포트폴리오

이지원
(Jiwon,Lee)
(1995.09.01)

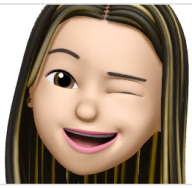
💌E-mail 95jiwon91@gmail.com

💾깃허브

EZ195 - Overview

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload

https://github.com/EZ195



내 프로젝트 리스트

📝MEMO

👉자신만의 메모를 사진과 함께 기록하고 리스트를 확인해 볼 수 있는 메모장 프로젝트

https://github.com/EZ195/Memo

❤️EZONISTA

👉게시글 업로드, 좋아요, 댓글 달기 기능을 갖춘 SNS 프로젝트

https://github.com/EZ195/Ezonista

🔗THE LINK

👉취미를 통해 사람을 연결하는 커뮤니티

https://github.com/EZ195/TheLink

테스트용 계정
ID : ezez9591@gmail.com
PW : dlwldnjs1!

바로가기

👉나와 똑같은 취미를 공유하는 사람을 주변에서 찾는 것은 쉬운 일이 아니다. 더욱이 설령 관심이 있더라도 혼자서 시작하기 어려울 수도 있다. 더링크는 취미를 바탕으로 사용자가 모임을 직접 주최하기도 하고 다른 사람의 모임에 참여하면서 같은 취미를 공유할 수 있도록 도와주는 커뮤니티이다. 관심 있는 분야를 처음 도전해보고 싶지만 막상 혼자서는 시작해볼 용기가 나지 않는 사람들, 혹은 혼자만 즐기던 취미를 다른 사람들과 공유하고 싶은 사람들의 니즈를 채워주기 위한 프로젝트이다.

👉더링크 사용자는 누구나 게시글을 작성해 모임을 주최할 수 있고 누구나 관심 있는 모임에 참여할 수 있다. 개최자는 게시글을 작성해 어떤 모임을 주최할 지에 대한 소개를 기재하고 최대 모집 인원과 모임 일자 등 모임과 관련된 자세한 정보를 작성한다. 참여자는 원하는 모임 게시글에서 모임 참가하기 버튼을 눌러 참여를 신청할 수 있다. 참여자가 신청 버튼을 누르면 주최자는 해당 게시글의 참여자 리스트를 통해 신청한 사람들의 내역을 볼 수 있고 주최자가 참여 승인을 하면 최종적으로 모임에 참여를 할 수 있게 된다.
주최자는 프로필에서 자신의 정보 및 자신이 참여 및 주최한 모임에 대한 정보도 함께 확인이 가능하다. 또 팔로우 기능을 이용해 많은 다른 유저들과 친분을 맺을 수도 있고 자신이 참여한 모임에 대한 후기도 남길 수 있다.

프로젝트를 진행하며...

해당 프로젝트를 진행하며 개발 시작 전 설계 단계의 중요성에 대해 많이 느꼈다.
프로젝트 초반에 나름 꼼꼼하게 DB를 설계했다고 생각했다. 그러나 프로젝트 진행 도중 닉네임 수정 기능을 만들며 'userBO'와 'profileBO'가 순환참조 오류가 발생을 했다. 해당 오류를 해결하기 위해 user 테이블에 있던 nickname을 profile 테이블로 옮겼던 경험이 있다. 다행히 개발 초반 단계라 테이블 수정이 코드에 큰 영향을 미치진 않았지만 만약 개발 후반부에서 유사한 일이 발생했다면 더 많은 코드를 대대적으로 수정해야 했을 것이다.

한편으로는 설계에 대한 유연적 태도도 필요하다는 것을 느꼈다.
미리 프로젝트를 설계하면 해당 설계를 따라 그대로 기능들을 만들기만 하면 된다고 생각했다. 그래서 프로젝트를 금방 완성할 수 있을 것이라고 생각했다. 하지만 개발 중간중간 만들다 보니 새로운 기능을 추가하기도 하고 기능의 의도를 약간 바꾸는 등 생각보다 많은 수정들이 발생했다.

개발 능력 측면에서도 많은 향상이 있었다.

기능 구현의 응용력이 프로젝트를 시작하기 전보다 많이 향상되었다고 느꼈다. THE LINK 프로젝트를 시작하기 전에는 한 기능을 구현하면 그 기능만 구현할 줄 알았는데 프로젝트를 진행하다보니 해당 기능보다 조금 응용된 기능도 혼자 고민해보면서 스스로 구현할 수 있게되었다.

또한, 코드의 오류를 파악하는 능력도 향상되었다. 프로젝트를 진행하며 무수히 많은 오류들을 직면하며 오류의 종류들을 파악하고 이를 깃허브의 이슈사항에도 기재하며 나만의 오류 데이터를 만들었다. 그에 이제는 어느정도 자주 나오는 오류들은 바로 대략 코드의 어떤 부분에서 오류가 났는지 가늠할 수 있게 되었다.
또한, css 및 부트스트랩에 대한 이해도가 증가되었다. 내가 직접 프로젝트의 모든 부분을 만들다 보니 프론트앤드의 영역 또한 내가 직접 해야했다. 이에 필요한 css 및 부트스트랩 기능을 검색을 통해 혼자 공부해보고 적용시켰다. 이 과정을 통해 다양한 css와 부트스트랩에 대해 어느 정도 수준의 이해도를 갖추게 되었다.

예상치 못한 오류가 정말 많았다. 특히 회원가입 기능을 구현하는 단계가 가장 기억에 남는 파트였다. 'EZONISTA' 프로젝트를 하면서도 만들어본 기능이었고, 다른 곳에서도 많이 해본 기능이기에 하루면 완성할 수 있을거라고 예상했다. 하지만 생각보다 많은 오류들이 발생하며 예상보다 많은 시간이 걸려서 회원가입 기능을 완성할 수 있었다. 회원가입 기능은 프로젝트 시작하고 가장 처음 만든 기능이어서 처음부터 막히는 것을 보고 아직 내 실력이 많이 모자란 건 아닐까 하는 의구심이 들면서 불안하기도 했다. 하지만 많이 부딪히고 고민하고 혼자 연구해 본 끝에 기능이 잘 구현된 것을 봤을 때 성취감과 약간의 자신감이 생기기 시작했다. 그 후에 다른 기능을 구현할 때도 많은 오류들이 발생했지만 전처럼 불안해하기 보다는 문제점을 어떻게 해결할지 고민하는데 집중했고 프로젝트를 잘 진행할 수 있었다.

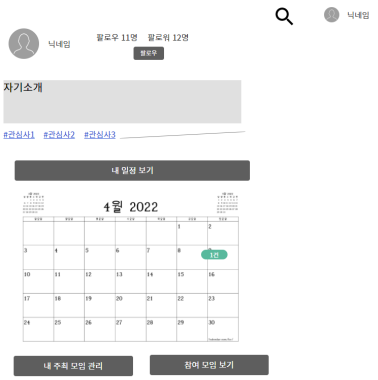
기술정보

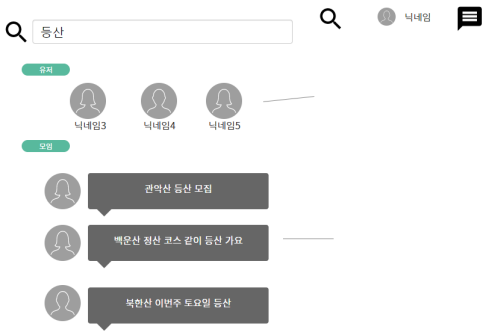
tomcat. Mysql, spring framework, Spring boot, Mybatis, JSTL, Bootstrap, jQuery, java, javascript, html

기획

오븐을 통해 ui, 페이지 구성 등을 미리 구상해봤다.

https://ovenapp.io/project/A7f353wQ1G9M1HEIunaXgJ7Xgn3I48DG#jBjrN





프로젝트 설계

1. DB 설계

https://docs.google.com/spreadsheets/d/1tvTfUghExmRRobg8bVFAnLEsTj6UBUfFjxusoX6ZcQw/edit#gid=0

데이터베이스는 스프레드 시트를 통해 설계했다.

테이블 : user

설명 : 사용자 정보 테이블

컬럼명	타입	NULL 가능 여부	AUTO_INCREMENT	설명
id	int	N	Y	primary key
loginEmail	varchar(64)	N	N	사용자 로그인 Email, 중복 체크 필요
password	varchar(64)	N	N	비밀번호, 암호화한 결과
bitrhYear	int	N	N	사용자 생일 연도
birthMonth	int	N	N	사용자 생일 월
birthDay	int	N	N	사용자 생일 일
gender	varchar(8)	N	N	사용자 성별
createdAt	timestamp	N	N	가입 날짜
updatedAt	timestamp	N	N	수정 날짜

테이블 : user_interest

설명 : 사용자 지정 관심사 테이블

컬럼명	타입	NULL 가능 여부	AUTO_INCREMENT	설명
id	int	N	Y	primary key
userId	int	N	N	사용자 ID (pk)
userInterest	varchar(16)	N	N	사용자 취미
createdAt	timestamp	N	N	생성 날짜

2. url 설계

view와 api를 분리해서 url을 설계를 했다.

https://docs.google.com/spreadsheets/d/1tvTfUghExmRRobg8bVFAnLEsTj6UBUfFjxusoX6ZcQw/edit#gid=1115453003

View URL			
제목	**URL**	**parameter**	**설명**
로그인	/user/singin_view		로그인 화면
회원가입	/user/singup_view		회원가입 화면
프로필	/user/profile/profile_view	userId : 사용자 ID	사용자 프로필 화면
프로필 수정	/user/profile/update_view	userId : 사용자 ID	사용자 프로필 수정 화면
내 모임 리스트	/user/myjoin_list_view	userId : 사용자 ID	사용자가 참여한 모임 리스트 화면
주최 모임 리스트	/user/myhost_list_view	userId : 사용자 ID	사용자가 주최한 모임 리스트 화면
참여 신청자 리스트	/post/join/approve_list_view	userId : 작성자 ID postId : 게시글 ID	모임에 참여 신청을 한 사용자 리스트 화면 게시글 작성자만 볼 수 있음
게시글 작성	/post/create_view		게시글 작성화면
게시글 수정	/post/update_view	postId : 게시글 ID	작성된 게시글 수정 화면
게시글 상세보기	/post/detail_view	postId : 게시글 ID	게시글 상세정보 화면
타임라인	/post/timeline_view		게시글 및 사용자 리스트 화면
검색 페이지	/search/search_view		사용자 및 게시글 키워드로 검색한 결과를 보여주는 화면
모임 그룹채팅 리스트	/chat/list_view		사용자가 참가하는 모임의 단톡방 리스트 화면
채팅창	/chat/detail_view		단톡방 채팅창 화면

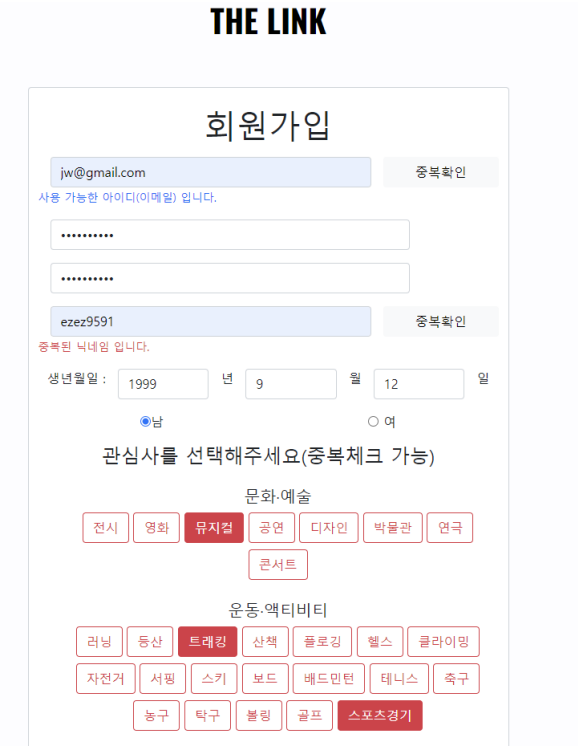
3. 일정설계

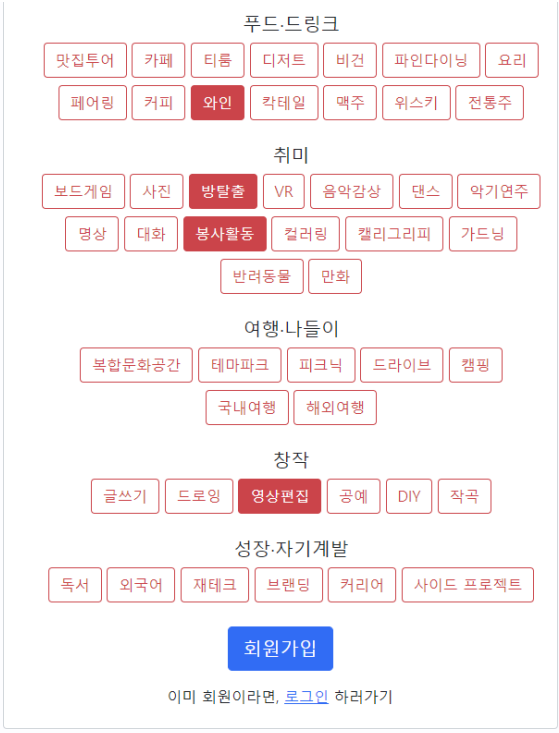
https://docs.google.com/spreadsheets/d/1tvTfUghExmRRobg8bVFAnLEsTj6UBUfFjxusoX6ZcQw/edit#gid=1995491386

단계		세부정보	완료 여부	4월																											5월				
				5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5
1	구상	프로젝트 기획	☑																															
2	설계	DB 설계	☑																															
	설계	URL 설계	☑																															
3	구현	회원가입 기능 구현	☑																															
	구현	관심사 저장 기능 구현	☑																															
	구현	로그인 기능 구현	☑																															
	구현	로그아웃 기능 구현	☑																															
	구현	프로필 생성 기능	☑																															
	구현	프로필 수정 기능 구현	☑																															
	구현	게시물 글쓰기 기능 구현	☑																															
	구현	게시물 리스트 구현	☑																															
	구현	게시물 세부화면 구성	☑																															
	구현	게시물 수정 기능 구현	☑																															
	구현	게시물 삭제 기능 구현	☑																															
	구현	모임 신청/취소 기능	☑																															
	구현	참여자 승인 기능	☑																															
	구현	후기 작성 기능 구현	☑																															
	구현	후기 수정 기능 구현	☑																															
	구현	후기 삭제 기능 구현	☑																															
	구현	팔로우/언팔로우 기능 구현	☑																															
	구현	해시태그 검색 기능 구현	☑																															
	구현	키워드 검색 기능 구현	☑																															
	구현	캘린더 기능 구현	☑																															
4	테스트	회원가입 및 로그인 테스트	☑																															
	테스트	타임라인 구성 테스트	☑																															
	테스트	게시글/후기 기능 테스트	☑																															
	테스트	프로필 및 팔로우/언팔 기능 테스트	☑																															

기능 소개

⭐회원가입 기능





회원가입 시 아이디(이메일), 비밀번호 닉네임, 생년월일 , 성별, 관심사를 기재하도록 구성했다.
회원 구분을 위해 아이디와 닉네임에 중복 체크 기능을 넣었다.

'관심사'는 다중 체크가 가능하도록 했다.
user 테이블이 있음에도 불구하고 관심사는 user_intereset 테이블에 별도로 저장되록 DB를 설계했는데 이는 기획 단계에서 해당 관심사를 해시태그의 기능으로 활용해 사용자가 자신과 같은 관심사를 가진 다른 사용자를 쉽게 찾을 수 있도록 구상을 했기 때문이다. 사용자가 회원가입을 할 때마다 고유의 id를 부여받는데 해당 id 컬럼을 user_interest테이블의 userId 컬럼으로 저장해 사용자 정보와 매치할 수 있도록 설계했다. 이렇게 관심사는 다른 DB 테이블에 저장함으로써 기능 구현 측면에서 더욱 다양하게 활용할 수 있을 것이다.

회원가입 시 사용자 프로필도 자동으로 생성되도록 기능을 구현했다. MD5 (Message-Digest algorithm 5)를 통해 비밀번호를 암호화했다.

😅trouble shooting

⚠️ 회원가입 시 성별 선택(라디오)의 유효성 검사가 작동하지 않았다

🛠️ 유효성 검사를 확인하는 스크립트에서 if 조건문을 라디오 값 == “”이라고 작성했던 것이 오류 발생의 원인이었다. 그래서 “” 를 null로 바꿔줬다. 위에 조건문들은 다 “”로 작성하다보니 라디오도 이렇게 작성하게 된 것 같다.

⚠️ 관심사를 다중체크 해도 user_interest 테이블에 하나의 관심사 데이터 값만 전달하는 문제가 발생했다.

🛠️ 회원가입 기능을 구현하는 MVC모델의 controller에서 체크된 모든 interest의 값들을 list에 담아 전달되도록 수정했다. 그 후 UserInterestMapper의 insert 쿼리문에서 foreach 반복문을 사용해 list로 가져온 userInterest의 아이템을 하나씩 꺼내 Value의 () 내용이 반복되도록 코드를 작성했다.

➕업데이트 예정

- spring security를 이용해 로그인을 관리할 예정
- 비밀번호 생성조건 강화 (ex. 8자리 이상/숫자/ 특수문자 필수 등)

⭐로그인 기능

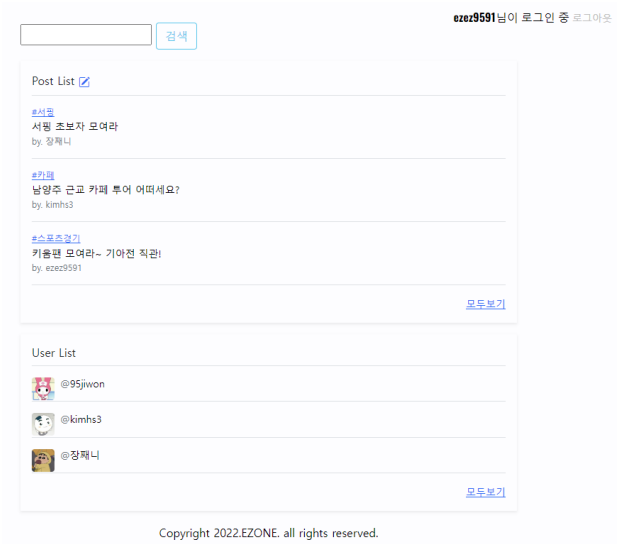
가입한 이메일과 비밀번호를 입력하면 로그인이 가능하다.
만약 가입된 정보가 아닌 이메일을 입력하거나 비밀번호를 잘못 입력시에는 아이디와 비밀번호를 확인하라는 알림 문구가 나타난다.

Interceptor를 통해 로그인 여부에 따른 접근 권한에 제한을 뒀다.
로그아웃 상태인 경우 로그인 페이지와 회원가입 페이지만 접근할 수 있도록 했고 그 외의 페이지 주소를 입력 시 로그인 페이지로 돌아가도록 설정했다.
반대로 로그인 상태인 경우 로그인과 회원가입 페이지로 접속시 타임라인 페이지로 돌아가도록 설정했다.

가입한 이메일과 비밀번호를 입력하면 로그인이 가능하다.
만약 가입된 정보가 아닌 이메일을 입력하거나 비밀번호를 잘못 입력시에는 아이디와 비밀번호를 확인하라는 알림 문구가 나타난다.

Interceptor를 통해 로그인 여부에 따른 접근 권한에 제한을 뒀다.
로그아웃 상태인 경우 로그인 페이지와 회원가입 페이지만 접근할 수 있도록 했고 그 외의 페이지 주소를 입력 시 로그인 페이지로 돌아가도록 설정했다.
반대로 로그인 상태인 경우 로그인과 회원가입 페이지로 접속시 타임라인 페이지로 돌아가도록 설정했다.

⭐타임라인



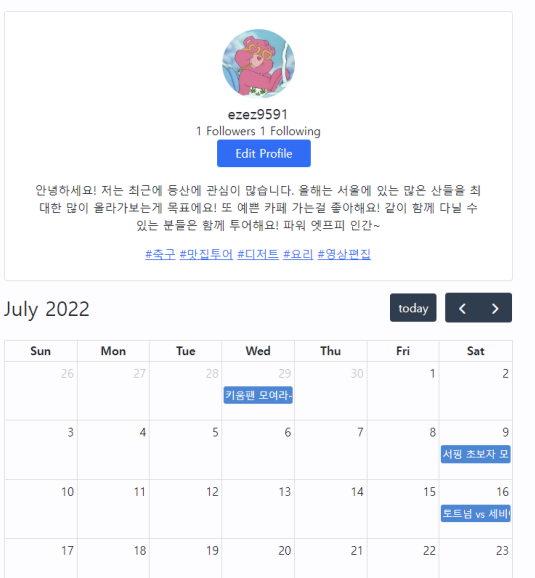
로그인 시 바로 보이는 메인 페이지이다.
페이지 상단에 로그인 한 유저의 닉네임이 뜨며 로그아웃 클릭 시 로그인된 계정은 로그아웃된다.

게시글 리스트와 유저 리스트를 전부 볼 수 있으며 최대 세개의 리스트까지 보여지며 더 많은 게시글 및 유저 리스트를 보고 싶은 경우 모두보기를 클릭하면 모든 게시글과 모든 유저의 리스트를 한 화면에서 볼 수 있다.

검색어를 입력하면 해당 검색어와 관련된 게시글 및 유저를 검색할 수 있는 검색창이 있다.

게시글 제목을 클릭하면 게시글 세부화면으로 넘어가 더 자세한 정보를 볼 수 있으며 유저의 경우 닉네임을 클릭하면 선택한 유저의 프로필 화면으로 넘어간다.

⭐프로필 화면



프로필 화면에서는 사용자의 닉네임, 자기소개, 프로필 화면, 해시태그 형태로 된 관심사를 볼 수 있다. 또한 사용자가 주최한 모임 리스트와 사용자가 참여신청 내역 리스트를 한번에 볼 수 있으며 주최 및 참여할 일정들도 캘린더에 표시되도록 화면을 구성했다.

다른 유저를 팔로우할 수 있다. 다른 유저를 follow 버튼을 눌러 팔로우하면 해당 버튼은 following으로 바뀐다. following버튼을

누르면 자신이 팔로우한 유저를 언팔로우할 수 있다. 자신의 팔로워 수와 나를 팔로우하고 있는 사람의 수도 확인이 가능하다.

본인이 본인의 프로필을 보는 화면과 다른 유저의 프로필을 볼 때의 프로필 화면 구성에 약간의 차이점을 두었다. 타인의 프로필과 자신의 프로필의 구분은 session에 저장된 userId와 프로필의 userId의 일치 여부를 통해 판달할 수 있도록 했다.
자신의 프로필을 보는 경우 프로필 수정 버튼이 나타나고 해당 버튼을 통해 자신의 프로필을 수정할 수 있다. 또한 모임 참여 내역이 없는 경우 모임 참여하기 링크를 통해 타임라인으로 이동할 수 있게 하고 모임 주최 내역이 없는 겨우 모임 주최하기 링크를 통해 바로 게시글을 작성하는 페이지로 이동할 수 있게 화면을 구성했다. 또한 자기 자신은 팔로우할 수 없게 팔로우 버튼도 나타나지 않는다.
반면 다른 유저의 프로필을 보는 경우 수정버튼은 보이지 않고 유저의 정보만 확인할 수 있게 되어있다. 또한 모임 참여 및 모임 주최 내역이 없는 경우 링크는 나타나지 않는 대신 모임 내역이 없다는 문구만 나타나도록 했다. 캘린더 일정도 자신의 프로필이 아니면 나타나지 않는다.

😅trouble shooting

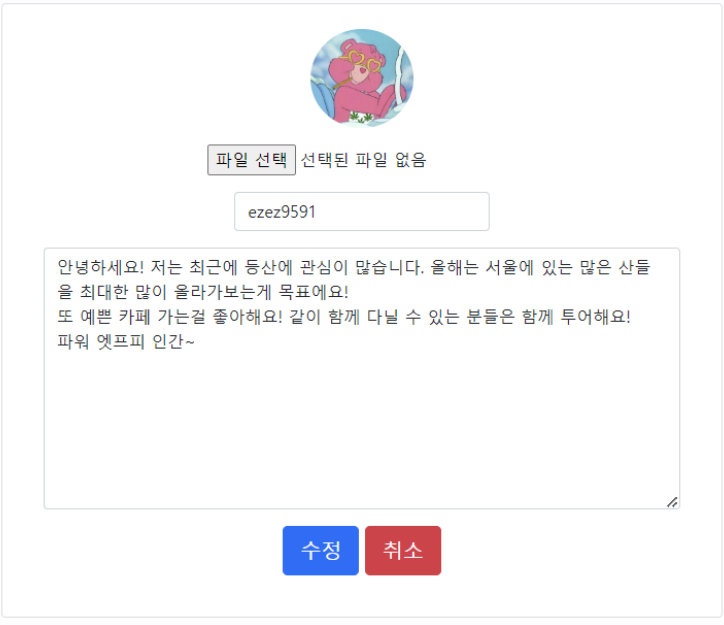
⚠️ fullcalendar를 이용해 스케줄 확인 기능을 만들고자 했다. 사용자들이 별도의 메모 없이 자신의 스케줄을 확인할 수 있도록 이용자가 모임을 개최할 때와 이용자가 모임 신청할 경우 자동으로 fullcalendar에 등록되록하는 기능이다. 내가 참가하는 모임 리스트와 내가 주최한 모임 리스트에 사용된 model에 담긴 리스트들을 활용해 스크립트에서 반복문을 이용해 기능을 구현하려고 했다. 그러나 for문을 사용해 기능을 구현했을 때 각 리스트의 원하는 항목들을 뽑아낼 수 없는 오류가 발생했다.

🛠️ 문제 해결을 위해 구글링에 검색해 본 결과 for문에서 변수로 설정한 i를 인식하지 못하기 때문에 리스트 객체는 for문을 이용한 반복문을 사용할 수 없다는 것을 알게 되었다. 그 대신 스크립트에서 c태그를 이용한 반복문을 사용할 수 있다는 것을 알 수 있었다. 그래서 c태그의 for 반복문을 통해 코드를 만들었고 원하는 기능을 구현할 수 있었다.

➕업데이트 예정

- 팔로워 및 팔로우 리스트

⭐프로필 수정기능



프로필 이미지, 닉네임, 자기소개를 수정할 수 있다.

😅trouble shooting

⚠️ 닉네임 수정 시 userBO와 profileBO의 순환 참조 문제가 발생했다.
처음에는 회원가입 시 nickname은 user 테이블에 저장되도록 DB를 설계했다. 그러나 프로필 화면에서 프로필 수정 기능을 만들 때 닉네임/ 자기소개/프로필/관심사가 수정되도록 했는데 이 때 문제가 발생했다. 회원가입 시 자동으로 프로필이 생성되도록 만들 때 이미 userBO의 addUser메소드에서 profileBO의 addUserProfile메소드를 사용했다. 근데 프로필 수정을 구현할 때 user테이블에 있는 닉네임 데이터를 함께 수정하는 과정에서 반대로 profileBO에서 userBO를 참조해야 했다.

🛠️ user테이블과 profile 테이블의 DB 설계를 다시했다. user테이블에 있던 nickname 컬럼을 profile테이블의 컬럼으로 DB 구조를 바꿔 회원가입 시 닉네임은 user테이블이 아닌 profile 테이블에 저장되도록 회원가입 기능 코드도 함께 수정했다. 이렇게 함으로써 닉네임을 수정할 때 userBO를 참조할 필요가 없게 되어 순환 참조의 문제를 해결할 수 있었다.

⚠️ 프로필 이미지를 수정해도 프로필에 이미지가 뜨지 않는 오류가 발생했다.

🛠️ 업로드된 이미지의 파일을 생성하는 FileManagerService에서 return 값인 이미지 파일 경로를 잘못 입력해서 오류가 발생함을 발견했다. 해당 return 값이 ImagePath파라미터에 담겨 view에서 보여줄 때 제대로된 경로가 있어야 이미지를 보여주는데 return 값 자체가 잘못되어 발생한 문제였다. 올바른 경로를 입력해 오류를 수정했다.

⚠️ 기존의 프로필 이미지가 있는 상태에서 프로필 이미지를 변경하는 경우 에러 페이지가 뜨는 오류가 발생했다.

🛠️ profileBO의 updateProfile메소드에 조건문을 추가해 프로필이 수정되는 경우의 수를 3가지 나눠 분기문을 짰다. file값이 null일 때와 새로 입력되었을 때로 나눠 기본값이 입력되도록 수정했다.

➕업데이트 예정

- 관심사 수정 기능 추가 예정

⭐모임 주최하기 기능

타임라인 화면에서 연필 모양의 아이콘(📝)을 누르면 바로 모임을 주최할 수 있다.
어떤 모임인지 알 수 있도록 제목을 입력하고 하단에는 어떤 모임인지에 대해 자세한 내용을 기재하도록 되어있다.
또한 모임날짜를 캘린더 형식으로 입력이 가능하며 모임 주최자가 수용가능한 최대 인원도 설정할 수 있다.

TMI) 원래 초기 DB설계 당시 post_interest 테이블을 만들어 게시글의 카테고리를 따로 저장해 관리하려고 했다. 검색에 잘 노출될 수 있도록 모임 주최자가 모임과 관련 없는 카테고리를 무분별하게 선택할 수도 있겠다는 생각이 들었다. 그래서 게시글의 카테고리를 단일 선택하도록 제한하는 것이 더 좋은 서비스를 제공할 수 있을 것이라고 생각했다. 게시글 관련 카테고리가 중복 선택이 아니라면 굳이 카테고리를 독립된 DB로 관리할 필요가 없어졌다. 이에 카테고리 컬럼도 post 테이블에서 한번에 관리하도록 DB를 다시 설계했다.

➕업데이트 예정

- 주의사항 작성칸 따로 만들기
  → 대부분 사람들은 긴글을 읽고 싶어하지 않을 뿐더러 자세히 읽지 않음, 그래서 설령 모임 주최자가 주의사항을 적어도 참여자들이 주의사항을 제대로 읽지 않을 가능성이 높음. 그래서 글을 작성할 때 주의사항 작성란을 따로 만들어 자세히 글을 읽지 않는 사람도 주의사항을 쉽게 숙지할 수 있도록 가독성을 높일 예정

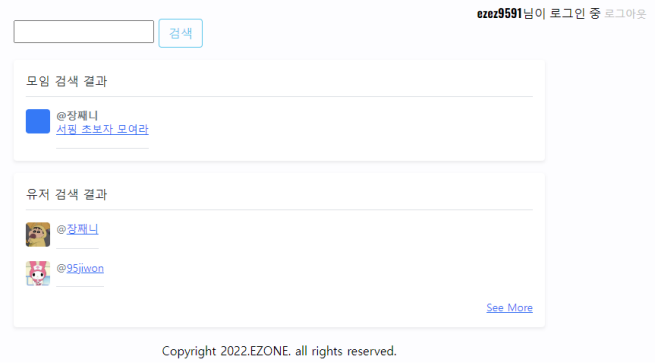
⭐후기 작성 기능

모임이 진행된 후 모임 참여자들은 모임 게시글에 댓글처럼 모임이 어땠는지에 대한 간단한 한 줄 평을 남길 수 있는 기능

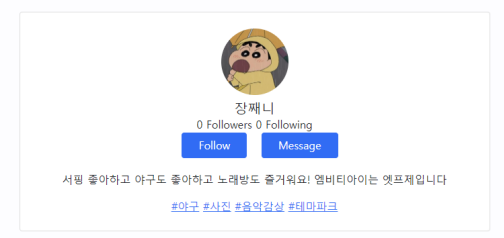
➕업데이트 예정

- 주최자의 매너 온도(당근마켓처럼)
  → 모임이 원래 모임의 취지와 맞게 잘 진행되었는지, 주최자가 모임을 잘 진행 시켰는지 참가자들이 잘 참여할 수 있도록 잘 지도했는지 등 모임을 진행했던 주최자의 태도를 참가자들이 평가하는 기능, 이런 평가들을 통해 좀더 건전한 모임활동이 개최될 수 있도록 하기 위함

⭐검색 기능



'서핑'이라는 키워드로 검색한 검색결과 화면



해당 유저가 서핑으로 검색된 이유는 서핑이 자기소개와 관심사에 있기 때문

검색어를 입력하면 해당 검색어와 관련된 게시글과 사용자가 조회되는 기능이다.

게시글은 검색어가 제목, 글 내용, 카테고리 중 일치하는 내용이 있으면 검색된다.
사용자는 자기소개나 관심사에 일치하는 내용이 있으면 검색된다.
사용자도 검색이 될 수 있도록 한 이유는 검색을 한다는 것은 그만큼 관심이나 흥미가 있다는 것이기 때문에 자신과 공통 관심사를 갖고 있는 회원을 쉽게 찾도록해 활발한 소통을 유도하기 위함이다.

기능은 크게 키워드 검색과 해시태그 검색이 있다.
해시태그 검색은 개인 프로필 내에 있는 관심사에 링크를 걸어 해당 해시태그를 클릭 시 그 해시태그를 검색어로 검색된 결과가 화면에 출력된다. 해시태그를 클릭하면 해당 해시태그가 검색어가 되어 해당 단어를 포함하고 있는 게시글 및 사용자가 있는지를 post, user, user_interest 테이블에서 DB를 가져온다. user_interest DB에는 userId와 관심사만 저장되어 있어 일치하는 사용자 정보를 가져오려면 가져온 데이터를 다시 한번 user테이블에서 select해와야 한다. user_interest에서 가져온 데이터의 userId를 이용해 user테이블과 일치하는 데이터를 가져온다. 만약 관심사와 자기소개에 같은 단어가 들어가 중복 검색된 사용자가 있다면 이를 Controller에서 중복 결과를 조회해 한쪽만 중복된 데이터를 삭제 후 다시 하나의 배열로 합쳐 결과를 view에 보여준다

위의 예시 이미지는 '서핑'이라는 단어로 검색한 결과이다. 유저의 경우 자기소개 부분과 관심사 부분에 모두 서핑이라는 단어가 들어간다. 따라서 원래대로라면 관심사에 서핑이 걸린 '장째니'의 프로필과 자기소개에 서핑이 걸린 동일한 '장째니' 유저가 두번 보여지게 된다. 하지만 중복 검색된 데이터를 모두 보여주는 것은 사용자 입장에서 비효율적인 서비스 제공이다. 이에 검색된 해당 유저의 userId를 비교해 일치한다면 관심사 쪽의 데이터를 지운 뒤 남은 데이터를 합쳐 결국 '장째니' 유저의 프로필이 한번만 검색되어 나오는 것이다.

키워드 검색은 키워드를 검색 창에 입력하면 검색된 결과가 화면에 출력된다.
키워드를 검색하면 post테이블에서 제목, 내용, 카테고리에서 해당 키워드가 포함된 게시글들을 배열 형태로 가져온다. 가져온 배열은 DAO, BO를 거쳐 Controller에서 모델에 담아 view 화면으로 보내화면에 결과를 보여준다.

😅trouble shooting

⚠️ 중복된 데이터를 가져오는 문제점이 있었다. 프로필을 검색할 때 자기소개와 관심사에서 검색된 키워드가 포함된 사용자의 정보를 가져온다. 그러나 다중체크된 관심사를 저장하기 위해 관심사는 user_interest테이블에서 따로 관리가 되고 있다. 그래서 검색 시 자기소개에 키워드가 포함되어 있는 프로필은 profile 테이블에서 select문을 통해 postHashtagResult 관심사에 키워드가 포함되어 있어서 user_interest테이블에서 userId를 가져와 이를 이용해 다시 profile 테이블에서 select를 통해 데이터를 가져오게 된다. 이 때 중복된 프로필을 가져오게 되는 문제점이 발생했다. postHashtagResult

🛠️ 이런 방식으로 가져온 두 가지의 데이터를 searchController에서 for문을 통해 두 데이터들의 userId를 비교했다. userId가 일치하면 관심사를 통해 유저 프로필을 가져온 배열의 해당 데이터를 삭제했다. 이후 중복된 데이터가 삭제된 관심사가 일치하는 유저 프로필 배열을 자기소개에 검색한 키워드를 포함하는 유저 프로필 배열에 추가했다. 이를 모델에 담아 search.jsp에서 결과를 출력했더니 중복된 데이터 없이 일치하는 유저 프로필을 잘 가져올 수 있었다.

⚠️ 특정 몇 개의 검색어를 입력하면 데이터가 검색 결과가 null이 나온다는 오류를 발견했다. 디버깅을 통해 확인해 봤더니 controller까지는 일치하는 데이터를 잘 가져오는 것을 확인했다. 그 다음 단계인 데이터의 중복을 체크하는 부분이 오류인 것 같다는 추측을 했다.

🛠️ 검색을 통해 배열의 데이터를 지우기 위해서는 for문을 사용하기 보다는 iterator를 사용하는 것이 적합하다는 것을 알 수 있었다. 중복된 데이터를 삭제할 필요가 있는 postHashtagResult 배열은 iterator를 사용했고 postHashtagResult 배열의 객체는 post1에 담았다. 데이터 중복을 확인하기 위해 필요한 postResult 배열은 for문을 사용했고 객체는 post2에 담았다. 그리고 반복문이 돌아가는 동안 if문을 통해 post1과 post2의 Id를 가져와 비교해 id가 같을 때 postHashtagResult 배열에서 중복된 객체가 삭제되도록 코드를 구현했다.