1. Spring Interceptor를 이용해 해당 Login controller가 실행되기 전과 후에 실행되도록 설정하고 로그인 성공/실패, ip, 지역시간, 요청시간, 요청URL, 아이디를 생성하여 DB에 저장하도록 했습니다.

이슈

1. 로그인 시도 한 이메일 정보와 성공 실패 여부를 가져오려면 session에 담아서 사용하면 편리하다. 하지만 그 정보는 한번만 쓰고 한번의 요청에서만 필요하기 때문에 request에 담아서 해결했다.

5. AWS Server 구성



관련 Tech Stack: AWS

역할: Back-end

구현사항

1. AWS EC2 micro 인스턴스에 MariaDB를 설치하여 연동했습니다.
2. AWS EC2 small 인스턴스에 Apache와 Tomcat을 설치하여 연동했습니다.
3. AWS S3의 Java SDK를 사용하여 이미지 관리.

이슈

1. 프로젝트 제작 기간 중 팀원들끼리 DB를 공유하지 않았기 때문에 불편함이 있었습니다. AWS에 DB를 공유하게끔 하여 제작에 편리하게 했고 향후 DB를 분리함으로써 메모리 사용을 최적화 했습니다.
2. S3 Storage가 파일 관리를 담당하여 편리하다. 하지만 Credentials 파일을 공유해야 하기 때문에 불편했지만 Trello를 사용하여 파일을 공유함으로써 해결했다.

5. Spring Scheduled



관련 Tech Stack: Spring

역할: Back-end

구현사항

1. 매일 오전 6시에 S3의 있는 데이터와 DB에 있는 데이터를 비교하여 쓰레기 데이터 삭제.
2. 매일 오전 10시에 예약 테이블 DB에 있는 오늘 날짜인 체크인 데이터를 가져와서 상태 column을 체크인으로 변경
3. 매일 오후 12시에 예약 테이블 DB에 있는 오늘 날짜인 체크아웃 데이터를 가져와서 상태 column을 체크아웃으로 변경

이슈

1. S3 Storage를 쓰면 파일 관리를 담당해서 편리하지만 파일을 저장할 때 예외가 발생할 수도 있다. 그렇게 되면 DB에는 저장이 되지 않는 데이터가 생겨버리는데 이를 쓰레기 데이터라 한다. 쓰레기 데이터가 많이 생기면 정리 해줘야 하는데 Scheduled를 사용하여 자동화 시켰다.