Where is my Home

김근우 이우철

House On Budongsan

목차

1. 요구사항 정의서

2. 테이블 구조도

3. 클래스 다이어그램

4. 화면 설계서

1. 요구사항 정의서

WhereIsMyHome의 요구사항은 크게 4가지로 나뉜다

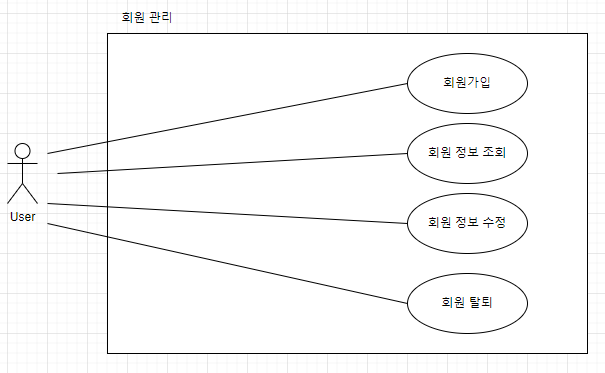
1. 회원관리

2. 게시판

3. 부동산 검색

4. 부동산 뉴스 관리

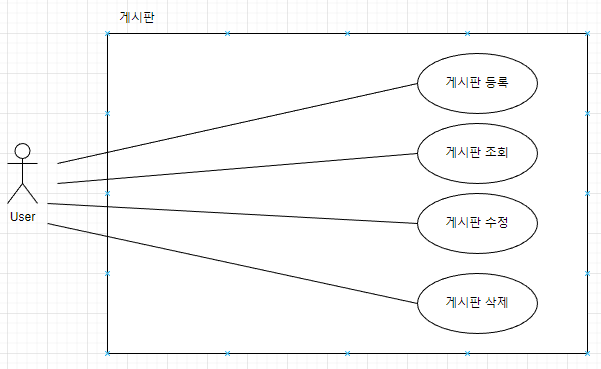
* 1. 회원관리



`회원관리`에서의 use case는 크게 위와 같다.

사용자 정보를 생성하기 위한 회원가입, 이에 대한 정보를 조회할 수 있는 기능, 패스워드 및 닉네임과 같은 정보를 수정할 수 있는 기능, 또한 탈퇴 기능이 필요하다.

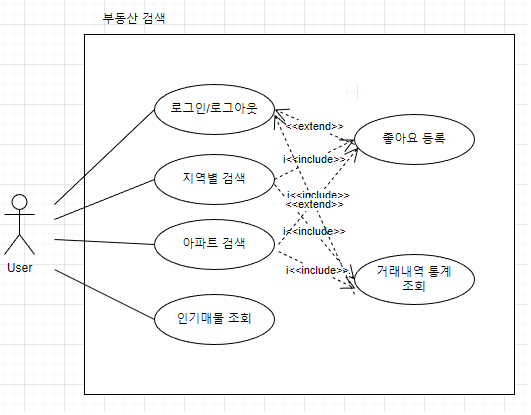
* 1. 게시판



`게시판`에서의 use case는 크게 위와 같다

제목과 내용으로 구성된 하나의 글을 게시하는 게시판 등록 기능, 본인 및 타인이 작성한 게시글에 대해 접근하여 내용을 확인 할 수 있는 게시판 조회 기능, 본인이 작성한 게시글에 대해 수정할 수 있는 게시판 수정 기능, 또한 본인이 작성한 글에 대해 삭제할 수 있는 게시판 삭제 기능이 필요하다.

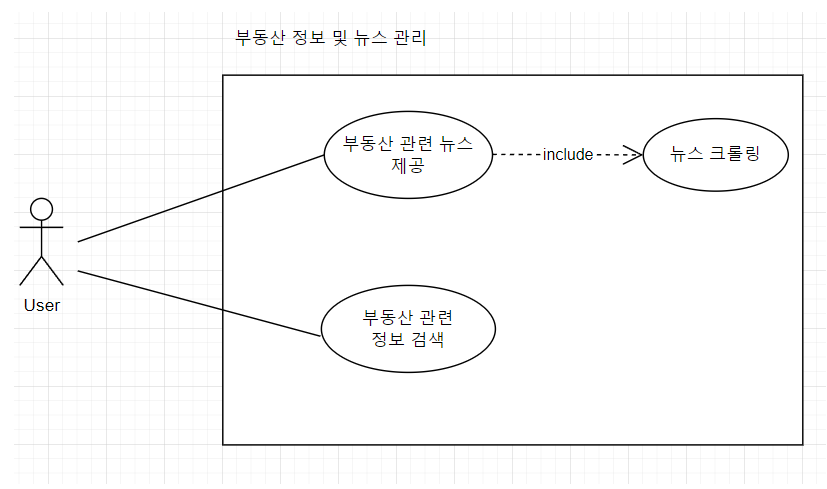
* 1. 부동산 검색



`부동산 검색`에서의 use case는 크게 위와 같다

좋아요와 같은 행위를 하기 위해 로그인 기능이 필요하며 이에 대한 로그아웃 기능 또한 필요하다, 검색 기능은 크게 지역별 검색과 아파트별 검색을 지원해야한다, 또한 각각의 장소별 및 아파트별 인기매물을 조회할 수 있는 기능이 필요하다.

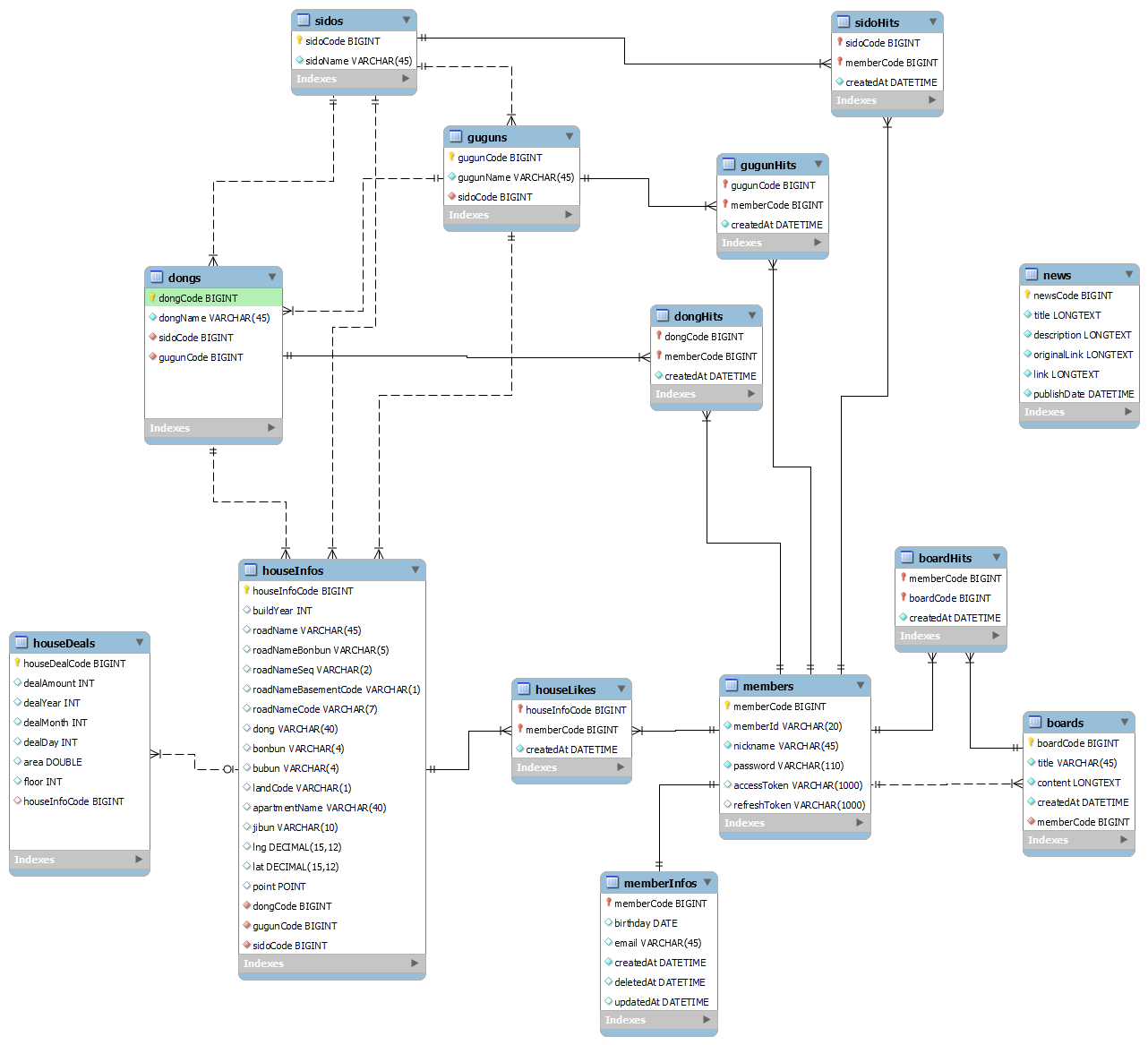
* 1. 부동산 뉴스 관리



`부동산 뉴스 관리` use case는 크게 위와 같다.

부동산 관련 뉴스를 사용자에게 제공하기 위하여 뉴스 크롤링 기능이 필요하다, 또한 이에 대하여 사용자가 접근할 수 있도록 부동산 관련 정보를 검색할 수 있도록 한다.

2. 테이블 구조도



데이터 베이스의 테이블 구조는 위의 그림과 같이 구성하였다.

크게 4가지로 나눌 수 있다.

1. 장소 정보  
2. 부동산 정보  
3. 회원 정보  
4. 게시판 정보  
 이 외에 회원과 부동산간의 관계, 회원과 장소간의 관계와 같이 관계 테이블 또한 구조에 포함된다

* 1. 장소 정보

|  |  |  |  |
| --- | --- | --- | --- |
| **테이블명** | **필드** | **타입** | **설명** |
| **sidos** | sidoCode | Long | 시도 식별자 |
| sidoName | String | 시도 이름 |
| **guguns** | gugunCode | Long | 구군 식별자 |
| gugunName | String | 구군 이름 |
| sidoCode | Long | 속한 시도 |
| **dongs** | dongCode | Long | 동 식별자 |
| dongName | String | 동 이름 |
| sidoCode | Long | 속한 시도 |
| gugunCode | Long | 속한 구군 |

장소 정보관련 테이블은 3개로 sidos, guguns, dongs이다.

각각의 테이블들은 고유한 식별자를 가지며 해당 장소가 속한 상위 장소에 대해 참조할 수 있는 키를 가진다.

* 1. 부동산 정보

|  |  |  |  |
| --- | --- | --- | --- |
| **테이블명** | **필드** | **타입** | **설명** |
| **houseInfos** | houseInfoCode | Long | 건물 식별자 |
| buildYear | Integer | 건축 년도 |
| apartmentName | String | 건물 이름 |
| ... | | |
| **houseDeals** | houseDealCode | Long | 거래 식별자 |
| dealAmount | Integeer | 거래 금액 |
| dealYear | Integer | 거래 년도 |
| area | Double | 면적 |
| houseInfoCode | Long | 건물 정보 참조키 |
| ... | | |

부동산 관련 테이블은 2개로 houseInfos, houseDeals이다

houseInfos는 건물에 대한 정보들을 나타내며 건물이 위치한 시, 도, 구군, 동과 같은 정보를 가지고 있으며 위, 경도를 통한 위치 및 공간 인덱스를 위한 point를 가진다. 이를 통해 추후 검색 시 빠른 검색을 지원한다.

houseDeals는 해당 건물에 대해 매매된 내역을 나타내며, 각각의 거래마다 거래 금액, 거래 일자, 층수, 면적과 같은 요소를 가진다. 해당 테이블은 houseInfos에 연관된 테이블이다.

* 1. 회원 정보

|  |  |  |  |
| --- | --- | --- | --- |
| **테이블명** | **필드** | **타입** | **설명** |
| **members** | memberCode | Long | 멤버 식별자 |
| memberId | String | 아이디 |
| nickname | String | 별칭 |
| password | String | 암호 |
| accessToken | String | JWT 토큰 |
| refreshToken | String | JWT 토큰 |
| **memberInfos** | memberCode | Long | 멤버 식별자 |
| birthday | DateTime | 생년월일 |
| email | String | 이메일 |
| createdAt | DateTime | 가입날짜 |
| deletedAt | DateTime | 탈퇴날짜 |
| updatedAt | DateTime | 수정날짜 |

회원관련 테이블은 2개로 members, memberInfos이다

Members는 회원에 대한 아이디, 패스워드, 별칭, JWT 토큰과 같은 요소로 구성된다. 이에 따라 로그인과 같은 행위를 할때 사용하게 된다.

memberInfos는 회원에 대해 추가적인 정보로 생년월일과 이메일과 같이 비교적 자주 사용되지 않은 정보들을 모아둔 테이블이다. 이에 따라 members와 1:1의 관계를 가지게 된다.

* 1. 게시판 정보

|  |  |  |  |
| --- | --- | --- | --- |
| **테이블명** | **필드** | **타입** | **설명** |
| **boards** | boardCode | Long | 게시글 식별자 |
| title | String | 제목 |
| content | String | 내용 |
| createdAt | DateTime | 게시 일자 |
| memberCode | Long | 작성자 참조 |
| **news** | newsCode | Long | 뉴스 식별자 |
| title | String | 제목 |
| description | String | 설명 |
| originalLink | String | 원본 기사 링크 |
| link | String | 기사 링크 |
| publishDate | DateTime | 게시 일자 |

게시판 관련 테이블은 boards, news이다.

boards는 사용자가 작성하는 게시글에 대한 정보를 저장하기 위한 테이블로 제목, 내용, 게시 일자, 작성한 멤버의 코드로 이루어진다.

news는 네이버 검색 API를 이용하여 크롤링링된 데이터를 저장하기 위한 테이블로 제목, 내용, 링크, 기사의 게시 일자로 이루어진다.

* 1. 관계 테이블

관계 테이블은 다음과 같이 3개로 구성된다  
1. 장소 조회  
2. 건물 좋아요  
3. 게시판 조회

**2-5-1. 장소 조회**

장소 조회는 사용자가 시도 또는 구군 또는 동과 같은 범위를 통해 부동산 매물에 대해 검색을 수행할 때 사용자가 검색한 기록을 저장하는 테이블로 각각의 사용자가 어떠한 장소를 검색했는지 저장한다. 이를 통해 최근 2주간 인기 지역과 같은 요소를 선정할 수 있다.

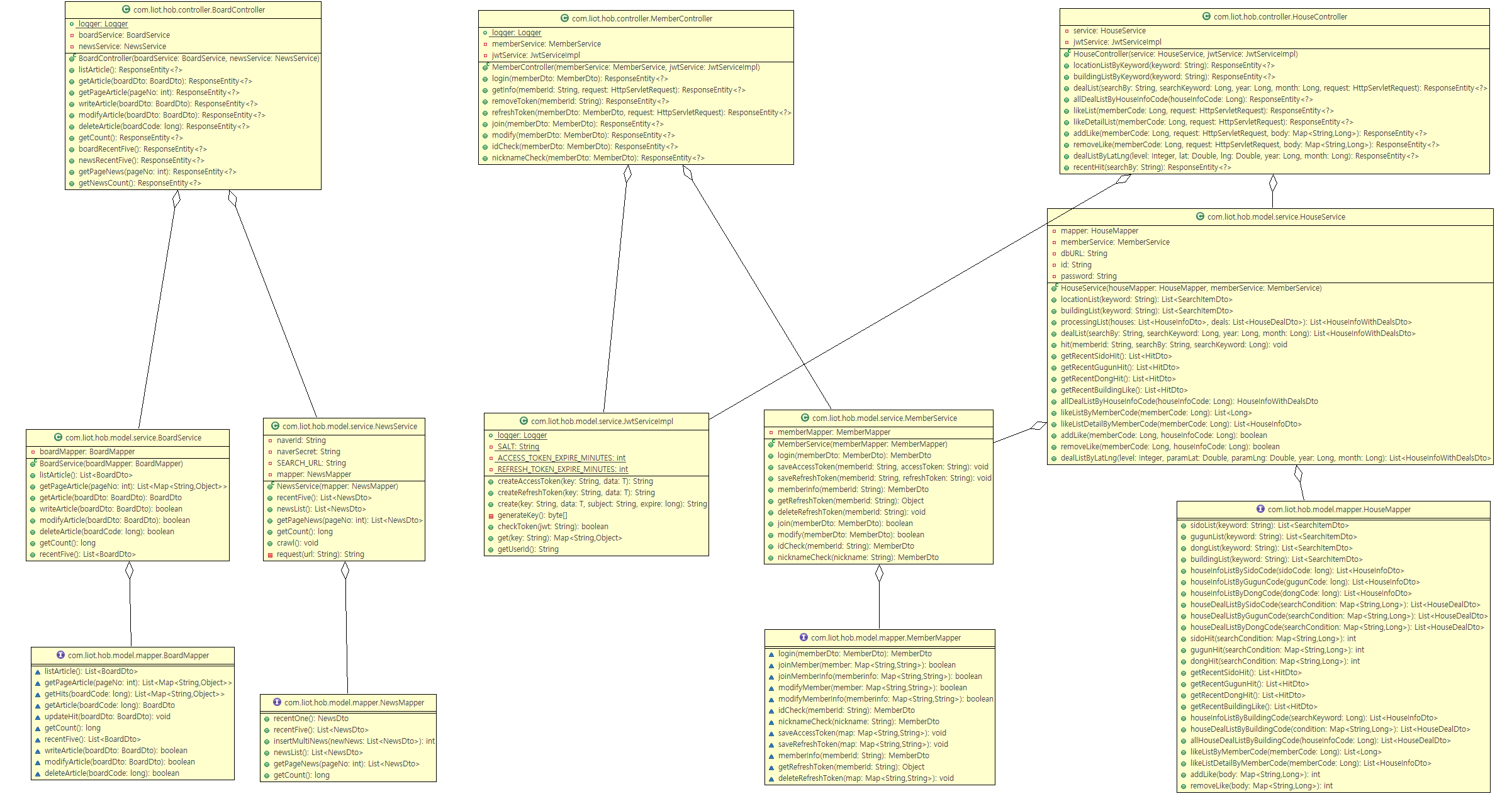
**2-5-2. 건물 좋아요**

각각의 사용자들은 건물에 대한 좋아요를 눌러 추후 좋아요 누른 건물만을 따로 볼 수 있도록 한다. 최근 2주간 좋아요가 많은 순으로 랭킹을 매겨 인기 매물을 선정할 수 있다.

**2-5-3. 게시판 조회**

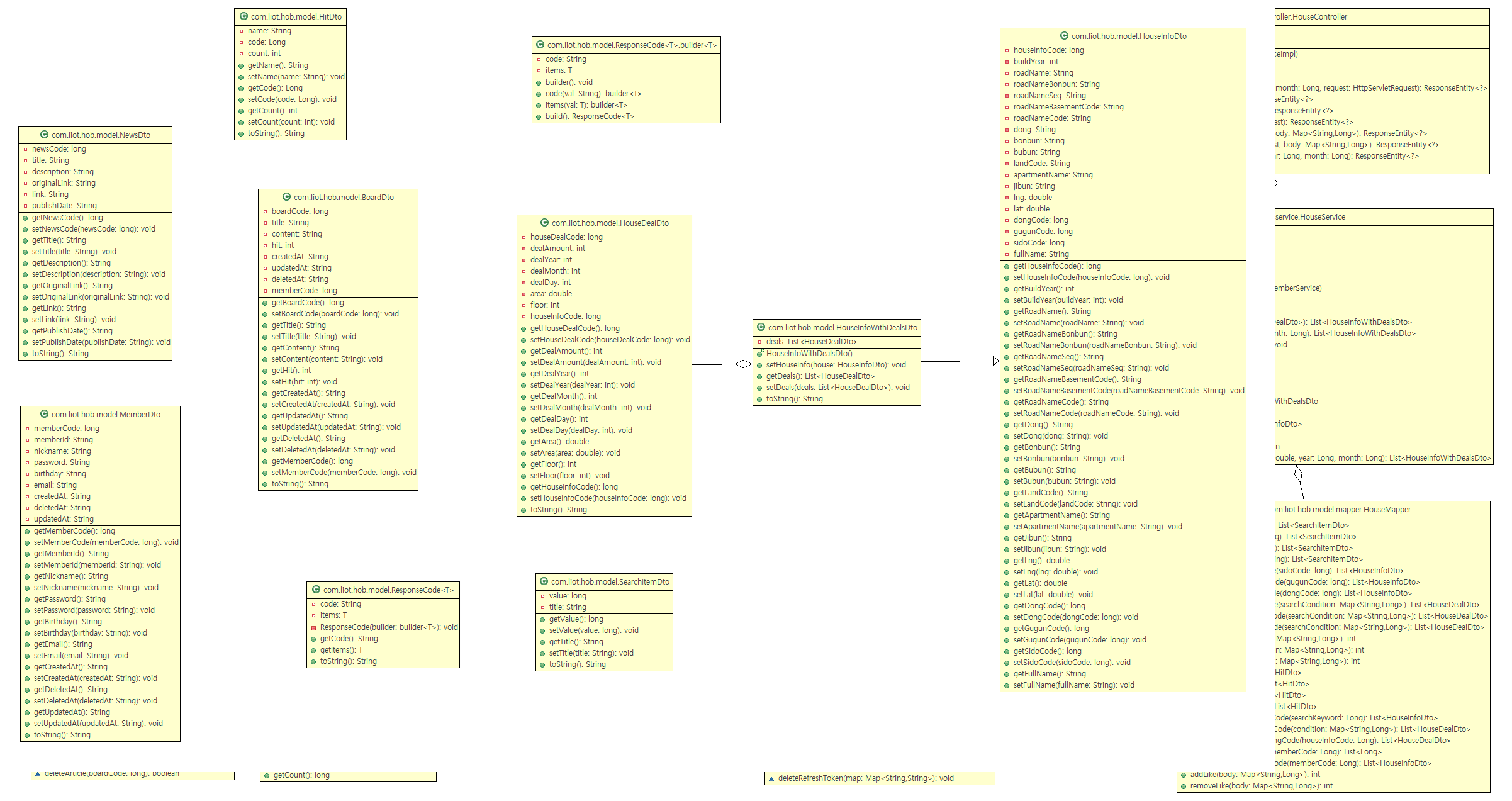
각각의 사용자들은 본인의 게시글 및 타인이 작성한 게시글을 조회할 수 있다. 이때 각각의 게시글 마다 조회한 정보를 기록하여 조회수를 표시할 수 있다.

3. 클래스 다이어그램



MVC패턴에서의 클래스 다이어그램은 위와 같이 구성된다.

크게 `board`, `member`, `house`의 3개의 도메인으로 나뉘며 각각의 도메인들은 controller - service - mapper의 레이어를 가진다. JWT를 다루기 위한 service와 news정보 관련 크롤링 및 조회 기능을 위한 news service를 추가적으로 가진다.



각 레이어간의 데이터 전달 및 REST에서의 response데이터를 반환하기 위한 DTO들은 위와 같이 정의하였다.

제 2절에서 설명한 테이블 구조도의 테이블들을 java의 클래스로 표현한 것들과 REST에서의 상태값과 같은 것들을 가지기 위한 ResponseCode클래스 및 거래정보를 표현한 HouseInfoWithDealDto와 같은 클래스들을 추가적으로 가지게 된다.

4. 화면 설계서

4-1. 메인페이지



메인페이지는 위와 같은 화면을 가지며 최상단의 헤더 그 바로 밑에 부동산 검색, 중단에 뉴스 및 자유게시판, 최하단에 실시간 지역 및 건물 랭킹을 나타낸다.

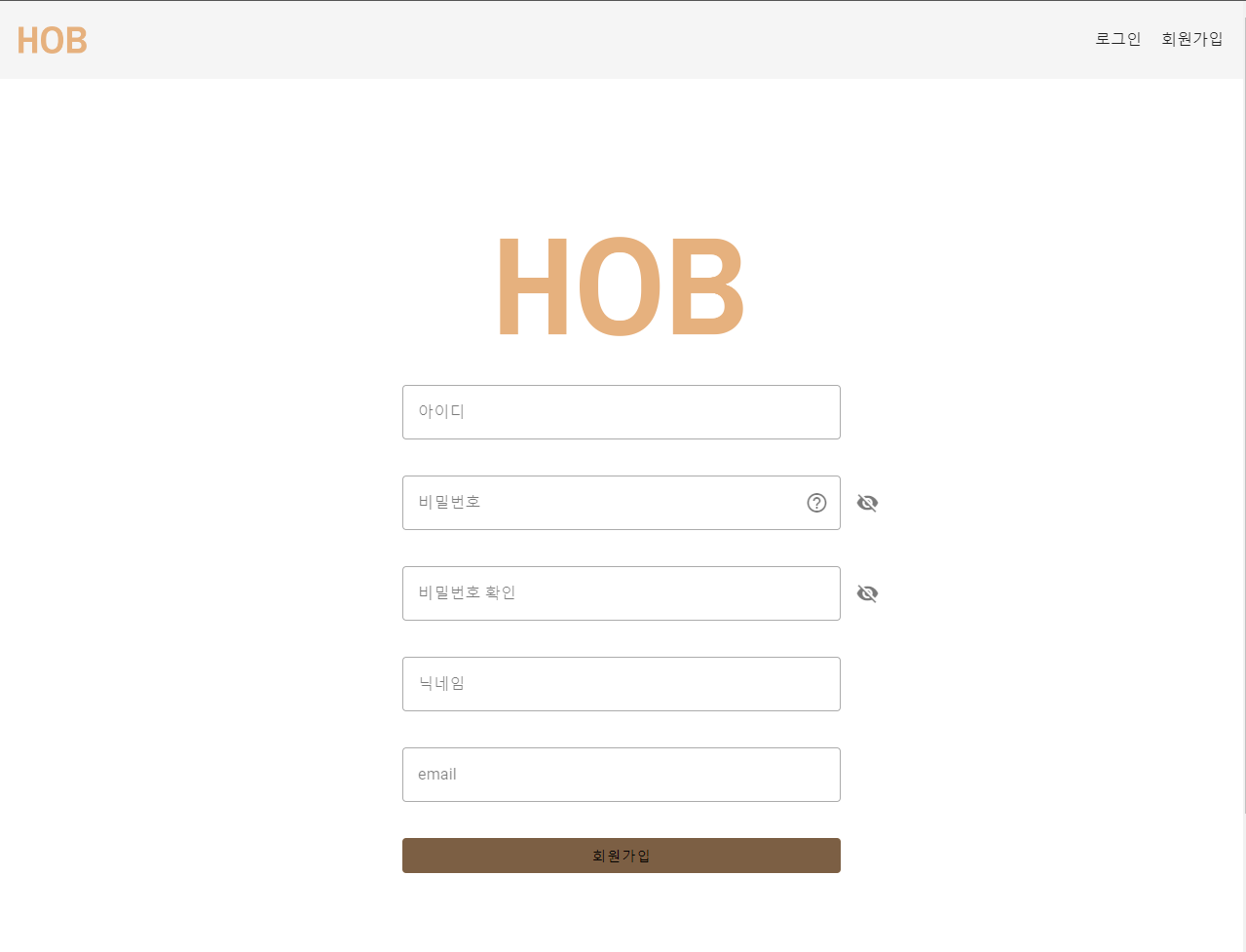
헤더의 좌측의 로고를 통해 홈을 바로 갈 수 있으며 로그인이 되어 있는 경우 우측의 프로필을 눌러 마이페이지로 이동 및 로그아웃을 수행할 수 있다. 만약 로그인이 되어있지 않은 경우 로그인과 회원가입으로 이동할 수 있는 버튼이 나타난다.

부동산 검색에서는 지역별 또는 건물명으로 검색할 수 있다. 이때 검색어가 변경됨에 따라 자동완성의 항목을 AJAX통신을 통해 가져온다. 이때 debounce를 적용하여 과도한 통신이 발생하지 않도록 하였다.

지역별 검색에서는 시 군구 동과 같은 범위로 검색을 하여 거래내역을 조회할 수 있다.

건물명 검색에서는 매물의 건물명을 조회하여 해당하는 건물에 대해서 거래내역을 조회한다.

4-2. 회원가입



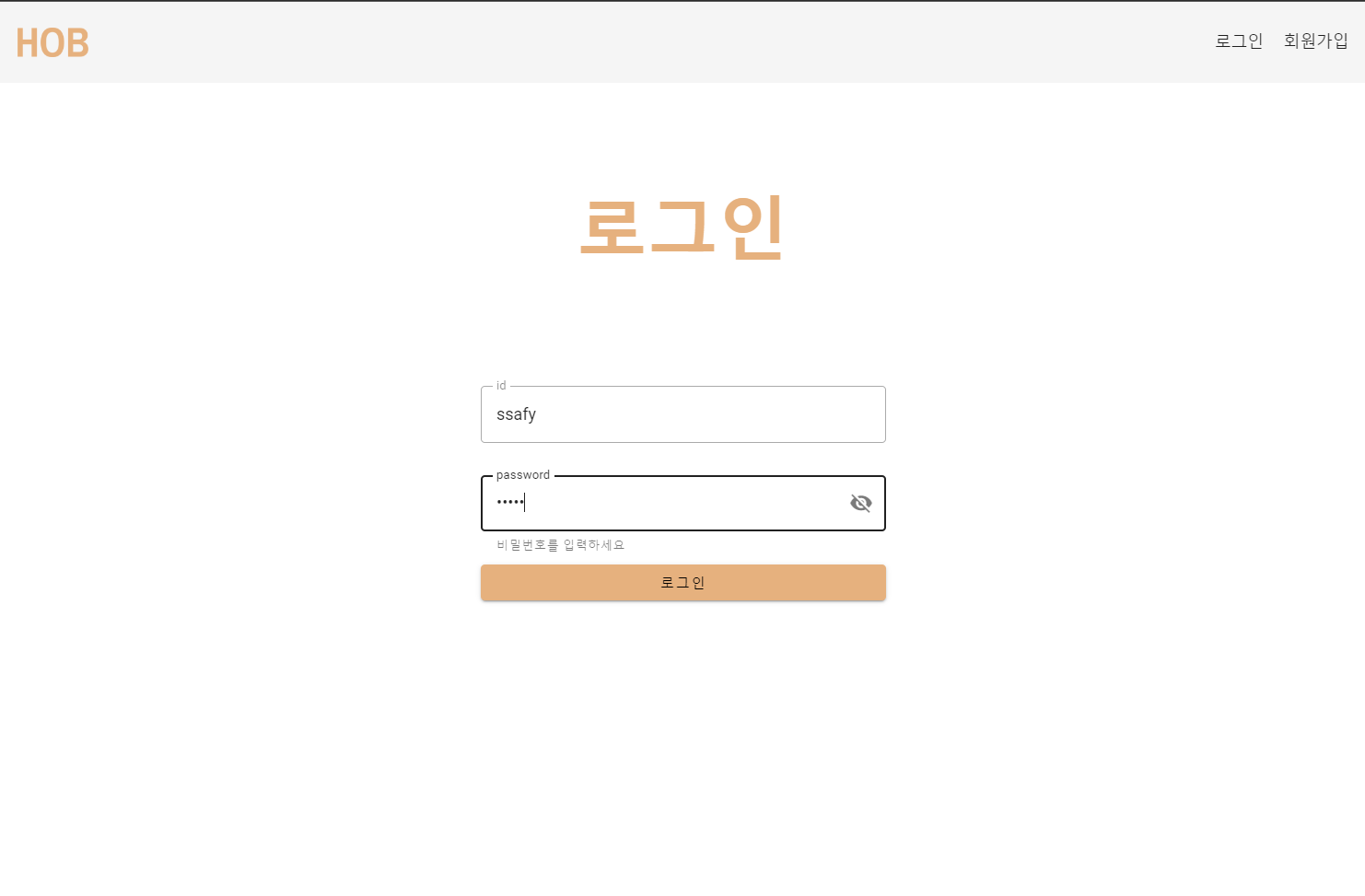
회원가입페이지는 위와 같이 아이디, 비밀번호, 비밀번호 확인, 닉네임, 이메일을 통해 가입할 수 있도록 구성한다.

아이디를 입력할 경우 AJAX통신을 통해 실시간으로 입력에 따라 아이디 중복 여부를 체크하여 이에 대해 중복 여부를 바로바로 나타낼 수 있도록 한다. 이때 과도한 API요청이 발생하지 않도록 debounce를 적용하였다.

비밀번호와 비밀번호 확인이 일치하지 않을 경우 이에 대해 경고 메시지를 나타낼 수 있도록 하였다.

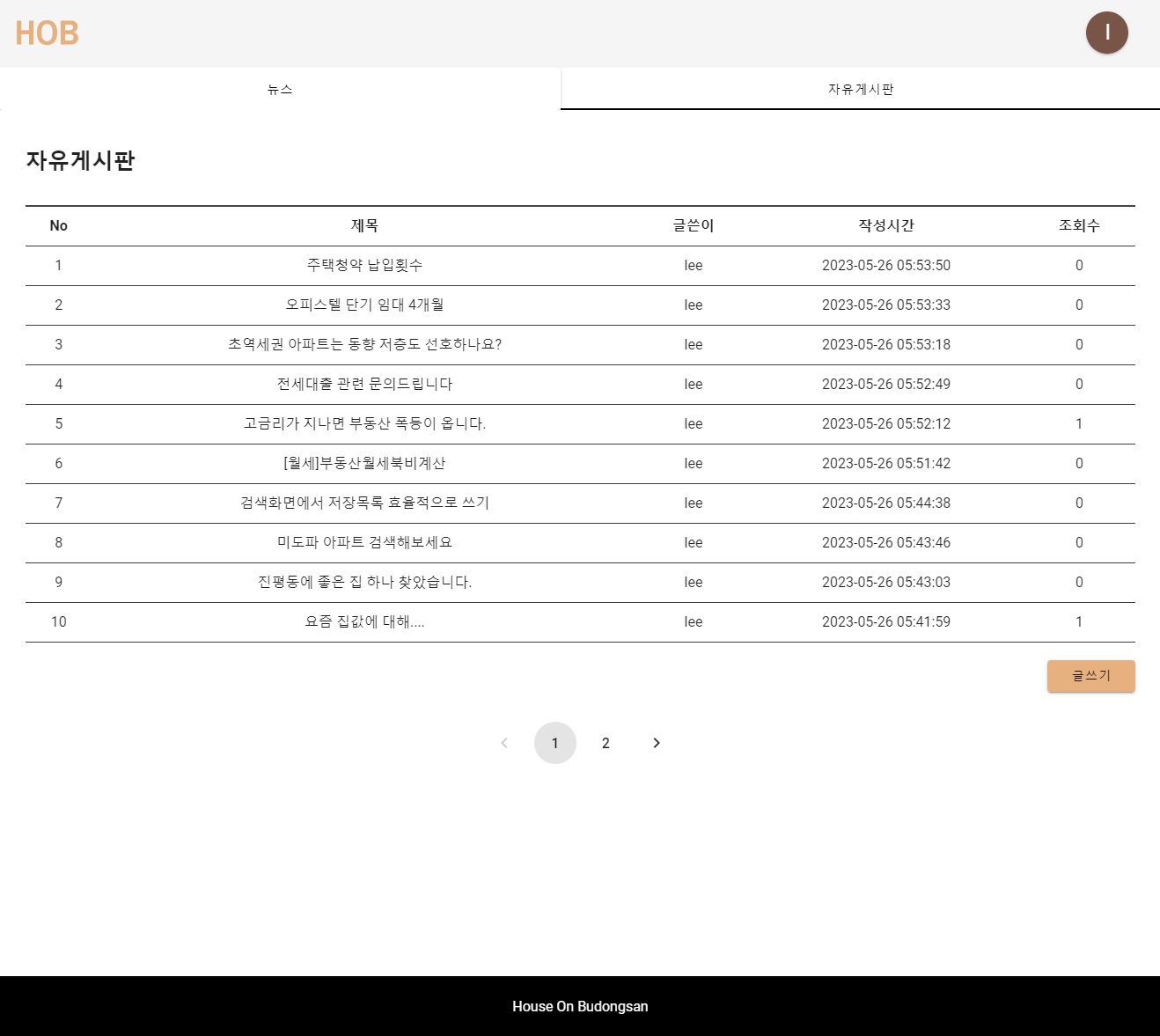
추가적으로 모든 입력란을 제대로 입력하여야 회원가입 버튼이 활성화되도록하였다.

4-3. 로그인



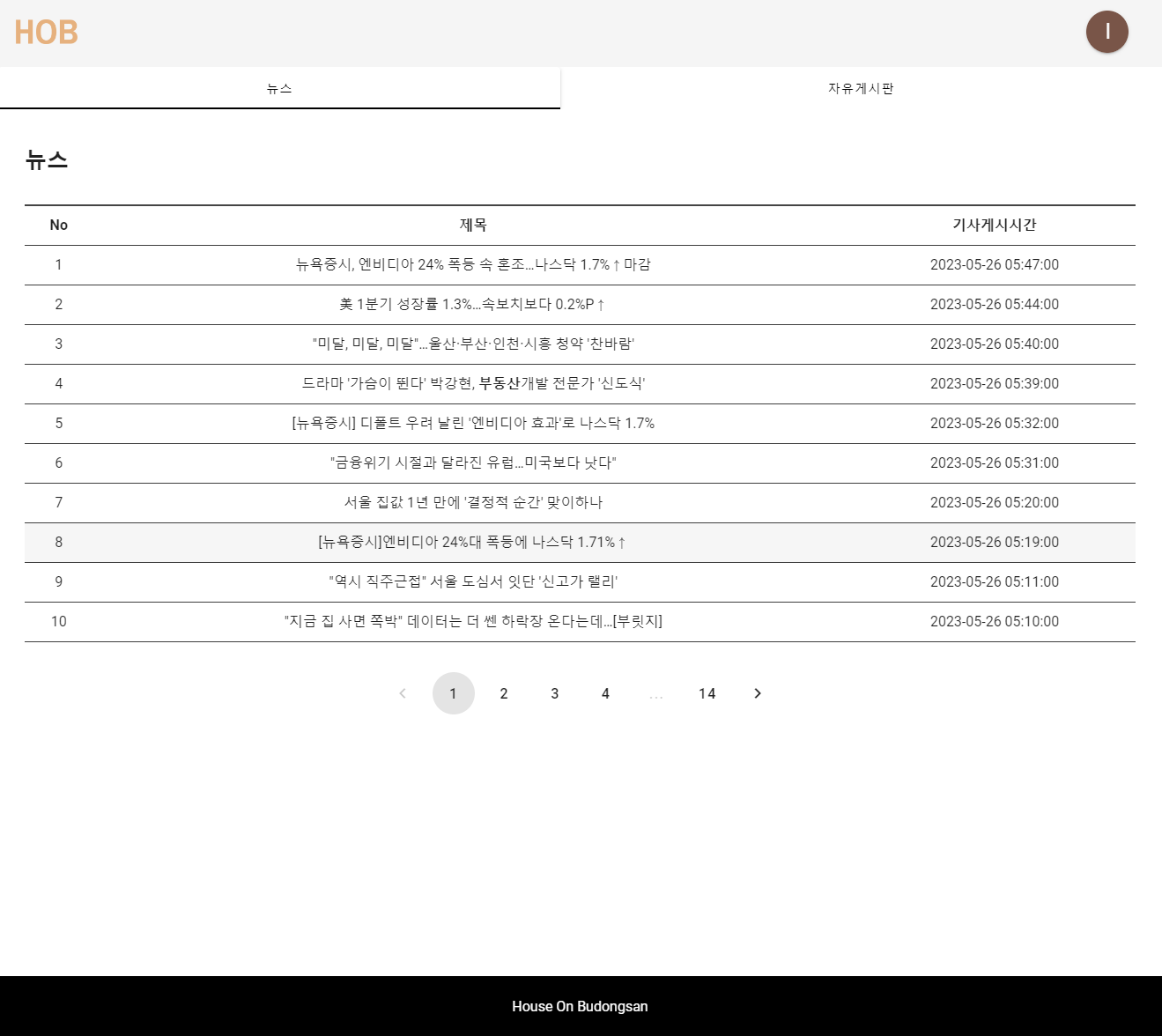
로그인페이지는 `4-2. 회원가입`에서 기입된 정보를 기반으로 사용자가 로그인을 수행할 수 있다. 이때 로그인 완료 시 AccessToken을 이용하여 각각의 사용자에 대해 로그인 유지 및 사용자 로그인 검증을 수행할 수 있도록 한다. 아이디 또는 비밀번호가 잘못 되었을 경우 이에 대한 경고창을 띄울 수 있도록 하였다.

4-4. 게시판



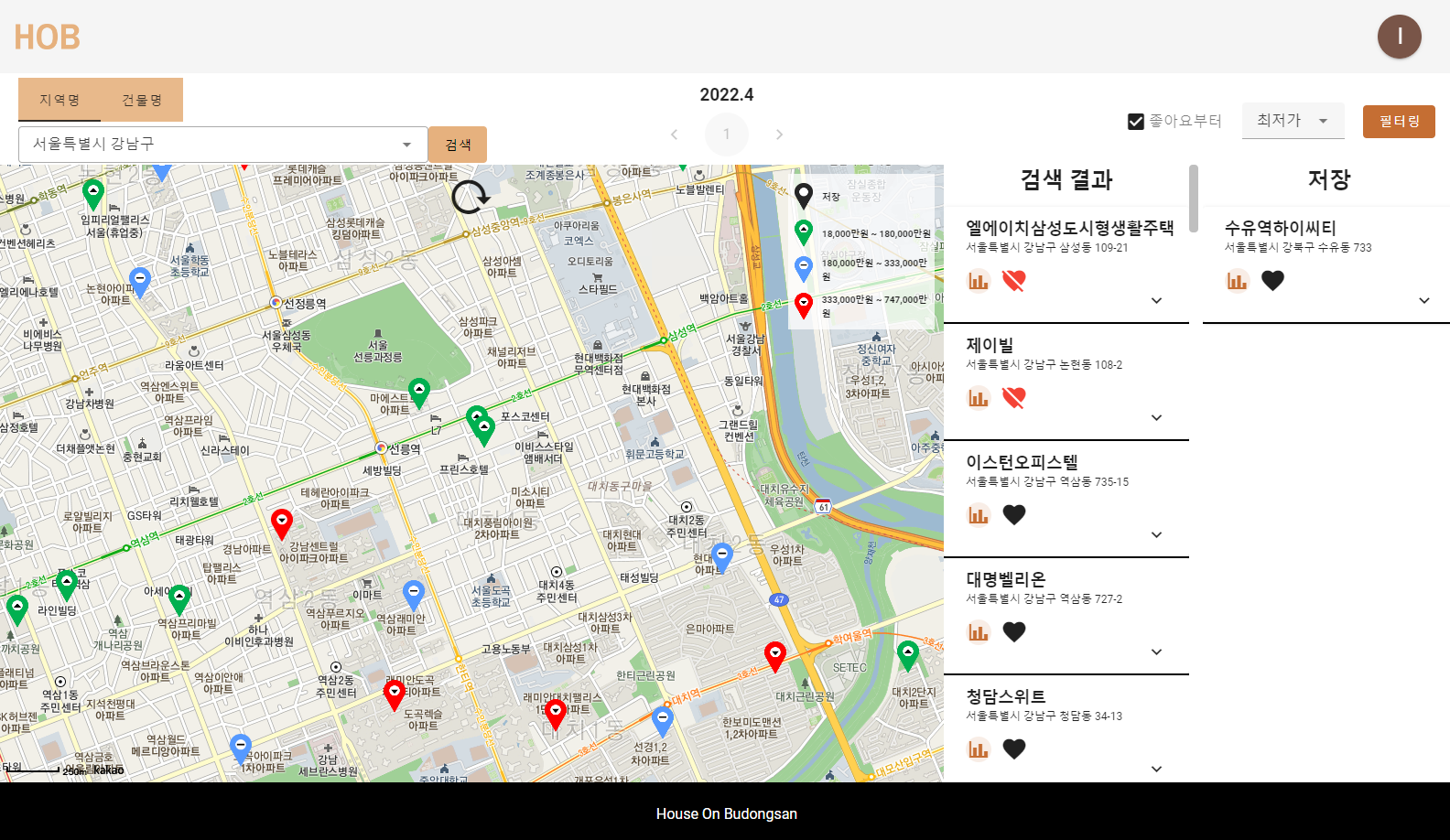
게시판페이지는 제목, 글쓴이, 작성시간, 조회수로 이루어지며 각각의 작성된 게시글을 눌러 상세히 볼 수 있도록 구성하였다. 만약 본인이 작성한 게시글일 경우 게시글에 대해 수정할 수 있도록 하였으며, 밑의 `글쓰기`버튼을 통해 로그인된 사용자인 경우 게시글을 작성할 수 있도록 하였다. 또한 Pagination을 통해 과도한 데이터 전송을 막았으며 사용자의 편의성을 늘렸다.

4-5. 뉴스



뉴스페이지는 네이버 검색 API를 활용하여 5분마다 자동적으로 `부동산`이라는 키워드가 들어간 뉴스들을 크롤링 하여 저장한 데이터를 기반으로 뉴스정보를 나타낸다. 각각의 뉴스들은 클릭시 원본 기사로 이동하도록 구성하였다.

4-6. 지역검색



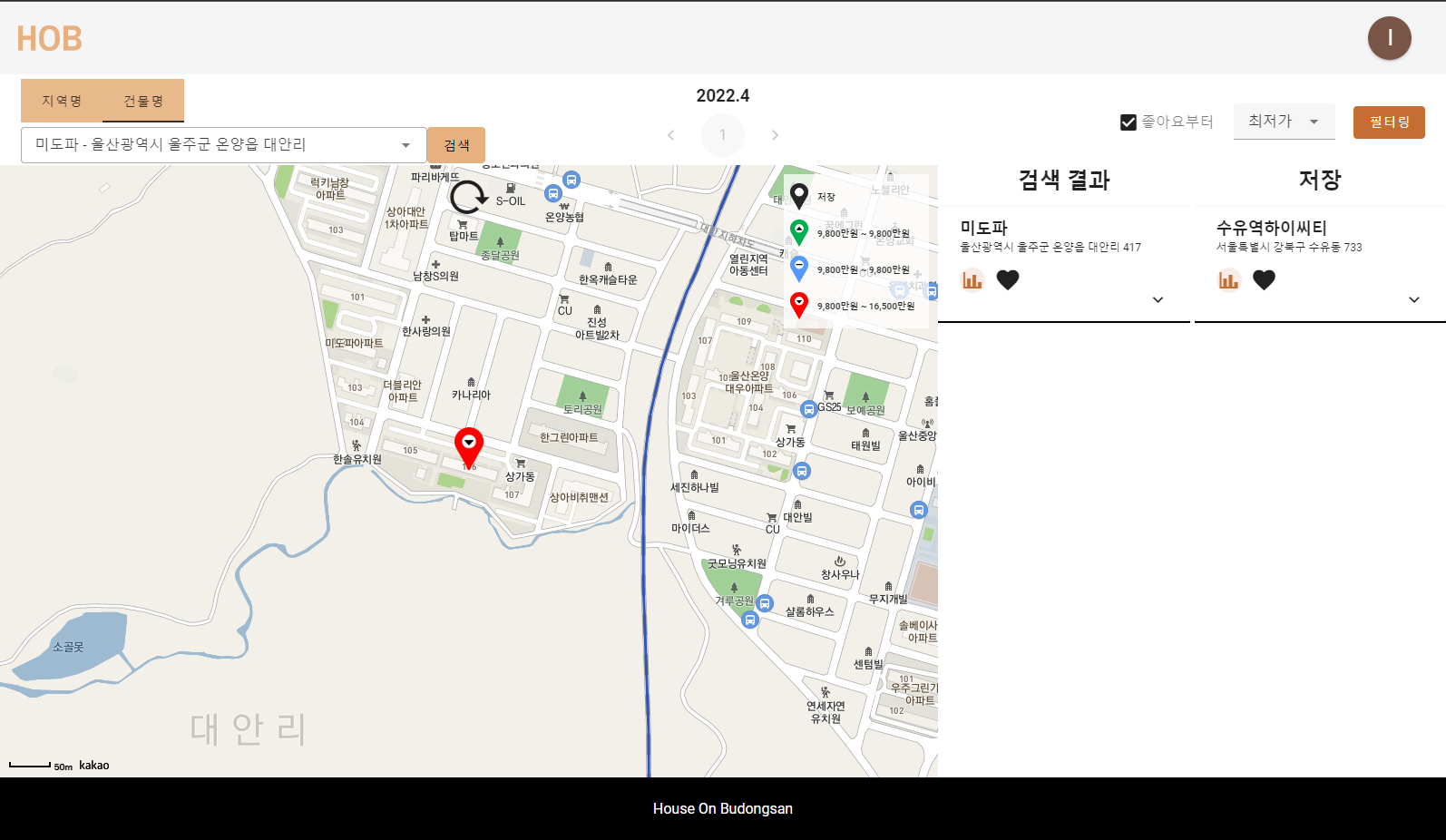
지역검색페이지는 4-1. 메인페이지에서 지역별 검색을 수행하였을 때 이동된다. 이에 따라 상단에는 좌측에는 AJAX통신을 통해 다시 검색 할 수 있는 입력폼, 중간에 데이터가 많을 경우 과도한 렌더링이 발생하므로 Pagination을 적용한 부분, 우측에 정렬 및 필터링을 할 수 있도록 구성하였다.

밑의 결과창에는 좌측에는 카카오맵을 통해 각각의 매물에 대한 위치와 검은색, 초록색, 하늘색, 빨간색 마커를 통해 사용자가 저장한 정보 및 정렬한 기준에 얼마나 부합하는지 정도를 나타내고 있다. 우측에는 두 개의 테이블로 구성되며 첫번째 테이블은 검색된 결과에 대해 정렬된 순서로 나타내며, 두번째 테이블은 사용자가 저장해 놓은 거래에 대해 나타내고 있다.

각각의 테이블은 통계 보기 버튼, 좋아요 및 좋아요 취소 버튼을 가지며 테이블의 요소들을 클릭할 경우 펼쳐지며 상세한 거래 정보를 나타낸다.

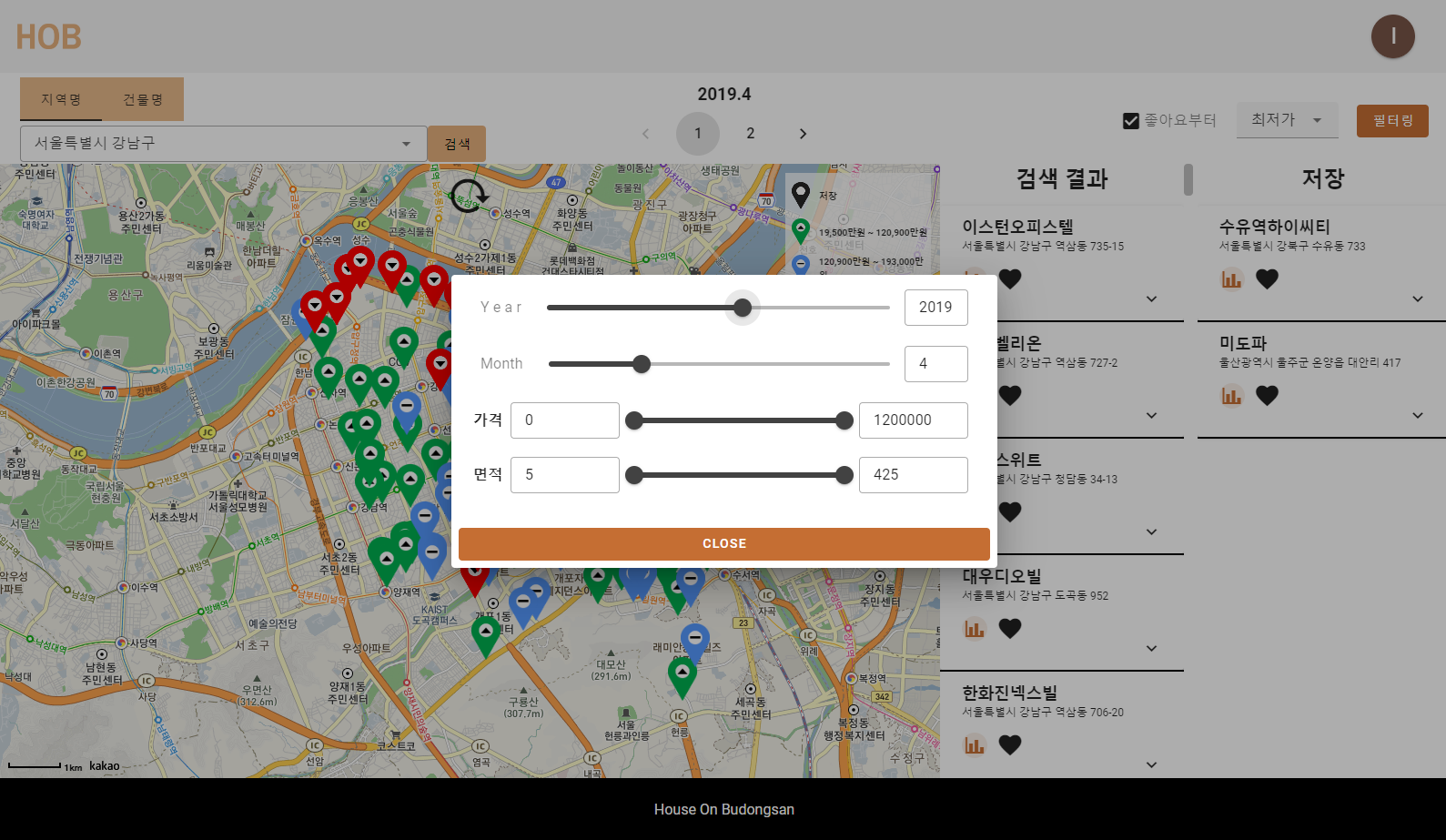
또한 맵 상단의 새로고침 버튼을 통해 현재 보고 있는 위치에서 매물 검색을 수행할 수 있다. 이때 현재 맵을 보고 있는 레벨에 따라 검색 범위가 바뀌며, 공간 인덱스를 통해 비교적 빠른 검색을 지원한다.

4-7. 건물검색



건물별 검색은 지역별 검색과 다르게 하나의 건물에 대해 수행하는 것으로 건물명을 통해 검색된 결과 1개를 나타내고 있는 것을 볼 수 있다.

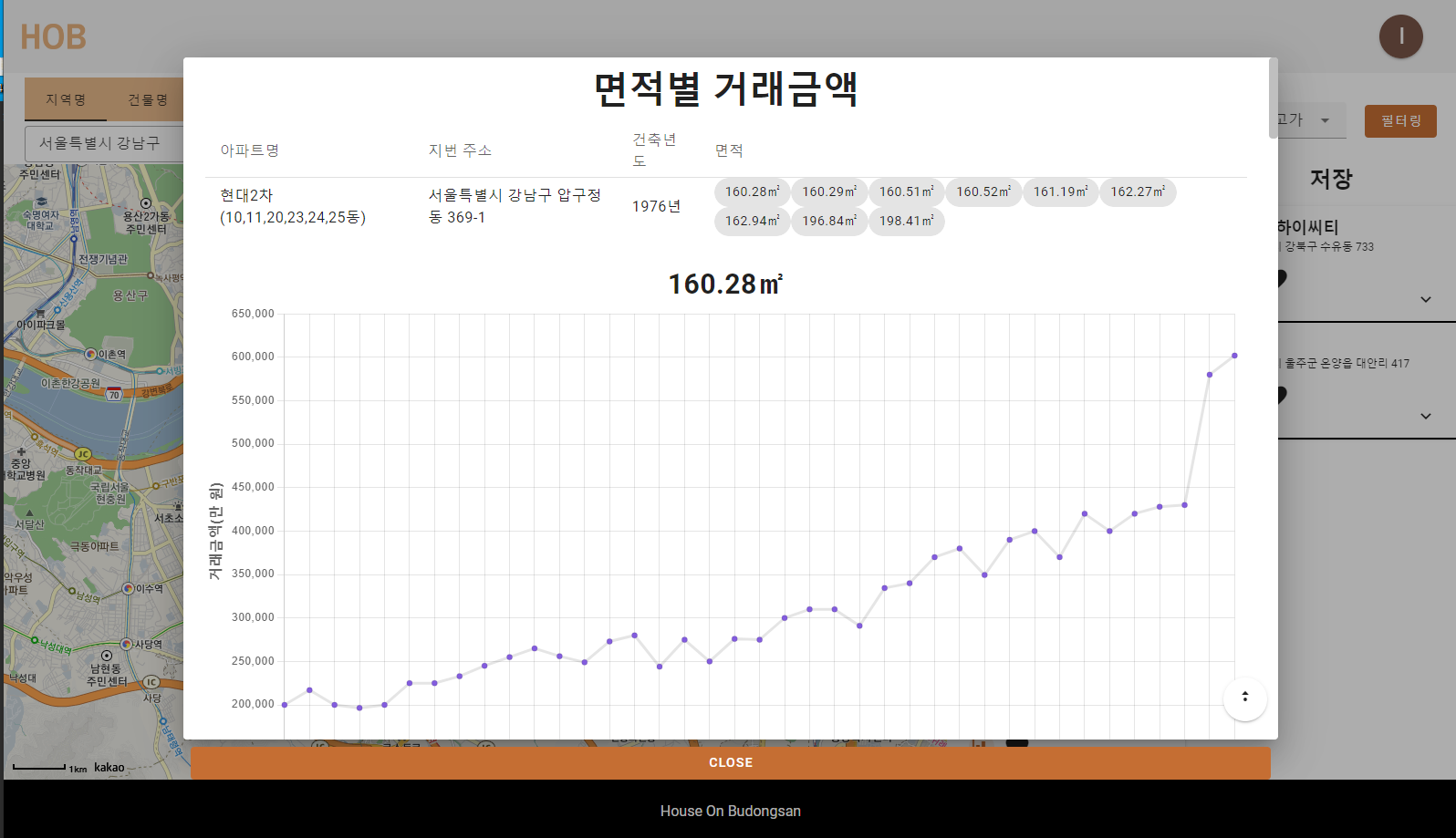
4-8. 필터링



4-6. 지역검색과 4-7. 건물검색에서 각각의 매물에 대해 필터링을 적용할 수 있다.

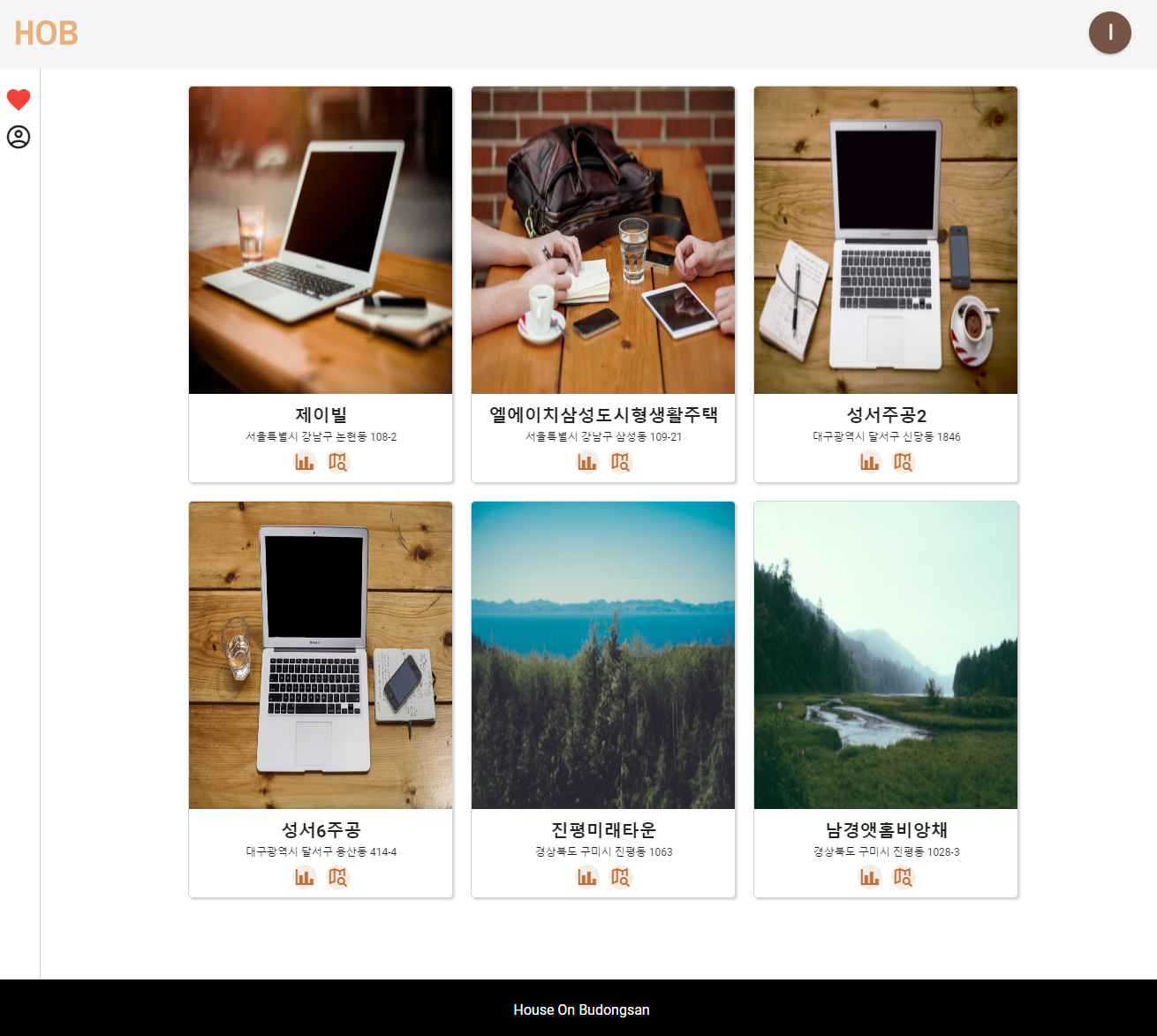
필터링의 항목으로 거래년도 및 월, 거래가격, 거래 면적을 최소 최대값을 통해 설정할 수 있다.이때 필터링은 다시 AJAX통신을 하는 것이 아닌 불러온 데이터를 기반으로 원본데이터에서 필터링을 수행하여 불필요한 AJAX통신을 수행하지 않도록 하였다.

4-9. 통계



4-6. 지역검색과 4-7. 건물검색에서 통계 버튼을 누를 경우 위와 같이 상세한 정보와 면적별로 나누어 일자별 거래 금액을 차트로 나타내었다.

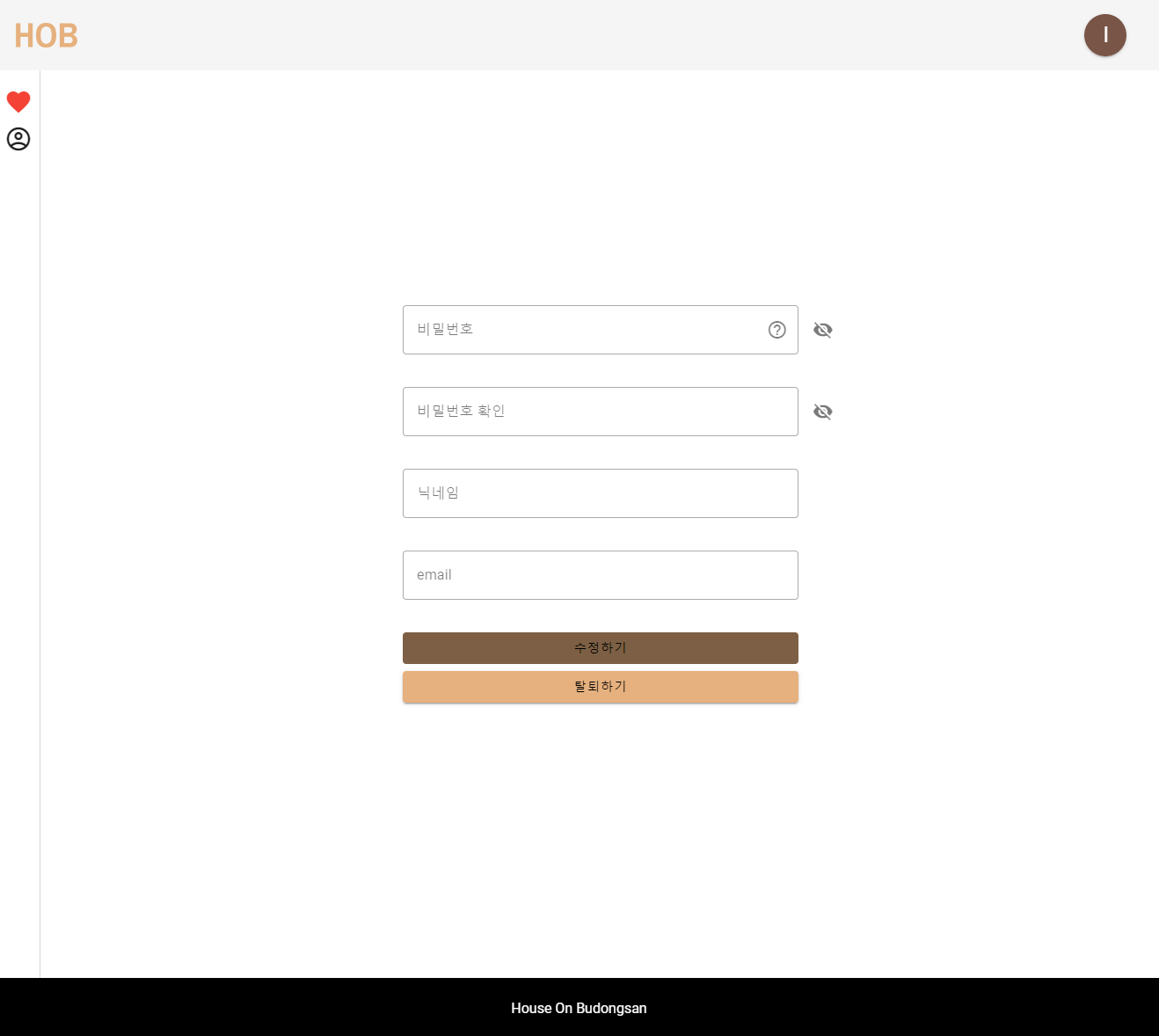
4-10. 좋아요



4-6. 지역검색과 4-7. 건물검색에서 좋아요 누른 매물에 대해 나타낸 페이지로 각각의 매물을 카드 형식으로 이미지와 함께 나타내었으며 검색결과와 동일하게 차트를 볼 수 있으며, 맵 아이콘의 버튼을 눌러 해당 매물에 대한 검색(4-7. 건물검색)으로 바로 갈 수 있도록 하였다.

위의 이미지의 경우 현재 더미이미지를 나타내었으나, 네이버 API중 이미지 검색을 통해 관계성이 높은 이미지를 넣을 수 있다. 하지만 부동산 데이터가 방대하여 모든 이미지를 넣기힘드므로 현재는 더미이미지를 넣어두었다.

4-11. 정보수정 및 탈퇴



4-3. 로그인 후 로그인 된 사용자에 대해 정보를 수정할 수 있다. 이때 비밀번호와 닉네임, 이메일에 대해 수정할 수 있도록 하였으며, 모든 입력폼이 입력되어야 `수정하기`버튼을 활성화하였다. 또한 탈퇴하기 버튼을 통해 탈퇴를 수행할 수 있다.