끈기 있는 개발자,
최주향 입니다

SPRING PROJECT

CREATIVE DEVELOPER

010-5645-7352
say1890@naver.com

끈기 있는 **개발자** 최주향입니다.

about me

최주향
2000.10.12
010-5645-7352
say1890@naver.com
https://github.com/say1890

certificate

정보처리산업기사 (최종)
정보처리기사 (필기)
토익 855
토익 스피킹 LV.6
컴퓨터 활용능력 1급
GTQ 1급

기술 정보

“mbting”

tomcat, mysql, mongodb

Spring Framework, Spring boot, Mybatis, Jstl

Bootstrap, JQuery, javascript, html

Java

MBTING

소개팅 사이트 프로젝트

프로젝트 소개

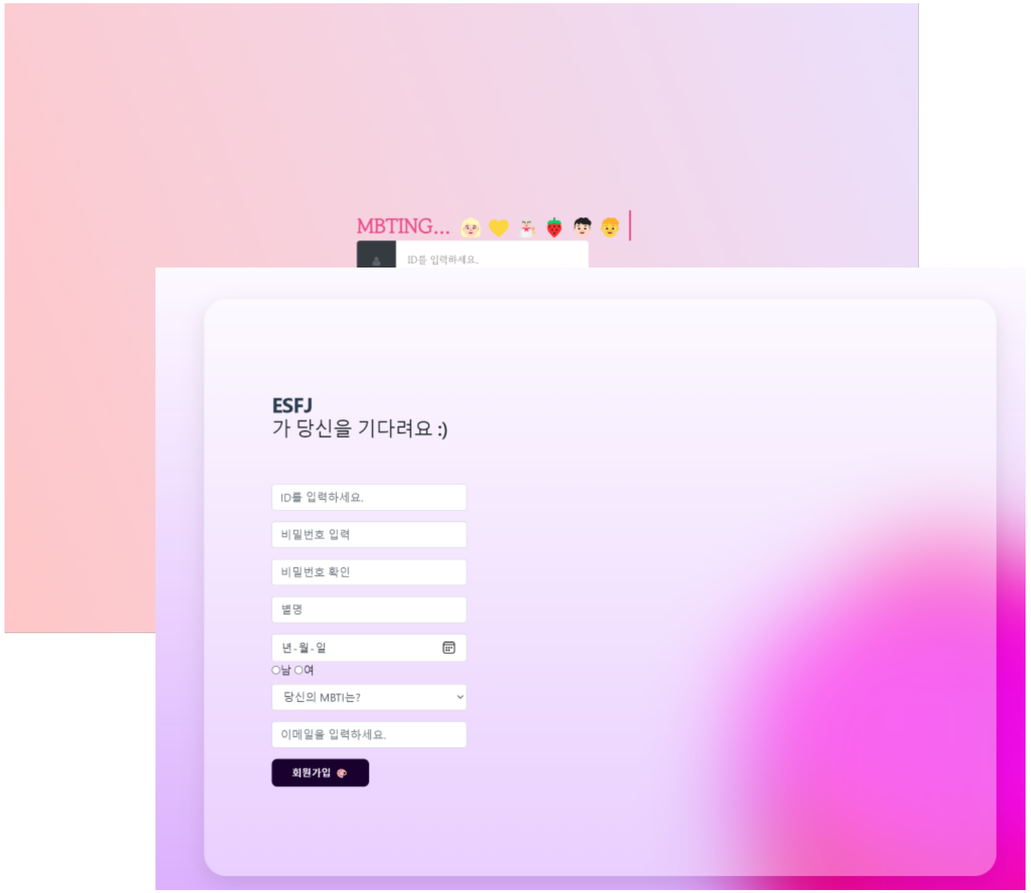
진행하며 느낀 점

진행 단계

샘플

mbting

진행기간	2022-02-21-2022-03-31
프로젝트 소개	프로필을 입력하면 사용자에게 맞는 사람을 추천해주는 소개팅 홈페이지



1. 기초가 중요하다

"내 프로젝트는 튼튼한 아파트인가?"

프로젝트를 진행하던 중, 사이트를 더 예쁘게 꾸미고 싶다는 욕구가 생겨 여러 가지 CSS, JS 파일들을 프로젝트에 넣게 됐다.
여러 가지 파일들이 들어가다보니 로딩 속도가 느려졌고, 이를 해결하기 위해서는
 보다 **'근본적인 지식'** 이 있어야 한다는 것을 깨달았다.
구글링을 하다가 웹 개발에는 서버 부하를 분산하는 ***로드밸런싱***, 시용자의 요청을 유실하지 않고 많이 받을 수 있는 ***메시징 큐***, 빠른 데이터 읽기를 위한 ***Redis***, 일괄적으로 대량의 작업을 하는 ***Batch***,
모듈간 통신을 위한 소켓 통신 및 ***rpc*** 등 시스템의 견고함을 높여주는 많은 기술이 있다는 걸 알게 됐고
이러한 기술들을 익혀서 , 추후에 더 큰 규모의 프로젝트를 맡게 됐을 때 더욱 **완성도 높은 웹**을 만들어야겠다는 생각을 하게 됐다.
또한 데이터를 가져오는 과정에서 mysql 코드 짜는 것이 생각보다 쉽지 않다는 것을 느끼게 됐고, 데이터 베이스쪽도 관련 자격증 취득을 하며 공부를 더 해야겠다는 생각을 하게 됐다.

2. 개발에 관하여

"내가 지금 무엇을 하고 있는지 파악하자"

1. MVC 모델에 관하여

MVC 모델을 적용한 프로젝트를 진행하면서 끊임 없이 되새긴 것은 바로
'각각을 분리해서 생각하자'이다.

각각의 기능을 모듈 형태로 분리하여 생각하는 것이 처음에는 쉽지 않았지만,
컨트롤러는 컨트롤러의 일이 있고, 뷰는 뷰의 일이 있고, 모델은 모델의 일이 있다고
정확히 이해를 한 뒤에는 비로소 일이 순조롭게 진행됐고, 단순히 개발 일정 진행에만 신경 쓰는 것이 아니라
내가 지금 무엇을 이용하여 무엇을 하려고 하는지를 다시 한 번 점검하게 되는 시간이었다.

2. 개발에 관하여

2. 코드에 관하여 (노력한 점)

1. 작업을 진행하다 보니 고정된 작업 흐름을 가지며 자주 ***중복 되는 코드*** 가 많았는데,
이를 최대한 ***분리*** 할 방법을 생각하려고 노력했다.

2. 코드를 작성할 때 최대한 ***규칙*** 을 맞추려고 노력했다.

3. 비동기, 동기 등의 프로그래밍 용어들이 처음에는 낯설었지만 검색을 통해 찾아보며 기본적인
용어의 뜻을 익혀갔다.

2. 개발에 관하여

2. 코드에 관하여 (노력한 점)

4. 에러가 발생하면 ***에러 로그***를 분석하고 해결해나가며 ***문제 해결 능력*** 을 키웠다.
크롬에서는 개발자 도구를, 이클립스에서는 디버깅 기능을 활용했다.

5. codepen에서 오픈소스를 활용하며 코드를 프로젝트에 맞게 변형시키는 경험을 하게 됐다.

6. 코딩을 시작하기 전 ***DB 설계*** 와 ***URL 설계*** 를 통해 구현하고자 하는 기능을 미리 생각했고,
이 덕에 ***작업 시간을 최소화***시켰다.
이 과정에서 프로그램의 구조를 자연스럽게 그리게 됐다.

3. 아쉬운 점

1. 사용자에 대한 것을 다 UserBO 쪽에 코드 작성을 하다 보니 *가독성*이 떨어졌다.

-> RecommendBO 를 만들어 사용자를 추천해주는 *메소드를 분리* 하는 것이 좋을 것 같다.

-> 내가 좋아하는 사람의 프로필 가져오는 것은 LikeBO 쪽으로 빼는 것이 더 좋을 것 같다.

코드의 재사용을 최대한 높이고 중복된 모듈을 최소화시켜야겠다.

2. 서버의 크기

아무래도 체험용 서버이다보니 서버에 부하가 많이 걸려서 자주 다운되고는 한다.

다음에 여유가 된다면 더 좋은 사양의 서버로 개발을 해보고 싶다.

3. 사용자 추천 부분

싫어요를 눌렀을 때 싫어요를 누른 사용자와 비슷한 사용자를 추천 순위에서 떨어뜨릴 수 있다면

퀄리티가 더 높아질 것 같다.

3. 아쉬운 점

4. mongodb 활용도

MongoDB를 Spring에 연동 시켜서 사용하는 것은 처음이었고,
MySql에 비해서 자료의 양과 사용자의 수가 현저히 적어서 개발하는 것에 한계를 느꼈다.
원래 구현하려고 했었던 채팅방 삭제 기능을 구현하지 못했던 것이 매우 아쉽다.

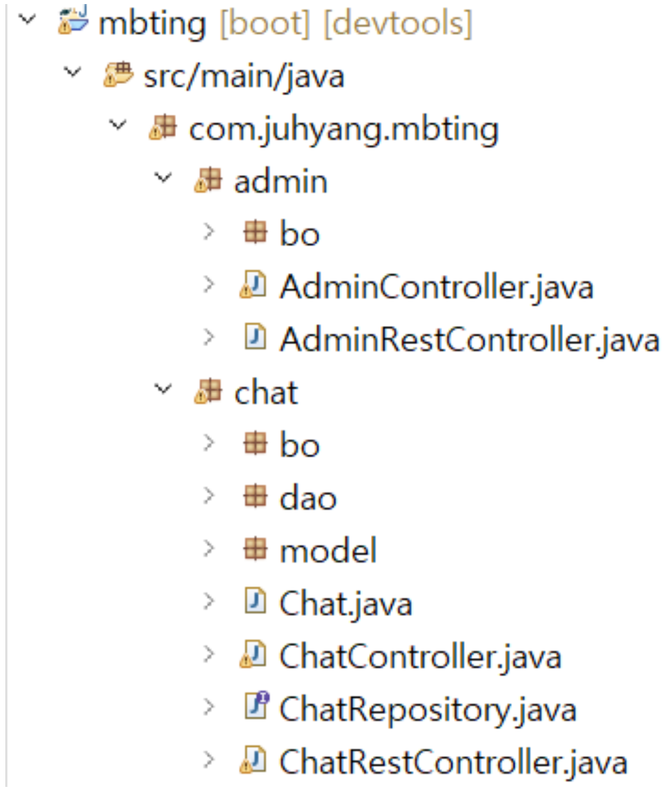
5. 캘린더 기능을 넣지 못한 것

관리자가 게시글을 올릴 때 직관적이게 볼 수 있도록 캘린더 형식에 데이터를 추가하면
사용자가 바로 볼 수 있게끔 하려고 했지만 시간 관계상 넣지 못한 점이 아쉽다.

4. 장점

1. 표준을 잘 따랐다.

들여쓰기의 크기, 파일과 변수의 명명규칙을 잘 지켜서 사용했다.



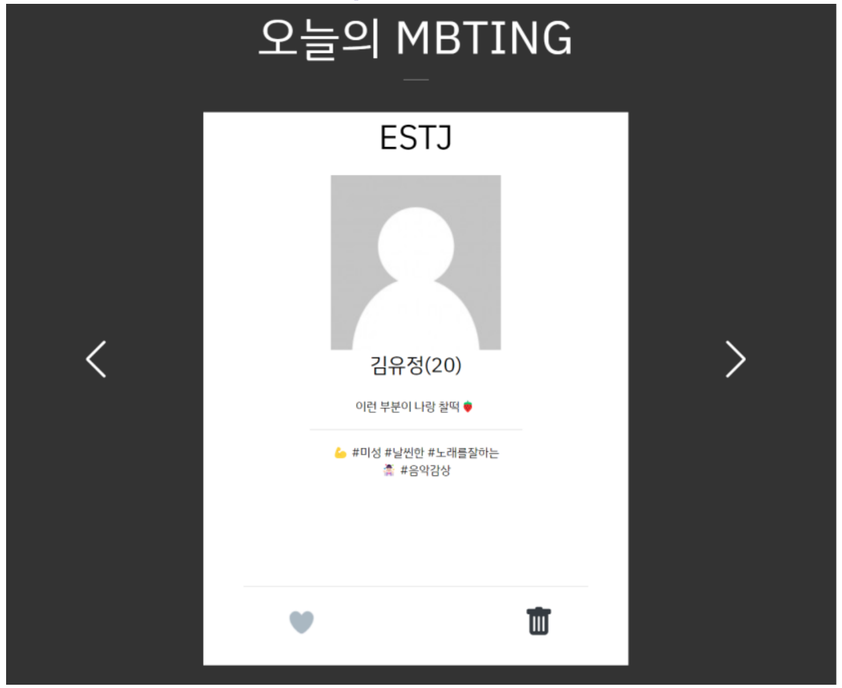

```
14      <link rel="stylesheet"  href="/static/css/heart.css"/>
15
16  </head>
17  <body>
18    <div class="slider">
19      <div class="container slidercontent">
20        <h1 class="hero">오늘의 MBTING</h1>
21
22        <div class ="row justify-content-center ">
23         <c:choose>
24            <c:when test ="${not empty userList}">
25             <div class="swiper">
26              <!-- Additional required wrapper -->
27                 <div class="swiper-wrapper">
28
29                    <!-- Slides -->
30                    <c:forEach var="user" items="${userList}">
31                    <div class="swiper-slide">
32                       <div id="profile-box" class ="bg-white  mx-auto">
33
34
35                                          <div id = "mbti" class=" bg-white mt-3 text-center">
36                                              ${user.user.mbti}
37                                          </div>
38                                          <!--  추천 상대 프로필 이미지 -->
39                                          <c:choose>
40                                              <c:when test = "${not empty user.user.profile}">
```

4. 장점

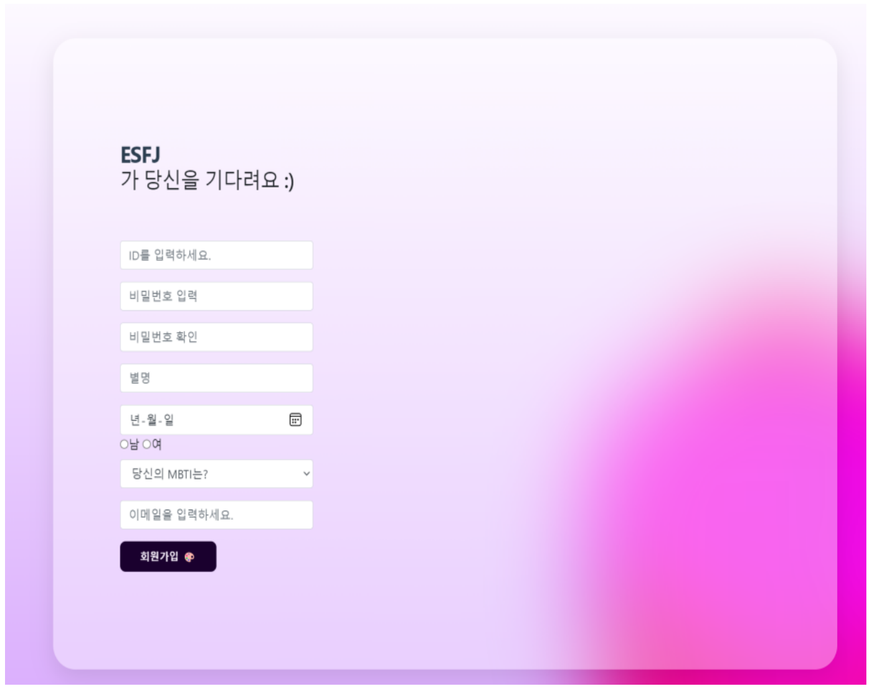
2. 플러그인, 외부 라이브러리, 소스코드를 잘 활용했다

사용자 추천창을 swiper를 사용하여 만들었고, 이 과정에서 플러그인 활용도가 높아졌다.
또한 jquery를 사용하여 개발에 편리성을 더했다.

플러그인 사용 (swiper) ▼

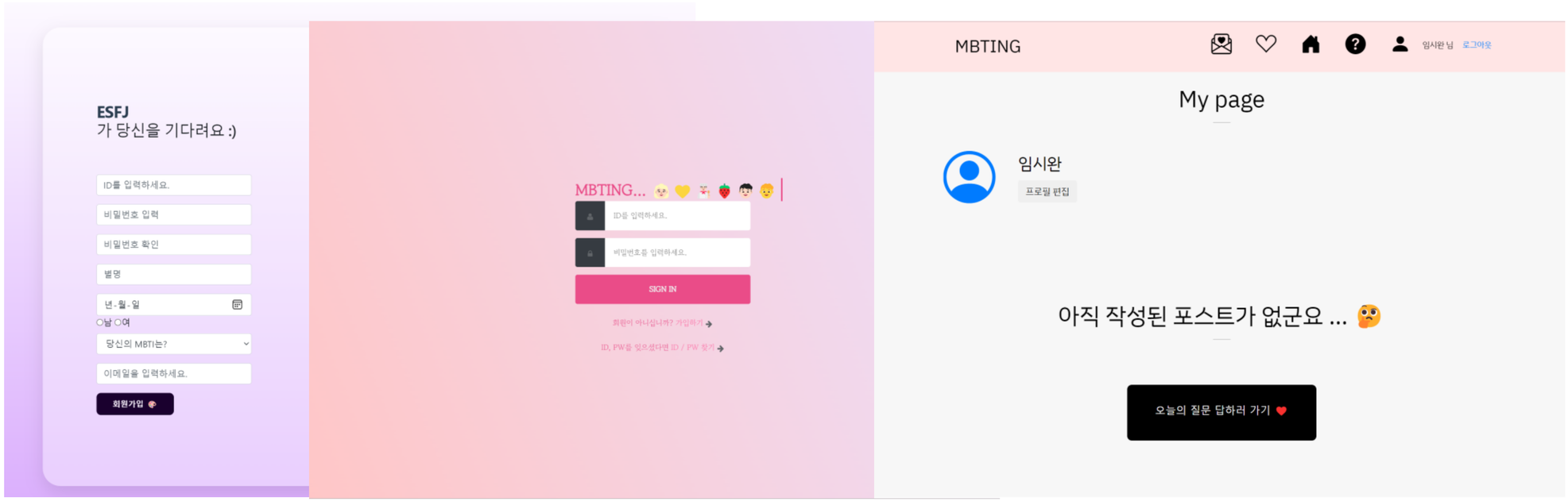


오픈 소스 사용 (코드펜) ▼



4. 장점

3. 디자인이 예쁘고 편리성이 좋다
직관적인 디스플레이를 제공함으로서 사용에 편리함을 제공했으며
트렌디한 디자인으로 눈이 가게끔 만들었다.



4. 장점

4. 많은 시도를 했다

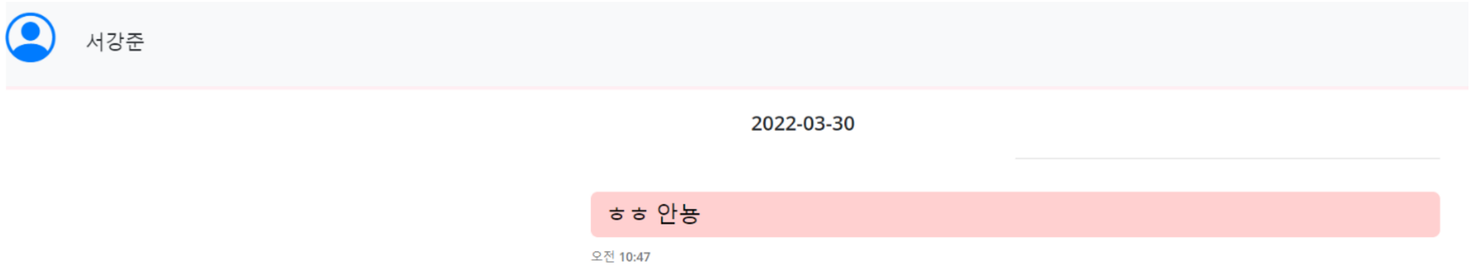
웹 사이트에서 흔히 쓰이는 '이메일로 ID, PW 찾기' 를 JavaMailSender를 이용해 구현하였다.

```
//비밀번호 찾기
@RequestMapping("/findPw")
public String sendMail(
        @RequestParam("email") String email
        ) throws MessagingException, UnsupportedEncodingException  {

    int result = userBO.EmailExist(email);
    if(result==1){
        Random r = new Random();
        int num = r.nextInt(999999);


        String setfrom = "say1890@naver.com";
        String tomail = email;
        String title = "[Mbting] 비밀번호 변경 인증 이메일입니다 ";
        String loginId = userBO.getUserByEmail(email);
        String password = Integer.toString(num);
        userBO.setPassword(loginId, password);

        String content = System.getProperty("line.separator") + "안녕하세요 회원님:)"
        + System.getProperty("line.separator")
        + "회원님의 id는 " + loginId +" 이고,"
        + "회원님의 변경된 pw는 " + num + " 입니다."
        + System.getProperty("line.separator");



        try {

            MimeMessage message = mailSender.createMimeMessage();
            MimeMessageHelper messageHelper = new MimeMessageHelper(message, true, "utf-8");

            messageHelper.setFrom(setfrom);
            messageHelper.setTo(tomail);
            messageHelper.setSubject(title);
            messageHelper.setText(content);

            mailSender.send(message);
        } catch (Exception e) {
            return (e.getMessage());
        }
```

4. 장점

4. 많은 시도를 했다

실시간으로 데이터를 주고 받을 수 있는 Flux를 이용하여 채팅 기능을 구현했다.




```
public interface ChatRepository extends ReactiveMongoRepository<Chat,String>{
    @Tailable // 커서를 안 닫고 계속 유지
    @Query("{ sender : ?0, receiver : ?1}")
    Flux<Chat> mFindBySender(String sender, String receiver); // Flux (흐름) response를 유지하면서 데이터를 계속 흘려보내기

    @Tailable
    @Query("{ roomNum: ?0 }")
    Flux<Chat> mFindByRoomNum(Integer roomNum);
}
```

4. 장점

5. 실제 서비스와 *비슷하게 구현*하려고 노력했다.

나는 소개팅 사이트에서 제일 중요한 부분은 *'매칭'* 이라고 생각한다.

그렇기에 MBTING를 만들면서 어떻게 하면 사용자가 하트를 눌렀을 때 '*매칭 될 가능성*'이 높을지,
매칭 결과에 대한 '만족도'를 높이려면 어떻게 해야 할지에 대해서 고민하며 쿼리와 java 코드를 짰다.
매칭 될 가능성이 높으려면 사용자가 상대를 마음에 들어하는 만큼 상대도 사용자를 마음에 들어해야 하기에, 사용자가 원하는 상대의 특성과 상대의 특성이 맞으면 점수를 주고,
그 반대로 상대가 원하는 사용자의 특성이 사용자의 특성과 일치할때도 점수를 줘서
'점수가 높은 순' 으로 화면에 띄우는 방식을 사용했다.

4. 장점

5-1 포인트 쌓이는 과정 (java code)

```
list = new ArrayList<>();
// 성격 비교 반복문 ( 사용자 기준 )
for (String i : myIdealCharacter) {
        for (String j : yourCharacter) {
                // 만약 내가 원하는 상대의 i번째 성격과 상대의 j번째 성격이 일치한다면
                if (i.equals(j)) {
                        point++;
                        if (list.size() < 3) {
                                list.add(j);
                        }
                        userDetail.setCharacter(list);

                }
        }
}

if (point >= 8) {

        userDetail.setUserCharacter(your);
        User user = userDAO.selectUserById(your.getUser_id());
        int ageForProfile = userDAO.getMyage(your.getUser_id());
        user.setAgeForProfile(ageForProfile);
        userDetail.setUser(user);
        userDetail.setPoint(point);
        boolean isLike = likeBO.isLike(userId, your.getUser_id());
        userDetail.setLike(isLike);
        MatchingList.add(userDetail);
}
```

4. 장점

5-2 포인트 높은 순으로 정렬

어떻게 하면 포인트가 높은 사용자가 그렇지 않은 사용자보다 더 먼저 보일 수 있을지에 대한 고민을 하다가 compareTo 메소드를 이용하여 해결했다.
이 과정에서 ArrayList안의 user 객체를 point 순으로 정렬하기 위해서,
Comparable interface를 implements 하고,
compareTo() 메소드를 override 했다.

```
/* 오늘의 추천 받아오기 */
// 사용자가 사용자 정보를 다 입력했을 경우 추천을 받아온다.

List<UserDetail> userList = userBO.getRecommendedUser(userId, sex);
Collections.sort(userList);
```

```
    @Override
    public int compareTo(UserDetail user) {
            if (user.point < point) {
                    return -1;
            } else if (user.point > point) {
                    return 1;
            }
            return 0;
    }
```

4. 장점

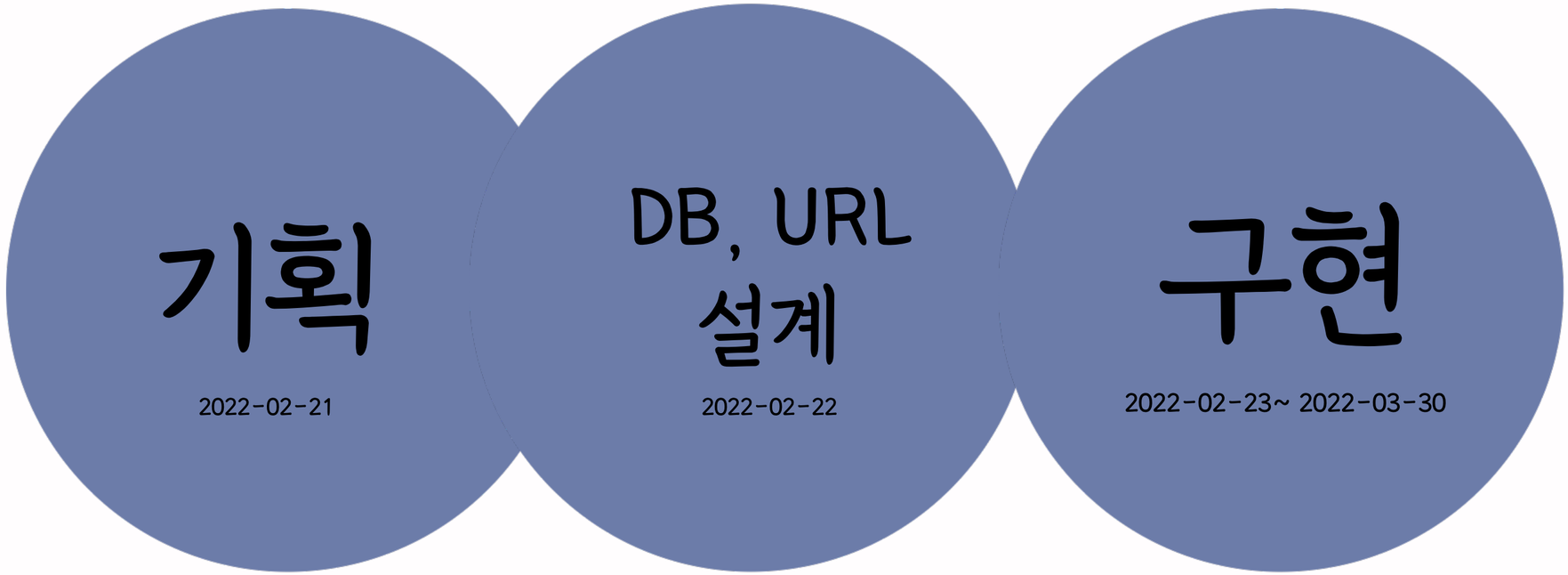
5-2 쿼리

또한, tracelogin이라는 테이블을 만들어 사용자가 로그인 했을때마다 접속 시간을 update 해줌으로서 활동하지 않는 사용자를 최대한 안 보여주게 설계했다.
또한 사용자가 싫어요를 보낸 사용자는 추천에 뜨지 않게끔 NOT IN 조건을 주었다.
또한, tracelogin이라는 테이블을 만들어 사용자가 로그인 했을때마다 접속 시간을 update 해줌으로서 활동하지 않는 사용자를 보이지 않게끔 설계했다.

```
WHERE

    sex != #{sex}

AND
    user.id  NOT IN(
    SELECT
            dislike.receiver
    FROM
            `dislike`
    WHERE
            dislike.sender = #{userId}
    )
AND
    year(current_date())-year(user.birthday)+1  ${chooseAge}  ${myage}

    ORDER by (select tracelogin.updatedAt from tracelogin where user.id = tracelogin.user_id) DESC
LIMIT 10;
    </select>
```

프로젝트 진행 단계

mbting



기획 mbting

페이지별 기획서 작성

https://ovenapp.io/project/rCTfNumClNAbDwremGYBe2sWkB4z63AE#YdGsG

메인 화면

받은 좋아요

설계

"각 컬럼의 타입, null 여부, auto_increment 여부, 설명 항목 등을 작성 했고, 이를 기반으로 쿼리를 썼습니다."

mbting

DATABASE 설계서 작성

https://docs.google.com/spreadsheets/d/15WWveTZaCIsB7vpDN0bKxUib1w3DehJRtV-59e1Dwx0/edit#gid=0

테이블 이름	user_character			
설명	사용자의 특성을 정리한 테이블			
컬럼명	**타입**	**NULL 가능 여부**	**auto_increment**	**설명**
id	int	X	O	primary key
user_id	int	X	X	user의 primary key
myMerit	varchar(100)	O	X	사용자의 장점
myHobby	varchar(100)	O	X	사용자의 취미
myCharacter	varchar(100)	O	X	사용자의 성격
yourMerit	varchar(100)	O	X	이상형의 장점
yourHobby	varchar(100)	O	X	이상형의 취미
yourCharacter	varchar(100)	O	X	이상형의 성격
age	varchar(6)	O	X	원하는 나잇대
updatedAt	timestamp	X	X	수정 날짜

설계

mbting

"VIEW와 API를 나눠서 설계 했습니다."

URL 설계서 작성 (VIEW)

https://docs.google.com/spreadsheets/d/15WWveTZaCIsB7vpDN0bKxUib1w3DehJRtV-59e1Dwx0/edit#gid=0

View URL

제목	url	parameter	설명
로그인	/user/signin_view		로그인 화면
회원 가입	/user/signup_view		회원가입 화면
ID / PW 찾기	/user/find_info_view		ID / PW 찾기 화면
메인 화면	/post/main		메인 화면
프로필 보기	/user/see_profile?userId=${userId}	userId	추천에 뜬 사용자의 프로필을 볼 수 있는 화면
채팅 리스트	/chattingList		채팅 리스트
좋아요	/like/see_like_view		받은 좋아요, 보낸 좋아요를 볼 수 있는 화면
마이페이지	/user/mypage_view		오늘의 질문에 답 한 내역, 프로필 편집으로 이어지는 버튼을 포함
회원 정보 수정	/user/mypage_edit_view		나의 정보를 수정할 수 있는 화면

설계

mbting

"VIEW와 API를 나눠서 설계 했습니다."

URL 설계서 작성 (API)

API URL				
1) 로그인				
	url : /user/sign_in			
	method: post			
	parameter			
	파라미터 명	데이터 타입	NULL 여부	설명
	loginId	String	N	로그인 아이디
	password	String	N	로그인 비밀 번호
	응답값(json)			
	성공시	{"result" : "success"}		
	실패시	{"result" : "fail"}		

THANK YOU°

CONTACT

010-5645-7352
say1890@naver.com