게임메이트

3팀 | 박원정 고의성 김경래 김연지 김이삭 조한휘

게임메이트

- 도메인 주소 :
http://ec2-54-82-46-230.compute-1.amazonaws.com/

- 테스트 계정
id : mate@elice.com
password : a123456789!

CONTENTS

01.
프로젝트 개요

프로젝트 개요

(1) 프로젝트 주제 및 선정 배경, 기획의도

🎮 게임 친구를 구할 수 있는 플랫폼이 없을까 ?

토픽 > 게임

게임친구는 어디서구해?

삼양사 · j*********

2022.02.17. 243 39

나 게임좋아하는데 휴무날 게임같이할사람이 음서
한동안 게임접었다가 다시좀 해볼라하는데
왠만한게임 다 하고 스팀겜 같은것도 함

0 게임 같이할 사람 구하는곳이있나요?

ZTUF3 | 7 871

게임 같은거 같이할 파티원 구하고싶은데
어디서 모집이나 구하는곳이 있을까요..
ㅠ.ㅠ2~4이서 같이 하면 너무 재밌고 행복할거같아요

특정 겜 같이 할 사람 어디서 구함?

3841727b 2022.01.05 1932

https://www.dogdrip.net/374792011

It Takes Two, Trine, 포탈2 , ASTRONEER
이런게임들;;

반드시 둘이서 해야하거나
혼자할순 없지만 둘이서 하면 꿀잼예상되거나 하는겜들
따로 듀오 구하는곳 있음?

잇테잌투는 신작이라 할친구 있는데
옛날게임들이나 많이 안유명한 수작게임들은...

기존 플랫폼보다 '게임 친구 추천'에 포커스가 맞춰진 플랫폼

프로젝트 개요

(1) 프로젝트 주제 및 선정 배경, 기획의도

🎮 '나와 잘 맞는' 게임 친구를 구할 수 있는 플랫폼이 없을까 ?

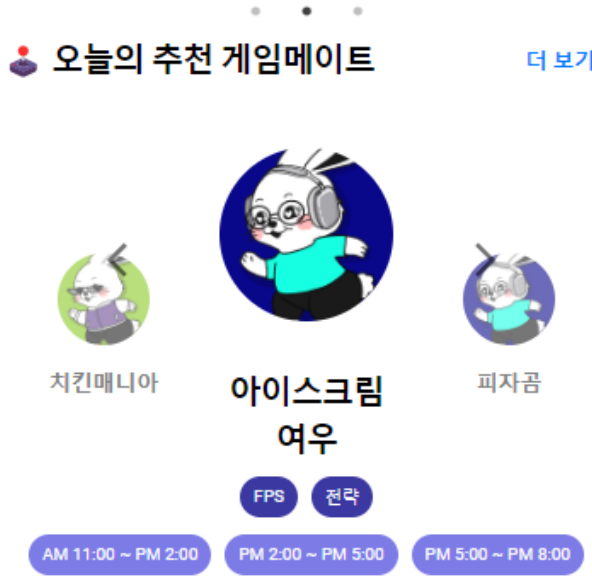
이름	김게임	취미	게임하기
나이	24세	Pain point	주변에 게임을 같이할만한 친구가 없다
성별	남	Needs	게임 취향이 맞는 사람과 같이 게임을 플레이하고 싶음
직업	학생		

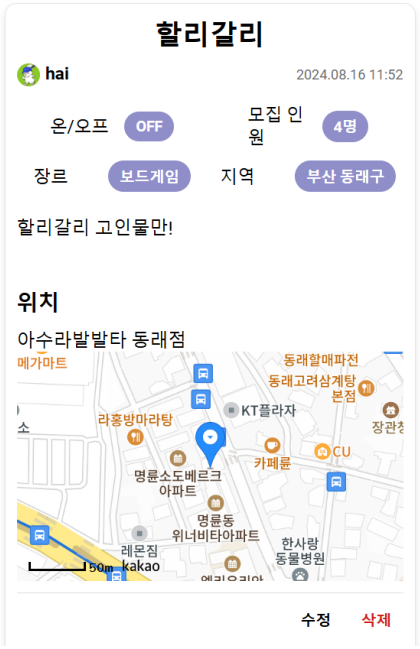
선호 게임 장르, 시간대 별로 게임 친구를 추천해줬으면 좋겠는데!

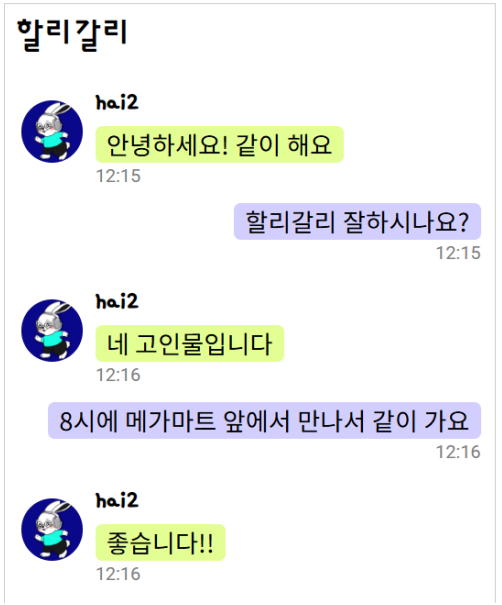
프로젝트 개요

(1) 프로젝트 주제 및 선정 배경, 기획의도

나와 맞는 게임 친구를 찾을 수 있는 게임 플랫폼 **게임메이트**







ML기반 친구추천

온라인 / 오프라인 게임

커뮤니티 중심

프로젝트 개요

(2) 프로젝트 내용

- Logo

게임메이트

- Main Color

#0A088A

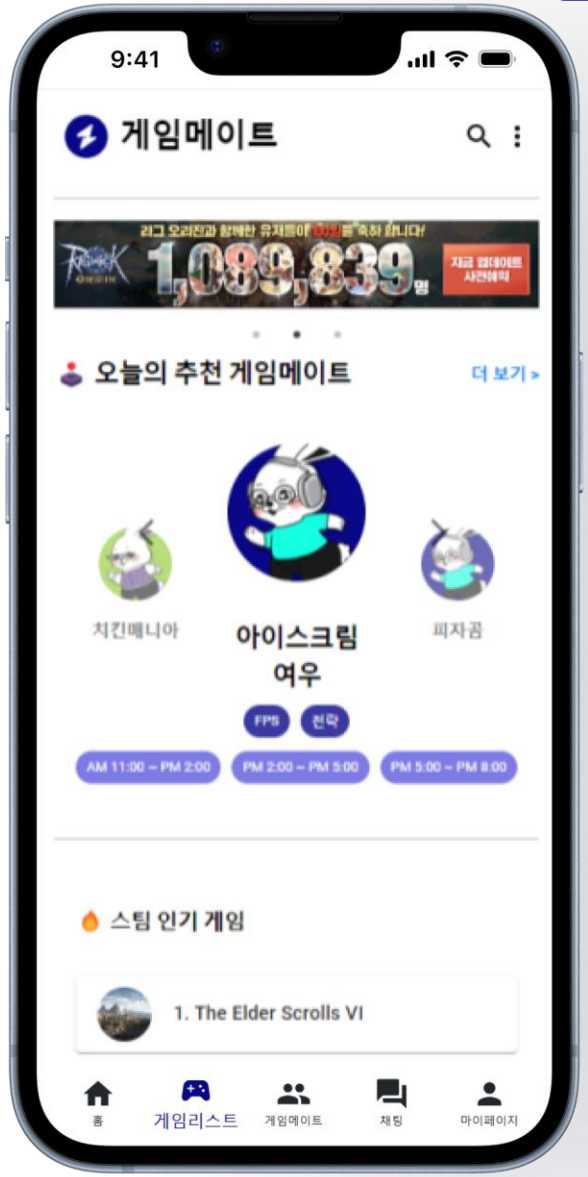
페이지 **포인트 컬러** 통일

- Concept

- 소통, **커뮤니티 중심**의 게임 정보 플랫폼

- Function

- **머신러닝** 기반으로 유저 추천 서비스를 구현
- Spring Boot, React 기반의 **모바일 / 웹 서비스** 구현



프로젝트 개요

(3) 활용 장비 및 재료

기술스택

Back

Front

커뮤니케이션

기획

프로젝트 개요

(4) 프로젝트 구조

Back

```
main
└── java
    └── com
        └── example
            └── gamemate
                ├── domain
                │   ├── auth          # 인증 관련 코드
                │   ├── chat          # 채팅 관련 코드
                │   ├── friend        # 친구 관리 코드
                │   ├── game          # 게임 관련 코드
                │   ├── post          # 게시물 관련 코드
                │   ├── s3            # AWS S3 관련 코드
                │   └── user          # 사용자 관리 코드
                └── global
                    ├── apiRes        # API 응답 관련 코드
                    ├── audit         # 감사 로그 코드
                    ├── common        # 공통 유틸리티 코드
                    ├── config        # 설정 관련 코드
                    ├── converter     # 데이터 변환 코드
                    ├── dataloader    # 데이터 로딩 관련 코드
                    ├── exception     # 예외 처리 코드
                    ├── interceptor   # 인터셉터 코드
                    └── listener      # 이벤트 리스너 코드
```

Front

```
src
├── apis
├── assets
├── components
├── hooks
├── layouts
├── pages
├── utils
├── App.css
├── App.js
├── App.test.js
├── axiosInstance.js
├── index.css
├── index.js
├── logo.svg
├── reportWebVitals.js
└── setupTests.js
```

프로젝트 개요

(5) 활용방안 및 기대 효과

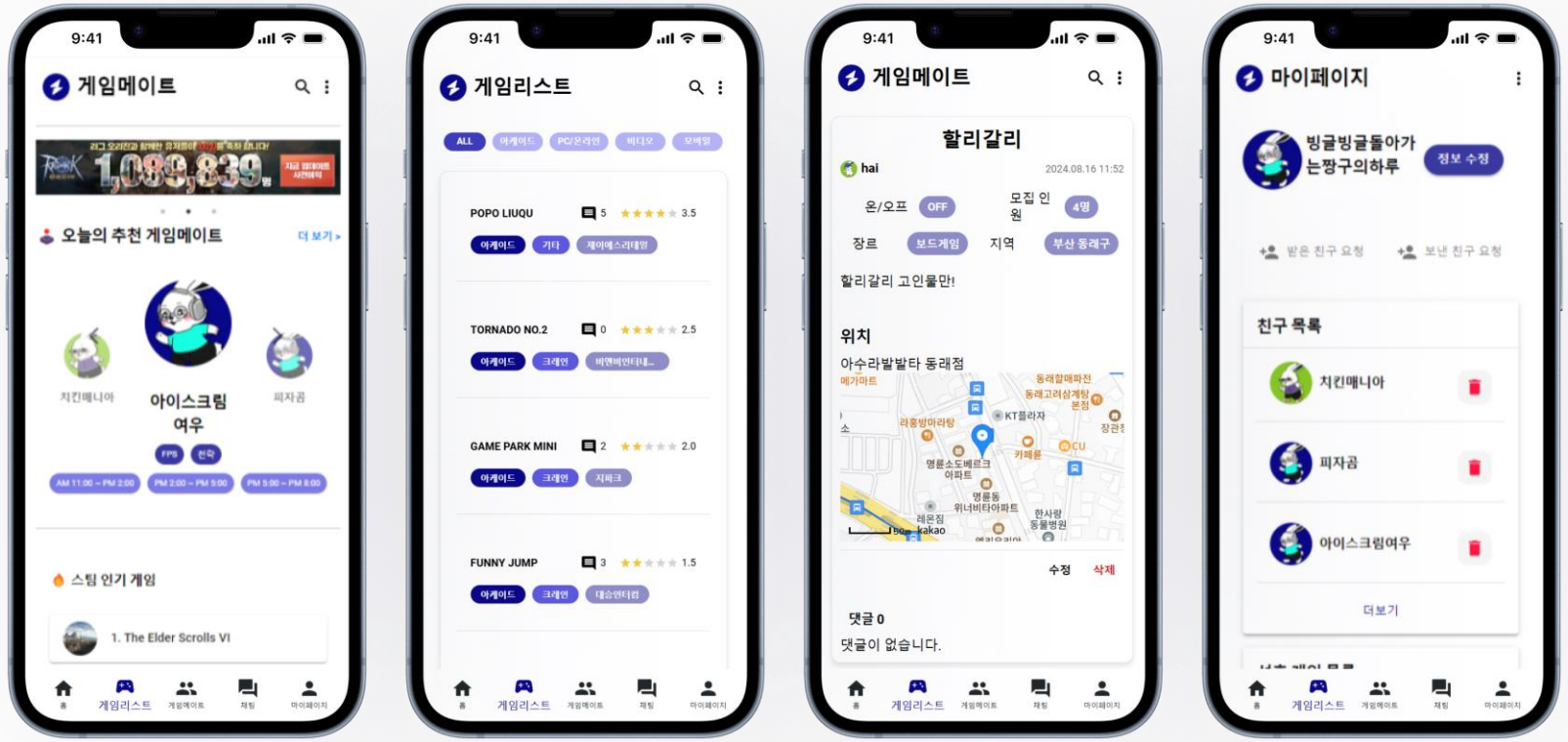
USER

게임 커뮤니티 활성화

PLATFORM

광고 및 마케팅 기회

데이터 활용 가능성



02.

프로젝트 팀 구성 및 역할

프로젝트 팀 구성 및 역할

(1) 팀 소개

주종현 BE 코치 — **박인수 FE 코치**

- 주 2회 이상 오피스 아워를 통한 **코드 리뷰, 피드백**

박원정 팀장

- **머신러닝** 모델 및 전용 서버 구축
- **유저 관계** 시스템 구현

고의성 팀원

- **회원가입** 기능
- **마이페이지** 기능 (BE,FE)

김경래 팀원

- Amazon s3을 이용 프로필 이미지 설정
- Google analytics 적용

김연지 팀원

- **게시글** 기능 (BE,FE)
- **BE 공통 파일** 작성

김이삭 팀원

- **웹소켓**과 **스톰프**를 사용하여 **실시간 채팅** 구현

조한휘 팀원

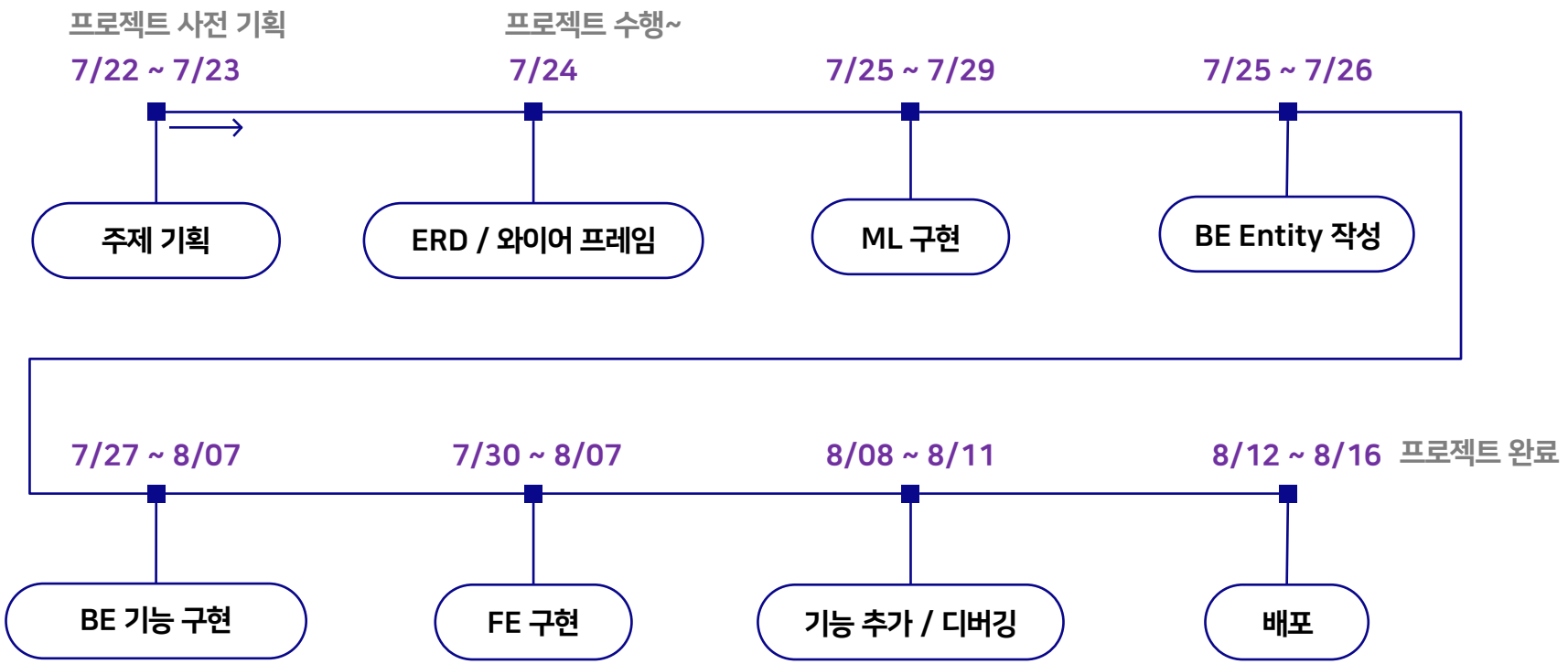
- **게임리스트** 기능 (BE,FE)
- **FE 레이아웃**

03.

프로젝트 수행 절차 및 방법

프로젝트 수행 절차 및 방법

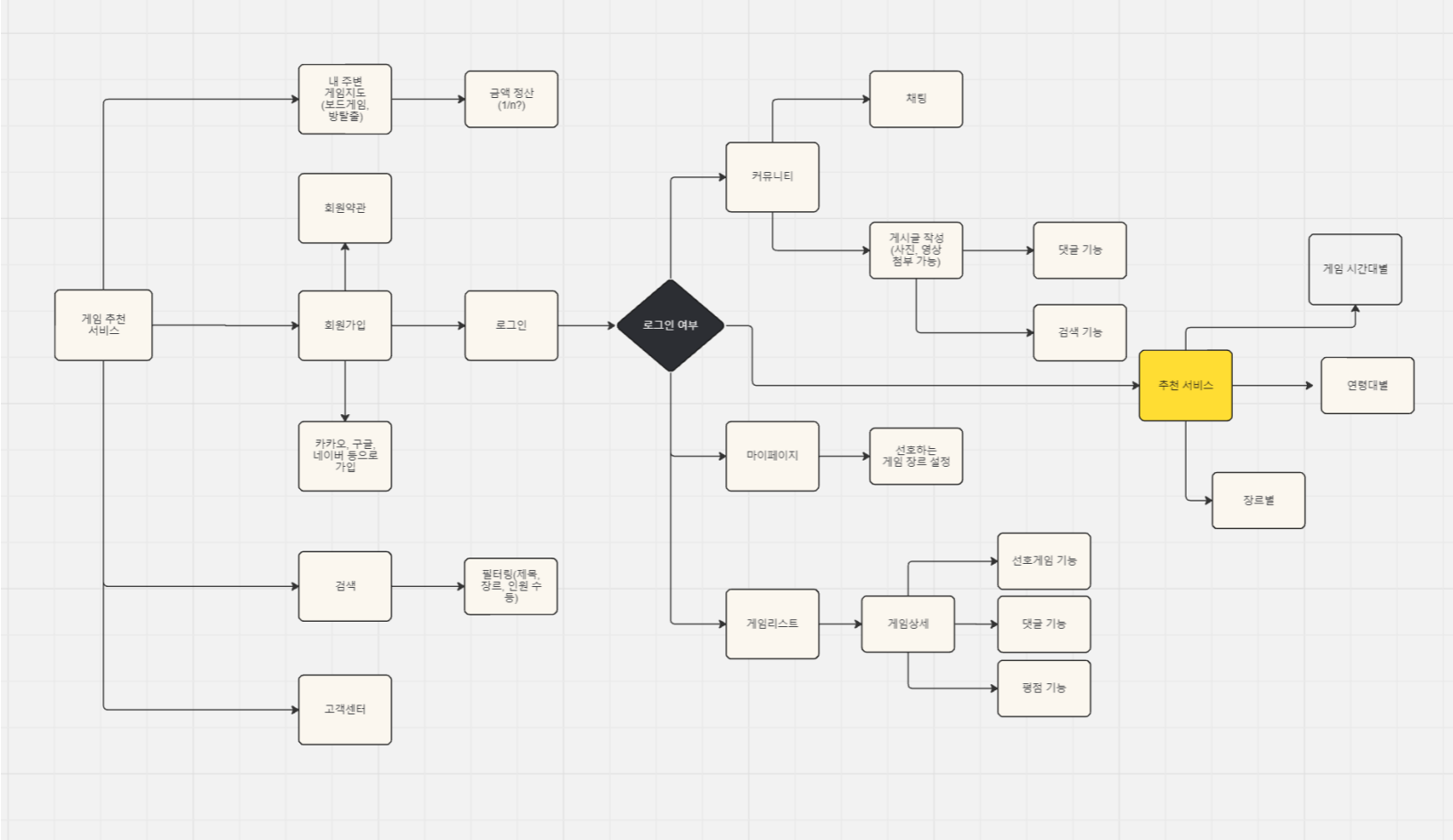
(1) 타임라인



총 개발 기간 : 7/22 ~ 8/16

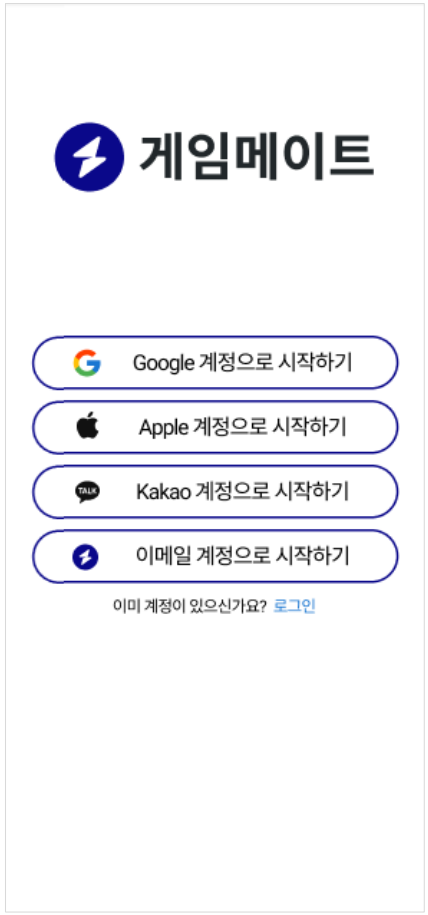
프로젝트 수행 절차 및 방법

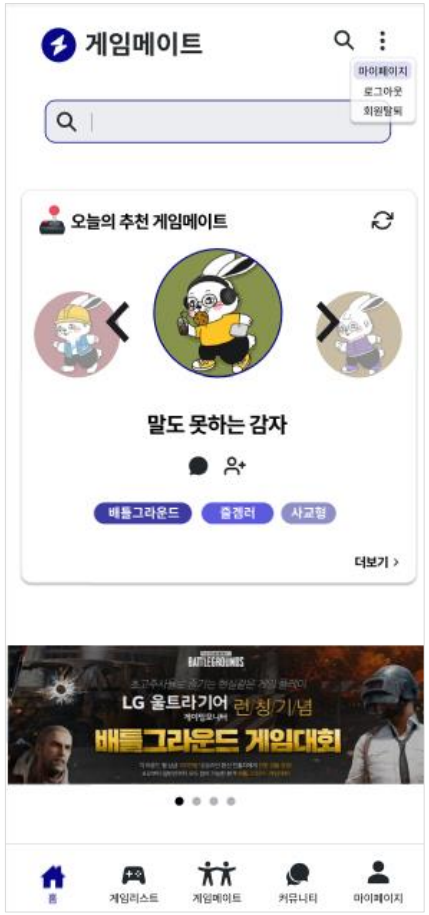
(2) 플로우 차트 설계

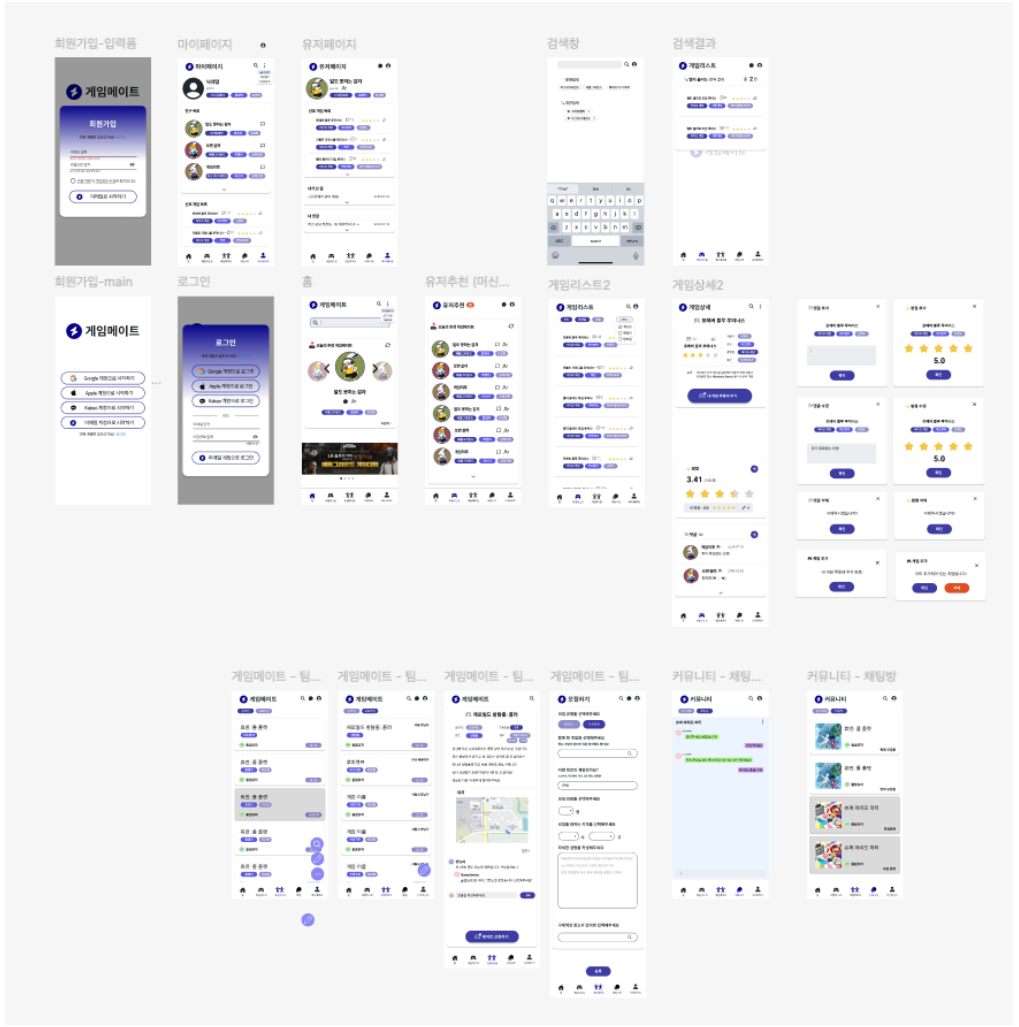


프로젝트 수행 절차 및 방법

(3) 피그마 설계

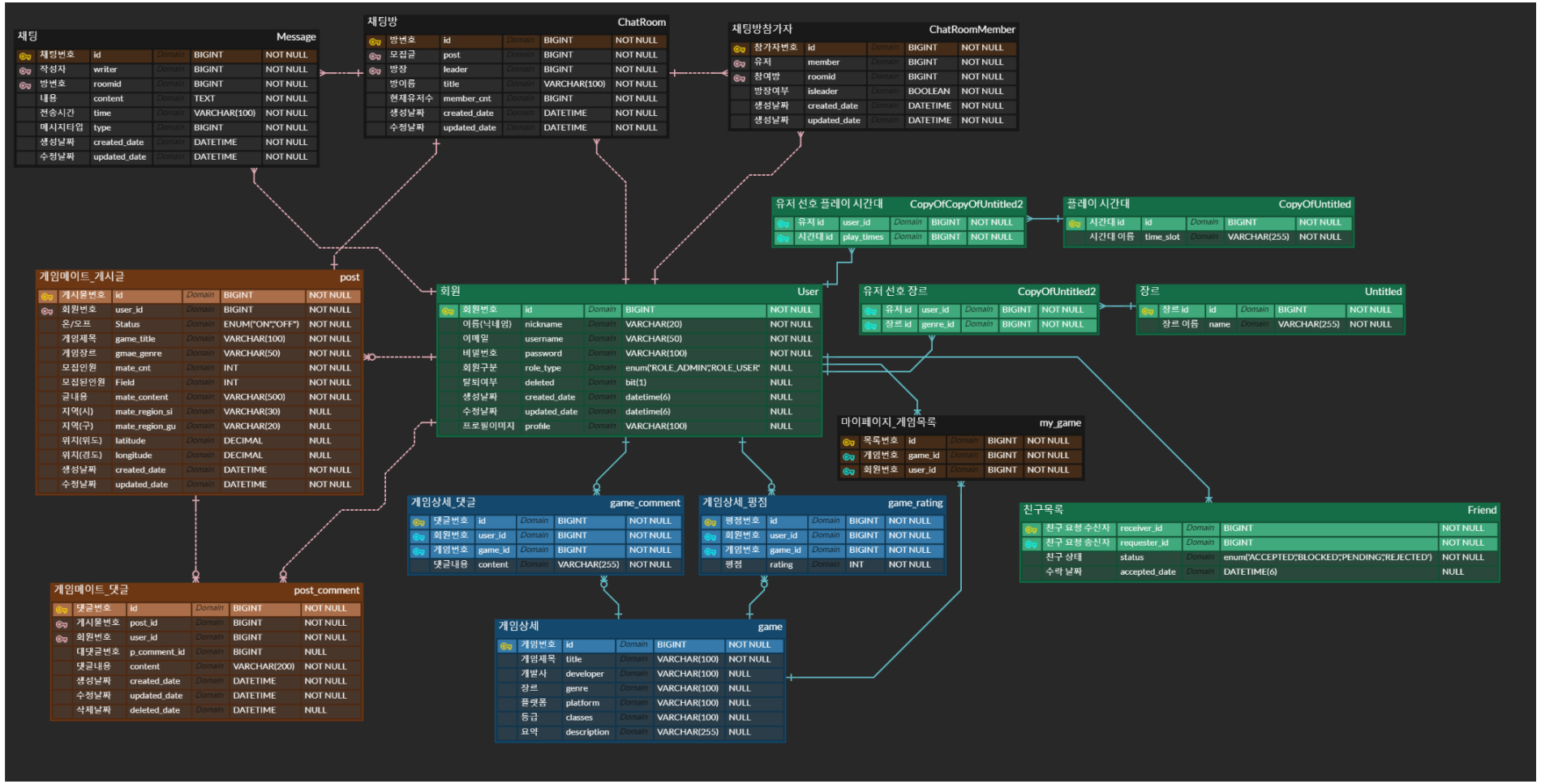






프로젝트 수행 절차 및 방법

(4) ERD 설계



프로젝트 수행 절차 및 방법

(5) Swagger

user-controller

PUT	/update
PUT	/restoreUser/{username}
POST	/profile/update
GET	/user
GET	/profile
GET	/mypage
GET	/info
GET	/delete

auth-controller

POST	/join

Friend 친구 관리 API

PUT	/friend/respond
PUT	/friend/cancel
GET	/friend/
POST	/friend/
GET	/friend/sent-requests
GET	/friend/received-requests
DELETE	/friend/{friendId}

프로젝트 수행 절차 및 방법

(5) Swagger

chat-room-member-controller

POST	/chat/addmember
DELETE	/chat/deletemember/{roomId}

message-controller

GET	/message/{roomId}/{messageId}
DELETE	/message/{id}

chat-room-controller

GET	/chat/{roomId}
DELETE	/chat/{roomId}
GET	/chat/post/{postId}
GET	/chat/

프로젝트 수행 절차 및 방법

(5) Swagger

Post 게임메이트 모집 게시글 API

GET	/posts/{id}
PUT	/posts/{id}
DELETE	/posts/{id}
GET	/posts
POST	/posts
GET	/posts/user

Post Comment 게임메이트 모집 글의 댓글 API

PUT	/post/{postId}/comment/{commentId}
DELETE	/post/{postId}/comment/{commentId}
GET	/post/{id}/comment
POST	/post/{id}/comment

프로젝트 수행 절차 및 방법

(5) Swagger

Game Game API

- GET /games/{id}
- PUT /games/{id}
- DELETE /games/{id}
- GET /games
- POST /games
- GET /fetch-games
- GET /fetch-and-convert-games

Game Rating Game Rating API

- GET /games/{gameId}/ratings
- PUT /games/{gameId}/ratings
- POST /games/{gameId}/ratings
- DELETE /games/{gameId}/ratings

Game Comment Game Comment API

- PUT /games/{gameId}/comments/{commentId}
- DELETE /games/{gameId}/comments/{commentId}
- GET /games/{gameId}/comments
- POST /games/{gameId}/comments

My Game My Game API

- POST /games/my-games/{gameId}
- DELETE /games/my-games/{gameId}
- GET /games/my-games
- GET /games/my-games/{gameId}/exists

04.

프로젝트 수행 경과

프로젝트 수행 경과

(1) 머신러닝

- 모델 구현

```
[ ]  # 추천 시스템 평가
     precision, recall, f1 = evaluate_recommendations(knn, test_features, test_indices, df, genres, times, similarity_threshold)

     print(f"정밀도: {precision:.2f}")
     print(f"재현율: {recall:.2f}")
     print(f"F1 점수: {f1:.2f}")
```

```
정밀도: 0.93
재현율: 1.00
F1 점수: 0.97
```

- Scikit-learn 라이브러리 내 **KNN 사용**
- 약 200개의 데이터를 활용하여 **모델 학습** 진행
- **예측 성능 93% 정확도**인 것 확인 완료

- 서버 구현

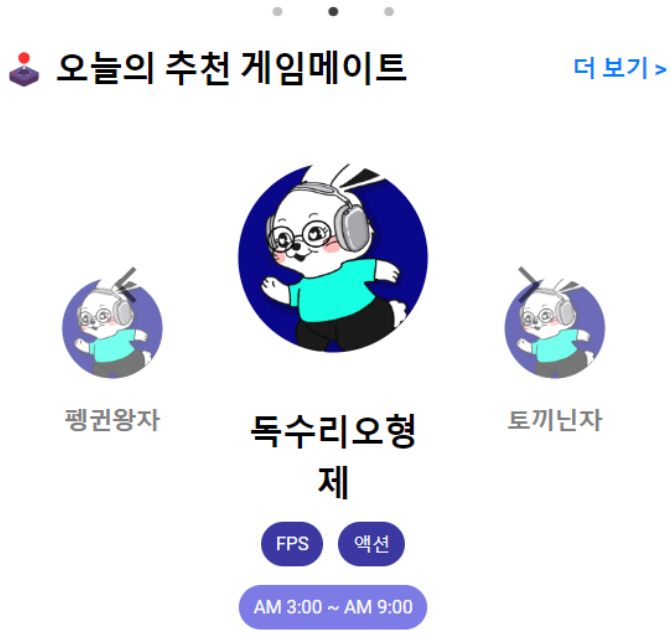
```
@app.post("/recommendation")
async def predict(user_features: UserFeatures, db: Session = Depends(get_db)):
    try:
        user_data = np.hstack((user_features.preferred_genres, user_features.play_times)).reshape(1, -1)
        distances, indices = model.kneighbors(user_data)

        similar_users = indices[0][1:]
        results = []

        for idx in similar_users:
            user = db.query(User).filter(User.id == idx + 1).first()
            if user:
                user_genres = [1 if genre.id in [g.id for g in user.preferred_genres] else 0 for genre in
                               db.query(Genre).all()]
                user_playtimes = [1 if playtime.id in [p.id for p in user.play_times] else 0 for playtime in
                                  db.query(PlayTime).all()]

                common_genres = get_common_elements(user_features.preferred_genres, user_genres, genres_list)
                common_times = get_common_elements(user_features.play_times, user_playtimes, times_list)

                user_info = {
                    "id": user.id,
                    "recommend_user": user.nickname,
                    "user_profile": user.user_profile,
                    "common_genre": common_genres,
                    "common_play_time": common_times
                }
                results.append(user_info)
            else:
                print(f"User with id {idx + 1} not found")

        return {"similar_users": results}
    except Exception as e:
        print(f"Error in recommendation: {str(e)}")
        return {"error": str(e)}
```

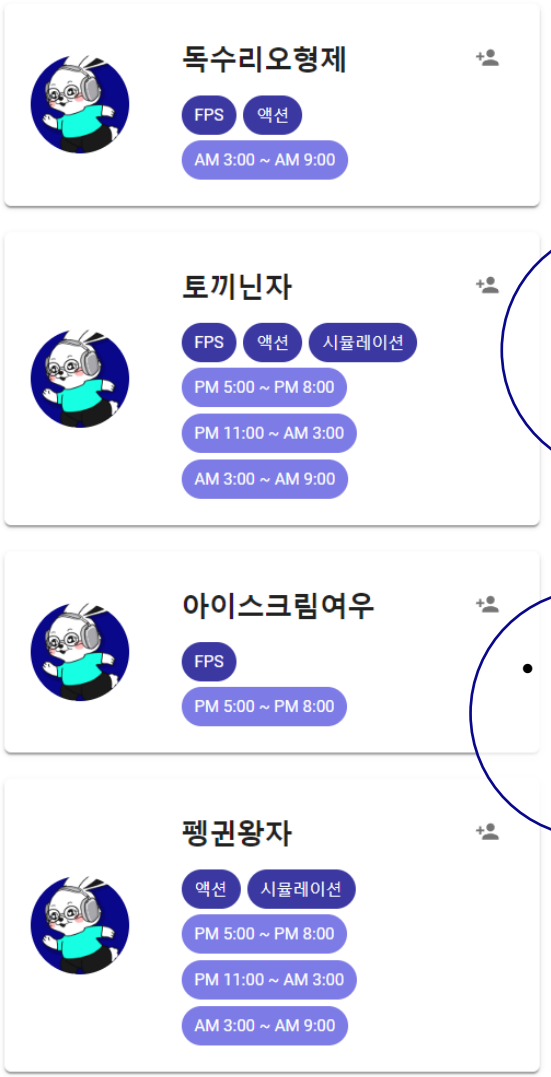
- FastAPI 프레임워크사용
- SQLAlchemy를 사용하여 **유저 데이터베이스와 연결**
- **uvicorn**을 사용하여 서버 실행

프로젝트 수행 경과

(1) 머신러닝

- 유저 추천





- 프론트엔드에서 로그인 후 **토큰 발급** -> 스프링 API에서 토큰에서 **유저 정보를 추출**, 해당 유저 정보 중 선호 게임 장르와 플레이 시간대를 **파이썬 서버로 전송**

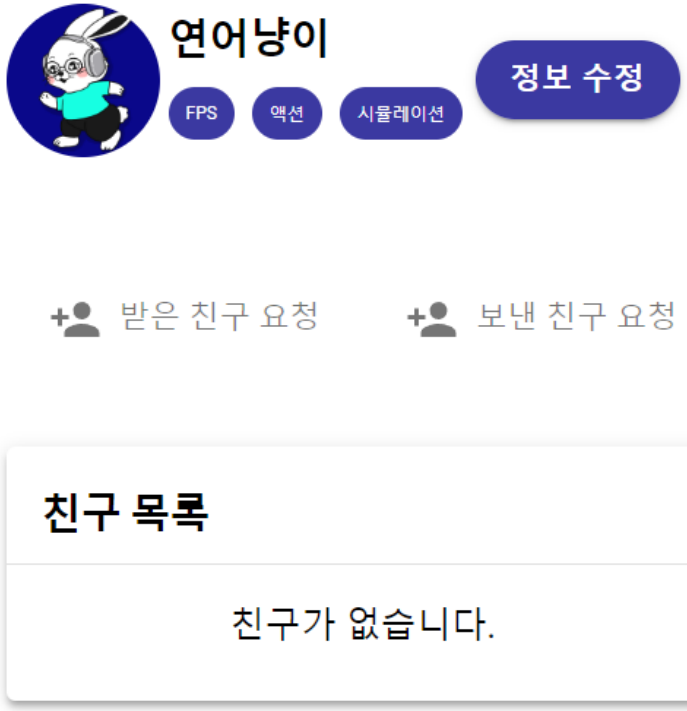
- 전송받은 **정보를 벡터화**하여 머신러닝 모델에서 데이터베이스에 저장된 유저의 정보 중 벡터화 된 정보와 코사인 **유사도가 높은 유저를 4명까지 추출하여 프론트엔드로 전송**

프로젝트 수행 경과

(2) 친구 기능

- 유저 추천

```
62 usages  Wonjeong Park
@Entity
@Table(name = "friend")
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@IdClass(FriendId.class)
public class Friend {

    @Id
    @ManyToOne
    @JoinColumn(name = "requester_id", nullable = false)
    private User requester;

    @Id
    @ManyToOne
    @JoinColumn(name = "receiver_id", nullable = false)
    private User receiver;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false)
    private Status status;

    @Column(name = "accepted_date")
    private LocalDateTime acceptedDate;

    29 usages  Wonjeong Park
    public enum Status {
        7 usages
        PENDING, ACCEPTED, REJECTED, CANCELED
    }
}
```



- 친구 관련 데이터베이스를 **친구 요청 수신자, 친구 요청 송신자, 친구 상태** 컬럼을 기반으로 저장
- 친구 요청 시 **친구 상태**를 PENDING, 친구 요청 수락 시 ACCEPTED, 친구 요청 거절 시 DECLINED 등으로 설정하여 관리

- 제공된 추천 유저 목록에서 친구 요청을 전송할 수 있고, 게임메이트 게시글 내 댓글을 통해 서로 친구 요청을 전송할 수 있음
- **마이페이지**에서 받은 친구요청과 보낸 **친구 요청을 관리**할 수 있으며, 수락한 경우 친구 목록에 추가됨

프로젝트 수행 경과

(3) 회원가입

- AuthController

```
no usages  ▲ [Cloud3] 고의성
@PostMapping("/join")
public String joinProcess(@RequestBody JoinDTO joinDTO) {

    log.info(joinDTO.getUsername());
    authService.joinProcess(joinDTO);

    return "ok";

}

no usages  ▲ [Cloud3] 고의성
@GetMapping("/check-availability")
public ResponseEntity<Map<String, String>> checkAvailability(
    @RequestParam("username") String username,
    @RequestParam("nickname") String nickname) {

    Map<String, String> response = new HashMap<>();

    boolean isUsernameExist = authService.isEmailDuplicated(username);
    boolean isNicknameExist = authService.isNicknameDuplicated(nickname);

    if (isUsernameExist) {
        response.put("emailError", "이미 존재하는 이메일입니다.");
    }

    if (isNicknameExist) {
        response.put("nicknameError", "이미 존재하는 닉네임입니다.");
    }

    if (response.isEmpty()) {
        return ResponseEntity.ok(response);
    }

    return ResponseEntity.badRequest().body(response);
}
```

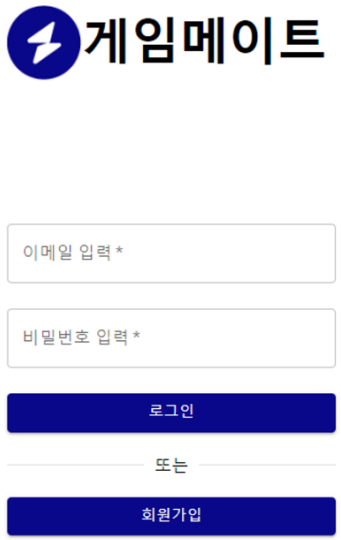
- AuthService

```
1 usage  ▲ Wonkeong Park *)
public void joinProcess(JoinDTO joinDTO) {

    if(isEmailDuplicated(joinDTO.getUsername())) {
        throw new IllegalArgumentException("이미 존재하는 이메일입니다.");
    }

    List<Integer> preferredGenres = joinDTO.getPreferredGenres();
    List<Integer> playTimes = joinDTO.getPlayTimes();

    User data = new User();

    data.setUsername(joinDTO.getUsername());
    data.setPassword(bCryptPasswordEncoder.encode(joinDTO.getPassword()));
    data.setNickname(joinDTO.getNickname());

    List<Genre> genres = IntStream.range(0, preferredGenres.size()) IntStream
            .filter(i -> preferredGenres.get(i) == 1)
            .mapToObj(i -> genreRepository.findById((long) (i + 1)).orElseThrow(() -> new IllegalArgumentException("Invalid genre ID: " + (i + 1)))) Stream<Genre>
            .collect(Collectors.toList());
    data.setPreferredGenres(genres);

    List<PlayTime> times = IntStream.range(0, playTimes.size()) IntStream
            .filter(i -> playTimes.get(i) == 1)
            .mapToObj(i -> playTimeRepository.findById((long) (i + 1)).orElseThrow(() -> new IllegalArgumentException("Invalid play time ID: " + (i + 1)))) Stream<PlayTime>
            .collect(Collectors.toList());
    data.setPlayTimes(times);

    userRepository.save(data);

}

2 usages  ▲ [Cloud3] 고의성
public boolean isEmailDuplicated(String username) {
    return userRepository.existsByUsername(username);
}

1 usage  ▲ [Cloud3] 고의성
public boolean isNicknameDuplicated(String nickname) {
    return userRepository.existsByNickname(nickname);
}
```

- **중복 이메일 가입 불가, 비밀번호 불일치 확인** 등 조건을 만족하면 가입 완료
- DTO에 담긴 값들을 통해 회원가입 진행
- 이메일/닉네임 중복 발생 시 그에 해당하는 정보들을 **ResponseEntity의 body**에 담아 리턴
- **Service단에서 비밀번호를 암호화**해서 실제 데이터베이스에 저장

프로젝트 수행 경과

(3) 회원가입




```
handleLoginAttempt(loginDto);
UsernamePasswordAuthenticationToken authToken =
    new UsernamePasswordAuthenticationToken(
        loginDto.getUsername(),
        loginDto.getPassword(),
        authorities: null
    );
return authenticationManager.authenticate(authToken);
```

로그인 페이지에서 입력한 이메일과 비밀번호를 받아옴

```
public class CustomUserDetailsService implements UserDetailsService {

    2 usages
    private final UserRepository userRepository;

    no usages  ▲ [Cloud3] 고의성
    public CustomUserDetailsService(UserRepository userRepository) {

        this.userRepository = userRepository;

    }

    2 usages  ▲ [Cloud3] 고의성
    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

        //DB에서 조회
        User userData = userRepository.findByUsername(username);

        if(userData != null) {

            //UserDetails에 담아 return하면 AuthenticationManager가 검증
            return new CustomUserDetailsDTO(userData);

        }

        return null;

    }
```

CustomUserDetailsService를 통해
CustomUserDetailsDTO에 담아
AuthenticationManager를 통해 내부에 id, pw 값을
던져 내부적으로 **로그인 검증**

프로젝트 수행 경과

(3) 회원가입

프론트에서 로그인 요청 후 정상적으로 로그인이 되면 **토큰을 쿠키에 저장**

```
//로그인 성공 시 실행하는 메소드(여기서 JWT 발급)
no usages  ▲ [Cloud3] 고의성
@Override
protected void successfulAuthentication(HttpServletRequest request, HttpServletResponse response, FilterChain chain, Authentication authentication)
        throws IOException {

    //UserDetails
    CustomUserDetailsDTO customUserDetails = (CustomUserDetailsDTO) authentication.getPrincipal();

    Long id = customUserDetails.getId();
    String username = customUserDetails.getUsername();

    Collection<? extends GrantedAuthority> authorities = authentication.getAuthorities();
    Iterator<? extends GrantedAuthority> iterator = authorities.iterator();
    GrantedAuthority auth = iterator.next();

    String role = auth.getAuthority();

    // JWT 생성
    String token = jwtUtil.createJwt(id, username, role, auth.getAuthority(), expirationTime);

    response.addHeader( s: "Authorization",  s1: "Bearer " + token);

}
```

검증 성공 시 successfulAuth를 통해 **JWTUtil에서 토큰을 만들어 응답**

```
useEffect(() => {
    // 쿠키에 토큰이 없으면 로그인 페이지로 이동
    if (!cookies.token) {
        navigate('/auth');
    }

    const fetchUserData = async () => {
        try {
            const response = await axios.get('/mypage', {
                headers: {
                    Authorization: cookies.token,
                },
            });

            if (response.status === 200) {
                setUser(response.data); // 사용자 정보를 상태에 저장
                setEditedUser({
                    nickname: response.data.nickname,
                    password: '' // 비밀번호 초기화
                });

                fetchUserPosts(response.data.id); // 사용자 ID를 인자로 전달하여 사용자가 작성한 글 목록을 가져옴
            }
        } catch (error) {
            console.error('사용자 정보를 가져오는 데 실패했습니다:', error);
        }
    };
```

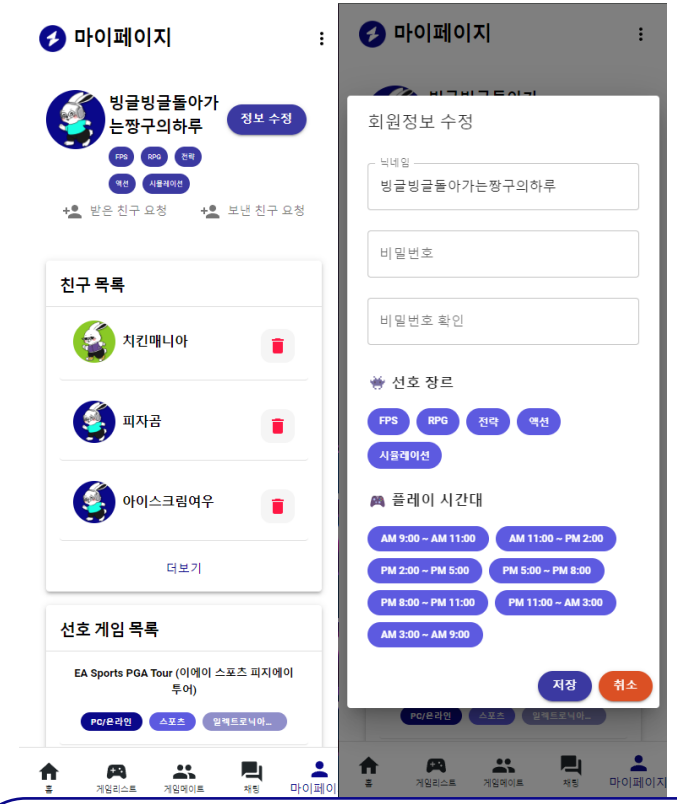
Cookie Value ☐ Show URL-decoded

Bearer%20eyJhbGciOiJIUzI1NiJ9.eyJpZCI6MSwidXNlcm5hbWUiOiJqamFuZzlAdGVzdC5jb20iLCJyb2xlIjoiUk9MRV9VU0VSIiwiaWF0IjoxNzIzNzY3OTYzLCJleHAiOjE3MjczNjc5NjN9.AFnkRa04goXKcJ6xcOKJOEls5N4KnBRjKOfNiDhZXoA

프로젝트 수행 경과

(4) 마이페이지

- 회원 정보 수정



친구 목록, 선호 게임 목록, 내가 쓴 글 목록 확인 가능

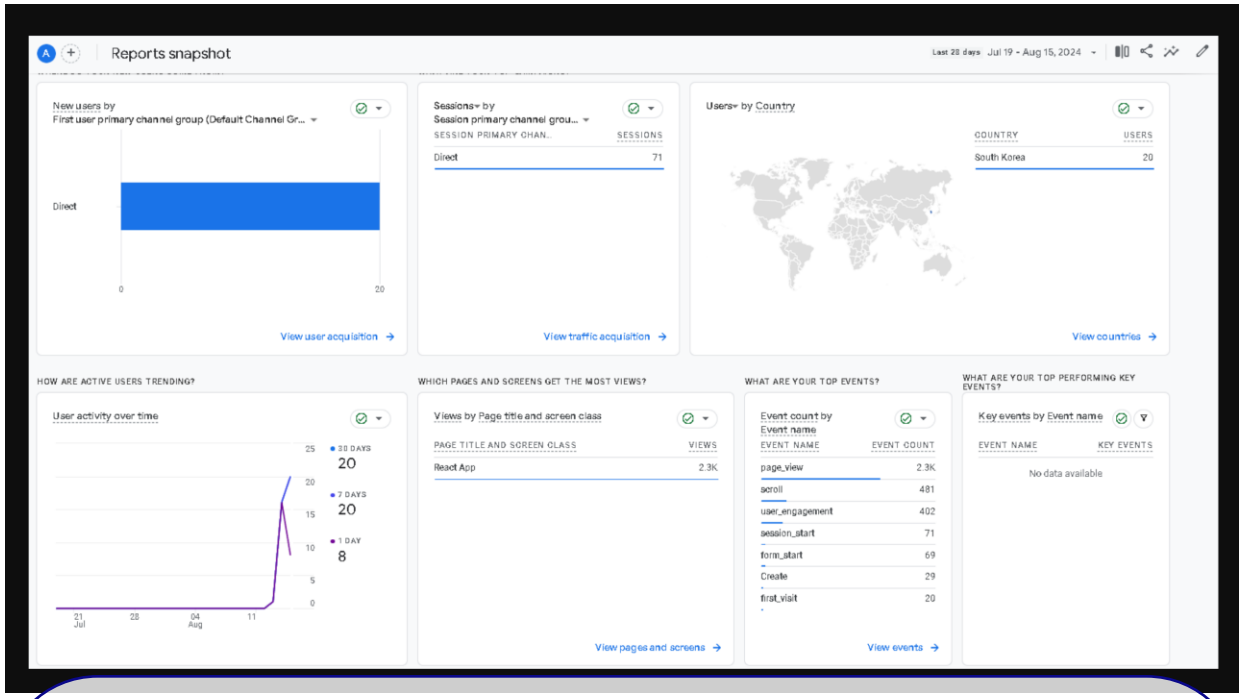
- 회원 탈퇴

```java
no usages  ± [Cloud3] 고의성
@GetMapping("/delete")
public ResponseEntity<String> deleteByName(@RequestParam("username") String username) {
    // 로그 추가
    log.info("deleteCheck");

    //게시글 삭제
    postService.deletePostsByUsername(username);
    log.info("deletePostsByUsername");

    // 댓글 삭제
    gameCommentService.deleteCommentsByUsername(username);
    log.info("deleteCommentsByUsername");

    // 평점 삭제
    gameRatingService.deleteUserRatingsByUsername(username);
    log.info("deleteUserRatingsByUsername");

    //친구 관계 삭제
    friendService.deleteUserRelatedFriend(username);
    log.info("deleteUserRelatedFriend");

    // 유저 삭제
    userService.deletedByUsername(username);
    log.info("deletedByUsername");

    return ResponseEntity.ok( body: "User deleted successfully");
}
```

- 소프트 딜리트로 회원 탈퇴 기능 구현
- 탈퇴 시 해당 유저가 작성한 게시글, 평점, 게임 댓글, 친구 삭제

프로젝트 수행 경과

(5) Amazon s3, google analytics



amazon s3 연동 - s3에 저장되어 있는 이미지를 프로필 이미지로 설정 가능

google analytics 적용 – google analytics를 통해 접속자 수, 평균 참여 시간, 행동 등을 파악 가능

프로젝트 수행 경과

(6) 게시글 기능

```
const KakaoMap = ({ post }) => {
    useEffect(() => {
        if (post) {
            const container = document.getElementById('map');
            const options = {
                center: new kakao.maps.LatLng(post.latitude, post.longitude),
                level: 3,
            };
            const map = new kakao.maps.Map(container, options);

            const setCenter = () => {
                const moveLatLon = new kakao.maps.LatLng(post.latitude, post.longitude);
                map.setCenter(moveLatLon);
            };

            const panTo = () => {
                const moveLatLon = new kakao.maps.LatLng(post.latitude, post.longitude);
                map.panTo(moveLatLon);
            };

            // 마커가 표시될 위치입니다
            var markerPosition = new kakao.maps.LatLng(post.latitude, post.longitude);

            // 마커를 생성합니다
            var marker = new kakao.maps.Marker({
                position: markerPosition,
            });

            // 마커가 지도 위에 표시되도록 설정합니다
            marker.setMap(map);

            setCenter(); // 초기 위치 설정
        }
    }, [post]); // post가 업데이트 될 때마다 실행

    return <div id="map" style={{ width: '100%', height: '100%' }}></div>;
};
```

게시글 작성

- 온라인, 오프라인 구분해서 글 작성
- 카카오맵 api를 활용해 사용자가 원하는 위치 검색 후 지도로 보여주기

```
/*
대댓글 삭제
 */
if(postCommentRepository.isRecomment(commentId)){

    //대댓글의 원댓글이 삭제된 상태이고, 삭제할 대댓글이 마지막 대댓글인 경우
    if(postCommentRepository.hasRecomments(pCommentId)== 1
            && postCommentRepository.isParentCommentDeleted(pCommentId)){

        //대댓글, 원댓글 모두 삭제
        postCommentRepository.deleteById(commentId);
        postCommentRepository.deleteById(pCommentId);
    }else {
        //대댓글만 삭제
        postCommentRepository.deleteById(commentId);
    }

 /*
   댓글 삭제
  */
}else if(postCommentRepository.hasRecomments(commentId) == 0){
    postCommentRepository.deleteById(commentId);
}else{
    //대댓글이 존재하는 경우 삭제된 메시지로 표시
    postCommentRepository.findById(commentId)
            .ifPresent(existingPostComment -> {

                //원댓글은 softDelete
                existingPostComment.deleteComment(LocalDateTime.now(), content: "삭제된 댓글입니다.");
                postCommentRepository.save(existingPostComment);
            });
}
}
```

게시글 상세 페이지

- 댓글 삭제하기
- 댓글 상태에 따라 삭제되는 로직 구분(ex 댓글이 달려있는 원댓글의 경우 삭제할 때 소프트하게 삭제, 삭제된 원댓글의 마지막 대댓글이 지워질 경우 삭제된 원댓글도 함께 삭제)

프로젝트 수행 경과

(6) 게시글 기능

```java
public PostResponseDTO createPost(String username, PostDTO postDTO) {
    User user = userRepository.findByUsername(username);
    Post.OnOffStatus status = "ON".equals(postDTO.getStatus()) ? Post.OnOffStatus.ON : Post.OnOffStatus.OFF;

    //게시글 생성
    Post post = createPost(user, postDTO, status);
    PostResponseDTO postResponseDTO = mapper.PostToPostResponse(post);
    postResponseDTO.setPostUsername(username);

    //채팅방 생성
    ChatRoom chatRoom = chatRoomRepository.save(new ChatRoom(postDTO.getGameTitle(), user, postDTO.getMateCnt(), post));
    chatRoomMemberService.addMember(chatRoom.getId(), user.getId(), isLeader: true);

    return postResponseDTO;
}
```

게시글 작성과 동시에 생성되는 채팅방의 모집인원에 따라 게시글을 모집인원 변경, 이후 모집완료로 변경

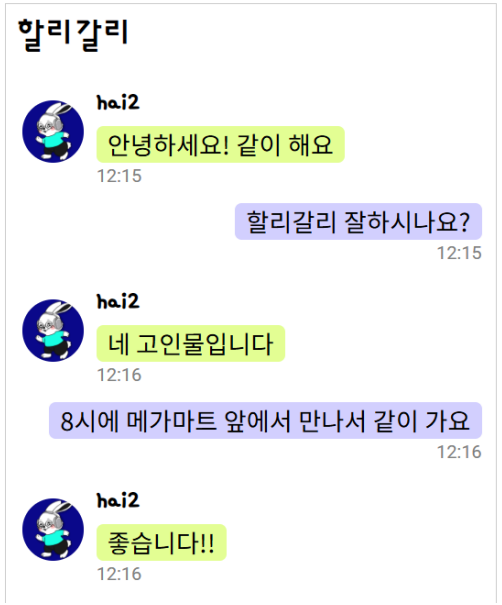
```jsx
const fetchGames = async (pageNumber) => {
    try {
        setLoading(true);
        const response = await api.get(`${apiUrl}?page=${pageNumber}&size=${size}&status=${status}`, {
            headers: {
                Authorization: cookies.token,
            },
        });
        const newPosts = response.data.content; // "content" 배열을 가져옵니다.
        setPosts((prev) => [...prev, ...newPosts]);
        setHasMore(!response.data.last);
    } catch (error) {
        console.error('Error fetching posts:', error);
        setError(error);
    } finally {
        setLoading(false);
    }
};

//status에 따라서 post, page hasmore 초기화
useEffect(() => {
    setPosts([]);
    setPage(0);
    setHasMore(true);

    // 새로운 status에 따라 첫 페이지 데이터 가져오기
    fetchGames(0);
}, [status]); // status가 변경될 때마다 실행

useEffect(() => {
    if (page > 0) {
        fetchGames(page);
    }
}, [page]); // page가 변경될 때마다 실행

const lastGameElementRef = useCallback(
    (node) => {
        if (loading) return;
        if (observer.current) observer.current.disconnect();
        observer.current = new IntersectionObserver((entries) => {
            if (entries[0].isIntersecting && hasMore) {
                setPage((prevPage) => prevPage + 1);
            }
        });
        if (node) observer.current.observe(node);
    },
    [loading, hasMore, status]
);

const handlePostClick = (id) => {
    navigate(`/gamemate/posts/${id}`);
};

if (loading && posts.length === 0) {
    return <div variant="h6">Loading...</div>;
}

return (
    <div>
        {posts.map((post, index) => (
            <div
                ref={posts.length === index + 1 ? lastGameElementRef : null}
                key={index}
                onClick={() => handlePostClick(post.id)}
            >
                <PostListCard {...post} />
```

게시글 조회(on/off)

온라인, 오프라인 별로 보여주는 항목 분리(온, 오프를 누를 때마다 페이지 변경 후 각각 스크롤(무한 스크롤 적용해서 내릴 때마다 새로운 페이지 로딩)

프로젝트 수행 경과

(7) 채팅 기능



- 채팅방에 참여한 인원끼리 **실시간 채팅**이 가능
- 채팅 내용이 DB에 저장 -> 방을 나갔다가 들어와도 이전 채팅을 볼 수 있다
- 채팅방 참가신청 메시지는 방장만 볼 수 있다.
- 방장에 의해 채팅방이 없어지면 더 이상 채팅방에 머무를 수 없음

```java
@Slf4j
@Configuration
public class WebSocketAuthConfig implements WebSocketMessageBrokerConfigurer{

    @Autowired
    private JWTUtil jwtUtil;

    @Override
    public void configureClientInboundChannel(ChannelRegistration registration) {
        registration.interceptors(new ChannelInterceptor() {
            @Override
            public Message<?> preSend(Message<?> message, MessageChannel channel) {
                StompHeaderAccessor accessor = MessageHeaderAccessor.getAccessor(message, StompHeaderAccessor.class);

                // 스톰프 메시지가 CONNECT 메시지일때 인증정보 세션에 저장
                if (StompCommand.CONNECT.equals(accessor.getCommand())) {
                    String token = accessor.getFirstNativeHeader("Authorization");

                    if (token != null && token.startsWith("Bearer ")) {
                        token = token.substring(7);
                        if (!jwtUtil.isExpired(token)) {
                            String username = jwtUtil.getUsername(token);
                            UsernamePasswordAuthenticationToken authentication = new UsernamePasswordAuthenticationTo
ken(username, null, new ArrayList<>());
                            accessor.setUser(authentication);
                            // webSocket 세션에 인증 정보 저장
                            accessor.getSessionAttributes().put("auth", authentication);
                        } else {
                            log.error("Token is expired at WebSocketAuthConfig");
                            throw new IllegalArgumentException("Invalid token");
                        }
                    } else {
                        log.error("Token is missing or does not start with 'Bearer 'at WebSocketAuthConfig");
                        throw new IllegalArgumentException("No token provided");
                    }
                }

                // 스톰프 메시지가 SEND 혹은 SUBSCRIBE 일때 세션의 인증정보 가져와 메시지에 저장
                if (StompCommand.SEND.equals(accessor.getCommand()) || StompCommand.SUBSCRIBE.equals(accessor.getComm
and())) {
                    // WebSocket 세션에서 인증 정보 가져오기
                    Authentication auth = (Authentication) accessor.getSessionAttributes().get("auth");
                    if (auth != null && auth.isAuthenticated()) {
                        accessor.setUser(auth);
                    }
                }

                return message;
            }
        });
    }
```

WebSocketAuthConfig를 활용하여 웹소켓 메시지가 서버에 들어오기 전 인터셉터 하여 각 메시지의 종류에 따라 **알맞은 처리를 진행**

프로젝트 수행 경과

(8) 게임 api 데이터 저장

```
//게임 api 데이터베이스 추가 api
[Cloud3] 조한휘
@GetMapping("/fetch-games")
public ApiResponse<String> fetchAndSaveGames(
        @RequestParam(required = false) String gametitle,
        @RequestParam(required = false) String entname,
        @RequestParam(required = false) String rateno,
        @RequestParam(required = false) String startdate,
        @RequestParam(required = false) String enddate,
        @RequestParam int display,
        @RequestParam int pageno) {

    log.info("Fetching and saving games with title: {}, entname: {}, rateno: {}", gametitle, entname, rateno);
    gameService.fetchAndSaveGames(gametitle, entname, rateno, startdate, enddate, display, pageno);
    return ApiResponse.successRes(HttpStatus.OK, data: "게임 정보가 성공적으로 저장되었습니다.");
}
```

```
1 usage  [Cloud3] 조한휘
public void fetchAndSaveGames(String gametitle, String entname, String rateno, String startdate, String enddate, int display, int pageno) {
    String jsonResponse = gameApiClient.fetchGames(gametitle, entname, rateno, startdate, enddate, display, pageno);
    logger.debug("Fetched response: {}", jsonResponse);

    if (jsonResponse != null) {
        try {
            GameApiResponse response = objectMapper.readValue(jsonResponse, GameApiResponse.class);

            if (response != null && response.getResult() != null && response.getResult().getItems() != null) {
                for (GameApiResponse.GameItem item : response.getResult().getItems()) {
                    Game game = gameMapper.toEntity(item);
                    gameRepository.save(game);
                    logger.debug("Game saved: {}", game);
                }
            } else {
                logger.warn("No games found in the response");
            }
        } catch (IOException e) {
            logger.error("Failed to parse JSON response", e);
        }
    }
}
```

- fetchGames 메서드를 호출 -> **API로부터 게임 정보 JSON 형식으로 가져옴**
- JSON 응답이 null이 아닌 경우, **GameApiResponse 객체로 파싱**
- 각 아이템을 Game 엔티티로 변환하여 데이터베이스에 저장

프로젝트 수행 경과

(9) 게임리스트

```java
package com.example.gamemate.domain.game.entity;

import ...

[Cloud3] 조한휘
@Table(name = "game_rating", uniqueConstraints = {
        @UniqueConstraint(columnNames = {"user_id", "game_id"})
})
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
@Entity
public class GameRating extends BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
```

- uniqueConstraints: user_id와 game_id를 조합하여 **유니크 제약 조건**을 설정
- 이를 통해 한 명의 유저가 **같은 게임에 대해 하나의 평점만** 남길 수 있다.

프로젝트 수행 경과

(10) 피드백 적용

/*elice*/ 주 3회 **오피스아워**

GitLab 매주 토요일 **코드리뷰**

[코치] 주종현 @jjh · 1 week ago

isLeader라는 값을 선언해서 넣어주는 이유가 따로 있나요?
방을 생성한 사람을 제외하고는 방장이 아니라면 컨트롤러단에서 선언할 필요없습니다.

[코치] 주종현 @jjh · 1 week ago

생성자에서 bean에 올라간 서비스를 주입해주세요.
@Autowired는 순환참조가 발생할 수 있습니다.

피드백 내용 반영 후 수정

Chore: 불필요한 내용 수정
Wonjeong Park authored 2 weeks ago

Feat: Exception 추가, autowired 제거
[Cloud3] 조한휘 authored 1 week ago

- 시연 영상 -

05.
자체 평가 의견

자체 평가 의견

(1) 트러블 슈팅

김이삭 팀원

문제 상황

웹소켓 통신에서 SecurityContextHolder를 사용하여 웹소켓 연결시 인증 정보를 저장하고 이후의 통신시 꺼내어 사용하려 하였으나 연결시에 분명히 인증정보를 저장하였으나 통신시에 SecurityContextHolder이 비어있는 문제 발생.

원인

SecurityContextHolder는 기본적으로 ThreadLocal을 사용하여 스레드별로 독립적인 SecurityContext를 유지한다고 하는데 웹소켓의 연결과 메시지 전송은 서로 다른 스레드에서 처리될 수 있음. 서로 다른 스레드여서 부모의 SecurityContext 내용을 자식에게 전파하는 SecurityContextHolder의 ThreadLocal 설정도 먹히지 않았음.

```
Current Message: CONNECT,
Thread: http-nio-8080-exec-5,
SecurityContextHolder.getContext().getAuthentication() : UsernamePasswordAuthenticationToken [Principal=test1@t.co
m, Credentials=[PROTECTED], Authenticated=true, Details=null, Granted Authorities=[]]

Current Message: SUBSCRIBE,
Thread: http-nio-8080-exec-8,
SecurityContextHolder.getContext().getAuthentication() : null
```

웹소켓 연결메시지와 구독메시지의 스레드와 SecurityContextHolder log.

해결 방법

스레드 별로 값이 달라지는 SecurityContextHolder가 아닌 클라이언트가 서버에 연결된 동안 동일한 상태가 유지되는 세션에 인증정보를 저장하여 사용하는 것으로 해결함.

자체 평가 의견

(1) 트러블 슈팅

박원정 팀장

문제 상황

복합키 매핑 오류 발생:
@IdClass 어노테이션을 사용하여 Friend 엔티티에 복합키를 매핑하려 했으나, 예상치 못한 오류가 발생함.

원인

@IdClass 규칙 위반:
@IdClass를 사용하면, 해당 클래스로 정의된 모든 필드가 @Entity 클래스에 반드시 포함되어야 함.

필드 매핑 오류:
FriendId 클래스의 필드 requester와 receiver가 Long 타입으로 정의되어 있으나, Friend 엔티티에서는 User 타입으로 사용

해결 방법

매핑 변경: FriendId 클래스에서 requester와 receiver 필드를 Long이 아닌 User 타입으로 수정하여 Friend 엔티티와 일치시킴.

대안 고려: @IdClass 대신 @EmbeddedId를 사용하는 것도 고려할 수 있음. 이 경우, FriendId를 @Embeddable로 정의하여 필드 타입 불일치를 해결할 수 있음.

자체 평가 의견

(2) 회고

박원정 팀장

해보지 않은 기술을 다양하게 사용하면서 새롭게 배울 수 있는 것도 많은 좋은 기회였습니다. 특히 열정적으로 참여해 주신 팀원 분들 덕분에 저 또한 개발에 많은 의지와 열정을 다지게 되는 아주 좋은 동기부여를 얻을 수 있었고, 최선을 다해 프로젝트를 마무리할 수 있었습니다. 물론 문제 상황도 가끔 발생하고, 모르는 부분도 상당히 많았습니다. 하지만 이러한 경험이 그래도 조금은 쌓이게 된 덕일까 새로운 문제나 모르는 개념이 등장하면 겁 먹지 않고 시도해보는 태도를 가지게 되었습니다. 이번 프로젝트는 앞으로 더 새롭고 다양한 것을 배워보고 싶은 마음과 활용해 보고 싶은 마음을 더욱 더 강해지게 해주는 계기가 되었습니다.

고의성 팀원

이전 프로젝트에서도 맡았을 때 많이 부족했던 회원 부분이어서 이번에 좀 더 공부해 볼 생각으로 다시 회원을 해보기로 했다. 하지만 내 학습 능력이 현저히 부족해 공부는 커녕 기간 내에 기능을 다 구현하지도 못 할 뻔 했다. 심지어 네이버, 구글 로그인 API를 활용하여 로그인하는 OAuth2 기능은 시도를 해보다 기간도 너무 오래 잡아먹고 돌파구가 보이지 않아 결국 제외시키기도 했다. 이번 프로젝트에서 가장 아쉬운 부분이다. 기회가 된다면 좀 더 알아보고 실습해본 후 기능도 추가하고 리팩토링 하겠다는 생각도 가지고 있다. 이전 프로젝트때도 지금 팀원들과 함께 프로젝트를 진행했는데 저번에 이어서 이번에도 부족함을 많이 보인 거 같아 미안하고, 같이 힘내보자며 응원을 해주던 팀원들에게 고마운 마음이 크다. 사실 가장 먼저 어느정도 구현이 되었어야 할 회원 파트였는데 담당자의 부족함으로 인해 팀원들이 기능을 구현하는 데 어려움이 있었을 것임에도 불구하고 믿고 기다려준 점에 대해 팀원들에게 무한하게 감사한다. 프로젝트를 마치며 처음부터 다시 공부해야 될 필요성을 많이 느끼고 있다. 그나마 어느 부분을 어떻게 공부해야 될 지에 대해 조금은 알아간 것 같다는 점을 긍정적으로 보는 중이다. 이번 프로젝트가 크게 한 발 내딛을 수 있는, 마인드의 변화를 줄 전환점이 된 것 같고 성장할 수 있는 디딤돌이 된 것 같다. 마무리로 함께 해 준 팀원에게 다시 한 번 감사를 표하며 그들의 앞날에도 행복만 있기를 기원한다.

자체 평가 의견

(2) 회고

김경래 팀원

이번 프로젝트는 저번 팀원들과 같이 해서 좀 편하게 진행할 수 있었던 프로젝트였다. 하지만 이번 프로젝트는 저번 프로젝트보다 더 적극적으로 하지 못한 부분이 많아 그 부분이 아쉬웠다. 내가 좀 더 많은 역할을 했다면 좀 더 높은 완성도를 낼 수 있었던 프로젝트라 아쉬운 부분이 많고, 다른 사람들이 짠 코드를 보면서 부족한 부분 (인증, 채팅 등) 을 공부해야겠다는 생각이 들었다. 이번에 알게 된 google analytics를 제대로 적용하지 못한 점이 아쉽다. 좀 더 효율적으로 적용하면 더 많은 정보를 얻을 수 있을 것 같아 이 부분을 다시 찾아보는 시간이 필요할 것 같다.

마지막으로, 이번 프로젝트는 너무 수동적으로 한 것 같아 팀원들에게 미안한 마음이 들었다. 이 부분을 개선하여 좀 더 적극적으로 참여하고 다른 동료들을 도와줄 수 있는 사람이 될 수 있도록 좀 더 많은 부분을 공부하여 알려줄 수 있도록 하면 좋을 것 같다.

김연지 팀원

이번에 게시판 파트를 맡게 되었는데, 첫번째 개인 프로젝트 때 이미 게시글 작성은 해봤기에 이번에는 기본적으로 코드를 최대한 잘 짜야겠다는 생각으로 프로젝트에 임했습니다. 그래서 팀원들이 공통으로 필요한 글로벌 예외처리, Api 응답 클래스 등을 짜면서 전체적인 구조를 잡아가는 것에 초점을 맞췄습니다. 또, 어떻게 하면 중복되는 코드를 줄일 지에 대한 생각을 많이했던 것 같습니다. 결과적으로 예전 프로젝트 때 작성했던 코드보다는 더 깔끔하게 작성할 수 있게 되어 뿌듯했습니다. 특히 백엔드, 프론트엔드 코치님 두 분의 피드백이 많은 도움이 되었습니다.

마지막으로 중간 프로젝트와 마지막 프로젝트를 같은 팀원분들끼리 진행하게 되었는데 한 분도 빠짐없이 열심히 하시는 모습에 더 열정적으로 개발을 할 수 있었습니다. 팀원분들께도 너무 고생하셨고 감사하다고 말씀드리고 싶습니다!

자체 평가 의견

(2) 회고

김이삭 팀원

이번에 처음으로 웹소켓을 다뤘는데 처음 사용해보는 거라 익숙하지 않아 초반에 어려움이 있었습니다. 특히 웹소켓을 프론트에서 연결하는 부분과 서버에서 인증하는 부분에서 많은 시간을 사용하였습니다. 그래도 기능이 제대로 구현되어 정상적으로 채팅이 실시간으로 주고받아 질 때는 굉장히 신기하였고 생각한대로 기능이 동작할 때 뿌듯하였습니다. 프로젝트 기간이 쉽지만은 안았지만 2차 프로젝트 때 함께했던 팀원들과 다시 함께하게 되어서 서로 원활하게 소통하고 응원하면서 마무리까지 잘 할 수 있었습니다. 각자 맡은 부분을 끝까지 잘 수행해준 팀원들에게 고맙고 감사합니다.

조한휘 팀원

이번 프로젝트는 엘리스 트랙에서 배운 기술 스택을 바탕으로 더 깊이 공부하고 개발할 수 있어 매우 뜻깊은 시간이었습니다. 게임 api 데이터 저장, 게임리스트 기능, FE 레이아웃을 담당하면서 많은 것을 배울 수 있었습니다. 이번 프로젝트에서도 팀원분들께 많은 도움을 받았습니다. 정말 감사합니다. 또한, 오피스아워와 코드 리뷰를 통해 지도해주신 코치님들께도 감사드립니다.

Thank you.