

Term Project Final Presentation

32191597 박민규

32191585 박도훈

32222174 서영빈

32223436 이은서



목 차

프로젝트 소개

01

추가된 기능

03

기술 스택

02

최종 결과물 설명

04

06



프로젝트 소개

프로젝트 배경 & 목표

- 커뮤니티의 한계: 기존의 취업 정보 제공 플랫폼들은 다수의 사용자들이 참여하는 커뮤니티의 형태. 커뮤니티는 다양한 의견을 얻을 수 있지만 개개인의 상황에 맞는 구체적인 정보를 찾기는 어려움
- 맞춤형 정보의 필요성: 취준생들은 자신만의 상황과 목표에 맞는 정보가 필요. 1:1 채팅을 통해 직접 현직자와 소통하면서 필요한 정보를 바로 받을 수 있음.

"현직자와 취준생을 1:1로 연결하여 실질적인 정보와 조언을 제공하는 플랫폼"



기술 스택

커리어타임



추가된 기능

1. 채팅기능

```

15  @RestController
16  @RequestMapping("/api/chats")
17  @RequiredArgsConstructor
18  public class ChatController {
19      private final ChatService chatService;
20      private final SimpMessagingTemplate messagingTemplate;
21
22      @MessageMapping("/chat.sendMessage")
23      public void sendMessage(ChatMessage chatMessage) {
24          ChatRoom chatRoom = chatService.getChatRoom(chatMessage.getSenderId(), chatMessage.getReceiverId());
25          if (chatRoom == null) {
26              chatRoom = chatService.createChatRoom(chatMessage.getSenderId(), chatMessage.getReceiverId());
27          }
28          ChatMessage savedMessage = chatService.saveMessage(chatRoom.getRoom_id(), chatMessage.getSenderId(), chatMessage.getRecei
29          messagingTemplate.convertAndSendToUser(chatMessage.getReceiverId().toString(), destination:"/queue/messages", savedMessag
30      }
31
32      // @GetMapping("/{chatRoomId}/messages")
33      // public ResponseEntity<List<ChatMessage>> getMessages(@PathVariable("chatRoomId") Long chatRoomId) {
34      //     return ResponseEntity.ok(chatService.getMessages(chatRoomId));
35      // }
36
37      @GetMapping("/{userId}")
38      public ResponseEntity<List<ChatResponse>> getChatResponses(@PathVariable("userId") Long userId) {
39          return ResponseEntity.ok(chatService.getChatResponses(userId));
40      }
41
42      @PostMapping("/{userId}/{yourId}")
43      public ChatRoom createChatRoom(@PathVariable("userId") Long userId, @PathVariable("yourId") Long yourId) {
44
45          ChatRoom chatRoom = chatService.getChatRoom(userId, yourId);
46          if (chatRoom == null) {
47              chatRoom = chatService.createChatRoom(userId, yourId);
48          }
49
50          return chatService.getChatRoom(userId, yourId);
51      }
52  }

```

ChatController.java

웹소켓 메시지 처리를 통한 메시지 송수신
&채팅방 관리 기능 제공

```

@Entity
@Table(name = "chat_rooms")
@Data
@NoArgsConstructor
public class ChatRoom {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long room_id;

    @ManyToOne
    @JoinColumn(name = "sender_id", nullable = false)
    private User sender;

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

```

ChatRoom.java

채팅 메시지의 데이터 구조를 정의&
MongoDB에 저장

```

@Data
@Document(collection = "chat_messages")
public class ChatMessage {
    @Id
    private String id;
    private Long chatRoomId;
    private Long senderId;
    private Long receiverId;
    private String message;
    private LocalDateTime timestamp;
}

```

ChatMessage.java

채팅방의 정보를 저장&
발신자와 수신자 정보 포함

추가된 기능

1. 채팅기능

커리어타임

문제 해결을 위한 자바 활용

```

26  @Service
27  @RequiredArgsConstructor
28  public class ChatService {
29      private final ChatRoomRepository chatRoomRepository;
30      private final ChatMessageRepository chatMessageRepository;
31      private final UserRepository userRepository;
32      private final ProfileRepository profileRepository;
33
34      public ChatRoom createChatRoom(Long senderId, Long receiverId) {
35          User sender = userRepository.findById(senderId)
36              .orElseThrow(() -> new IllegalArgumentException("Invalid sender ID: " + senderId));
37          User receiver = userRepository.findById(receiverId)
38              .orElseThrow(() -> new IllegalArgumentException("Invalid receiver ID: " + receiverId));
39
40          ChatRoom chatRoom = new ChatRoom();
41          chatRoom.setSender(sender);
42          chatRoom.setReceiver(receiver);
43          return chatRoomRepository.save(chatRoom);
44      }
45
46      public ChatRoom getChatRoom(Long senderId, Long receiverId) {
47          User sender = userRepository.findById(senderId)
48              .orElseThrow(() -> new IllegalArgumentException("Invalid sender ID: " + senderId));
49          User receiver = userRepository.findById(receiverId)
50              .orElseThrow(() -> new IllegalArgumentException("Invalid receiver ID: " + receiverId));
51          return chatRoomRepository.findBySenderAndReceiver(sender, receiver);
52      }
53
54      public ChatMessage saveMessage(Long chatRoomId, Long senderId, Long receiverId, String message) {
55          ChatMessage chatMessage = new ChatMessage();
56          chatMessage.setChatRoomId(chatRoomId);
57          chatMessage.setSenderId(senderId);
58          chatMessage.setReceiverId(receiverId);
59          chatMessage.setMessage(message);
60          chatMessage.setTimestamp(LocalDateTime.now());
61          return chatMessageRepository.save(chatMessage);
62      }
63
64      public List<ChatMessage> getMessages(Long chatRoomId) {
65          return chatMessageRepository.findByChatRoomId(chatRoomId);
66      }
67
68      public List<ChatResponse> getChatResponses(Long userId) {
69          List<ChatRoom> chatRooms = chatRoomRepository.findAll().stream()
70              .filter(chatRoom -> chatRoom.getSender().getUser_id().equals(userId) || chatRoom.getRe
71              .collect(Collectors.toList());

```

```

68  public List<ChatResponse> getChatResponses(Long userId) {
69      List<ChatRoom> chatRooms = chatRoomRepository.findAll().stream()
70          .filter(chatRoom -> chatRoom.getSender().getUser_id().equals(userId) || chatRoom.getRe
71          .collect(Collectors.toList());
72
73      return chatRooms.stream()
74          .collect(Collectors.groupingBy(chatRoom -> {
75              if (chatRoom.getSender().getUser_id().equals(userId)) {
76                  return chatRoom.getReceiver().getUser_id();
77              } else {
78                  return chatRoom.getSender().getUser_id();
79              }
80          }))
81          .entrySet()
82          .stream()
83          .map(entry -> {
84              Long otherUserId = entry.getKey();
85              List<ChatRoom> groupedChatRooms = entry.getValue();
86              User otherUser = userRepository.findById(otherUserId)
87                  .orElseThrow(() -> new IllegalArgumentException("Invalid user ID: " + othe
88              Optional<Profile> profileOptional = profileRepository.findByUser(otherUser);
89              String profilePicture = profileOptional.map(Profile::getProfilePicture).orElse("");
90
91              List<ChatMessageDTO> messages = groupedChatRooms.stream()
92                  .flatMap(chatRoom -> getMessages(chatRoom.getRoom_id()).stream()
93                      .map(msg -> new ChatMessageDTO(msg.getSenderId().equals(userId) ?
94                          .collect(Collectors.toList()));
95
96              ChatMessageDTO lastMessage = messages.isEmpty() ? null : messages.get(messages.siz
97
98              return new ChatResponse(
99                  groupedChatRooms.get(0).getRoom_id(),
100                 otherUserId,
101                 otherUser.getName(),
102                 profilePicture,
103                 lastMessage == null ? "" : lastMessage.getTime(),
104                 lastMessage == null ? "" : lastMessage.getContent(),
105                 messages
106             );
107         })
108     }.collect(Collectors.toList());
109 }

```

ChatService.java

채팅방 생성, 메시지 저장 및 검색, 사용자의 채팅방 응답 생성 등
채팅 관련 비즈니스 로직을 처리

추가된 기능

1. 채팅기능

```

1 package kr.ac.dankook.ace.careertime.dto;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6
7 import java.util.List;
8
9 @Data
10 @AllArgsConstructor
11 @NoArgsConstructor
12 public class ChatResponse {
13     private Long room_id;
14     private Long receiver_id;
15     private String receiver_name;
16     private String receiver_profileImage;
17     private String lastChatDate;
18     private String lastChatMessage;
19     private List<ChatMessageDTO> messages;
20 }

```

ChatResponse.java

채팅방의 상태&메시지 내역을 요약하여
클라이언트에 전달

ChatMessageDTO.java
데이터 전송 객체
채팅 메시지의 간단한 정보를 전달하는 데 사용

```

1 package kr.ac.dankook.ace.careertime.dto;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6
7 @Data
8 @AllArgsConstructor
9 @NoArgsConstructor
10 public class ChatMessageDTO {
11     private String sender;
12     private String content;
13     private String time;
14 }

```

추가된 기능

2. 추천 기능

```

@Service
@RequiredArgsConstructor
public class RecommendationService {

    private final ProfileRepository profileRepository;
    private final BoardRepository boardRepository;
    private final UserRepository userRepository;

    public List<BoardResponse> recommendBoardsForUser(Long userId) {
        Optional<User> user = userRepository.findById(userId);
        if (!user.isPresent()) {
            return new ArrayList<>();
        }

        Optional<Profile> profile = profileRepository.findByUser(user.get());
        if (!profile.isPresent()) {
            return new ArrayList<>();
        }

        String[] userHashtags = profile.get().getHashtags().split(",");
        Map<Board, Long> boardMatchCount = new HashMap<>();
        for (String hashtag : userHashtags) {
            List<Board> boards = boardRepository.findByHashtagsContaining(hashtag.trim());
            for (Board board : boards) {
                boardMatchCount.put(board, boardMatchCount.getOrDefault(board, 0L) + 1);
            }
        }

        return boardMatchCount.entrySet().stream()
            .sorted(Map.Entry.<Board, Long>comparingByValue().reversed())
            .limit(10)
            .map(Map.Entry::getKey)
            .map(this::mapToBoardResponse)
            .collect(Collectors.toList());
    }

    public Profile getProfileByUser(User user) {
        return profileRepository.findByUser(user)
            .orElseThrow(() -> new IllegalArgumentException("Profile not found for user: " + user.getUsername()));
    }
}

```

```

private BoardResponse mapToBoardResponse(Board board) {
    User user = board.getUser();
    Profile profile = getProfileByUser(user);

    UserInfo userInfo = new UserInfo();
    userInfo.setUser_id(user.getUser_id());
    userInfo.setUsername(user.getUsername());
    userInfo.setUsercompany(profile.getCompany_name());
    userInfo.setUseremail(user.getEmail());
    userInfo.setUserimage(profile.getProfilePicture());
    userInfo.setUserinterest(Arrays.asList(profile.getHashtags().split(", ")));

    BoardResponse boardResponse = new BoardResponse();
    boardResponse.setPost_id(board.getPost_id());
    boardResponse.setTitle(board.getTitle());
    boardResponse.setHashtags(Arrays.asList(board.getHashtags().split(", ")));
    boardResponse.setContent(board.getContent());
    boardResponse.setPostdate(board.getPost_date().format(DateTimeFormatter.ISO_LOCAL_DATE));
    boardResponse.setUserinfo(userInfo);

    return boardResponse;
}

```



RecommendationService.java

사용자의 관심사에 따라 게시물을 찾아 추천 리스트 생성

추가된 기능

2. 추천 기능

```
@RestController  
@RequestMapping("/api/boards")  
public class RecommendationController {  
  
    private final RecommendationService recommendationService;  
  
    @Autowired  
    public RecommendationController(RecommendationService recommendationService) {  
        this.recommendationService = recommendationService;  
    }  
  
    @GetMapping("/recommend")  
    public ResponseEntity<List<BoardResponse>> getRecommendations(@RequestParam("userId") Long userId) {  
        List<BoardResponse> recommendations = recommendationService.recommendBoardsForUser(userId);  
        if (recommendations.isEmpty()) {  
            return ResponseEntity.noContent().build();  
        }  
        return ResponseEntity.ok(recommendations);  
    }  
}
```



RecommendationController.java

사용자의 관심사에 맞는 게시물을 추천하기 위해
해당 서비스의 함수 호출

Chapter 3

3. 검색 & 마이페이지

추가된 기능

<마이페이지>

The screenshot shows the user profile page for '박민규'. At the top, there is a large placeholder for a profile picture. Below it, the name '제 이름은 박민규 입니다.' is displayed, followed by the text '나이는 반오십...'. Underneath, the user's name '박민규' is shown again, along with a placeholder for company name and email ('회사명을 입력해주세요 parkm2ngyu00@naver.com'). A list of hashtags includes '#백엔드', '#스프링', and '#리액트'. At the bottom are two blue buttons: '프로필 수정하기' and '내 채팅 보기'.

This screenshot shows the same user profile page for '박민규', but with a different visual style. It features a large placeholder for a profile picture at the top. Below it, the name '제 이름은 박민규 입니다.' is displayed, followed by the text '나이는 반오십...'. Underneath, the user's name '박민규' is shown again, along with a placeholder for company name and email ('회사명을 입력해주세요 parkm2ngyu00@naver.com'). A list of hashtags includes '#백엔드', '#스프링', and '#리액트'. At the bottom are two blue buttons: '프로필 수정하기' and '상세 프로필 보기'.

<검색>

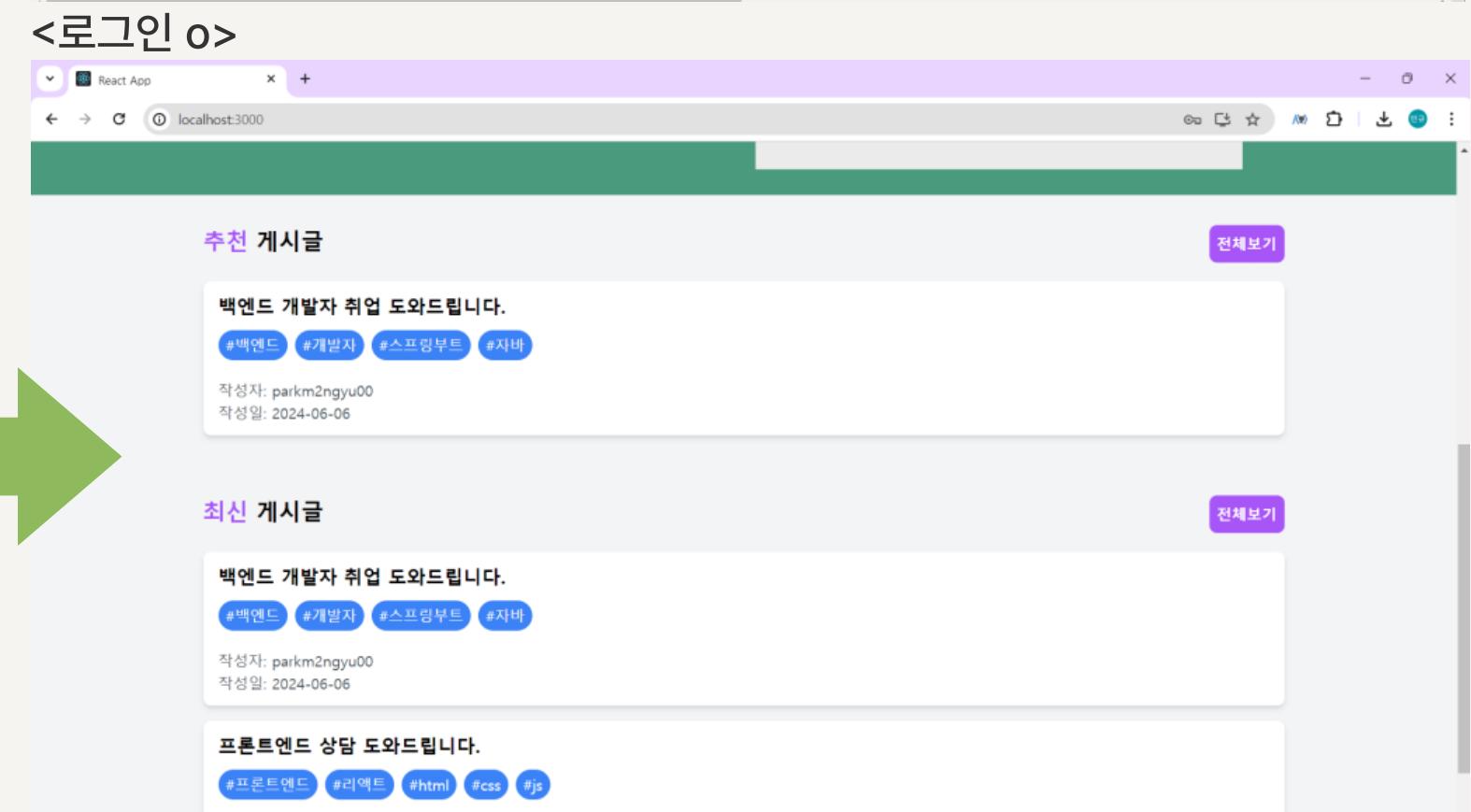
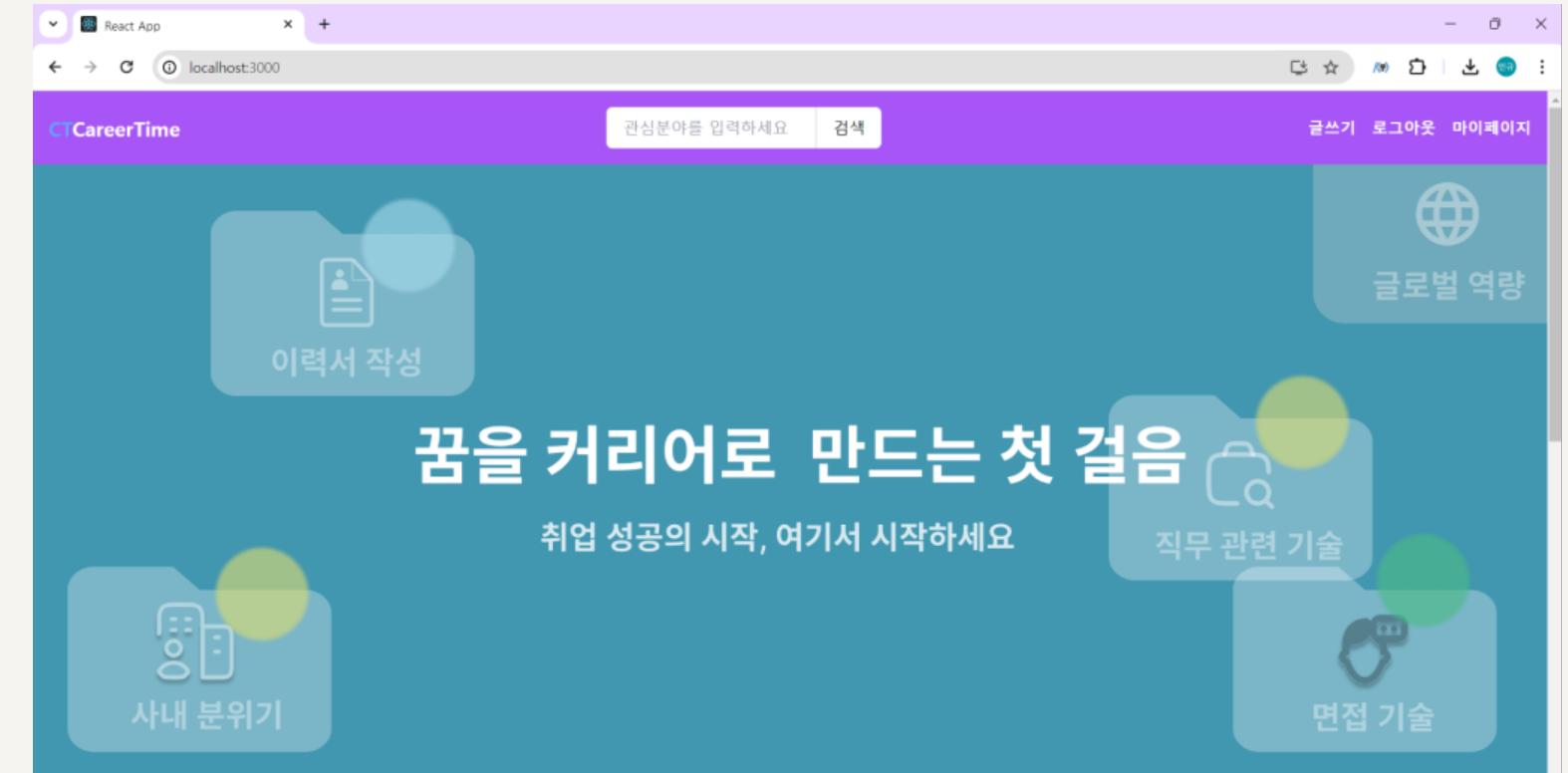
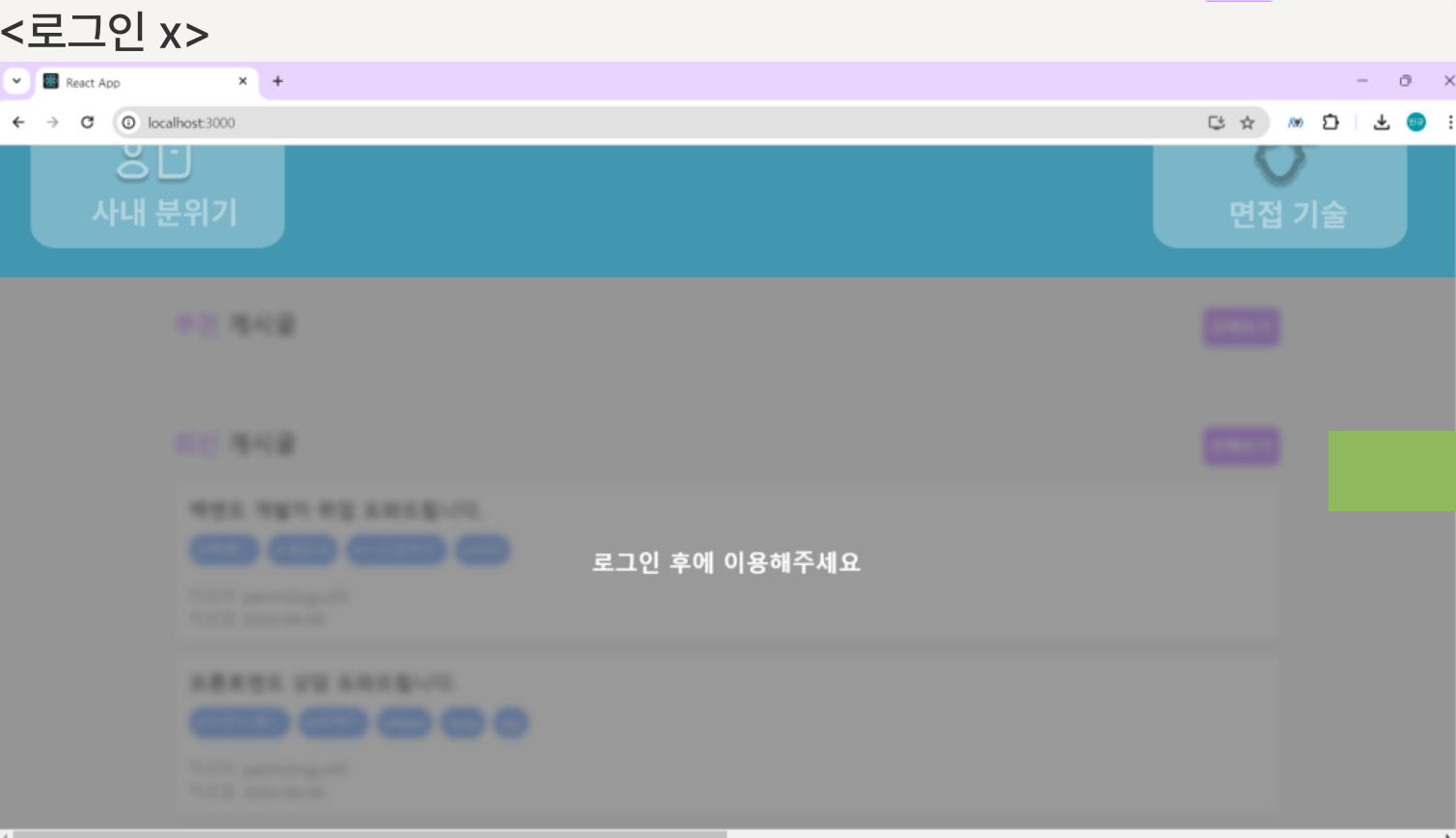
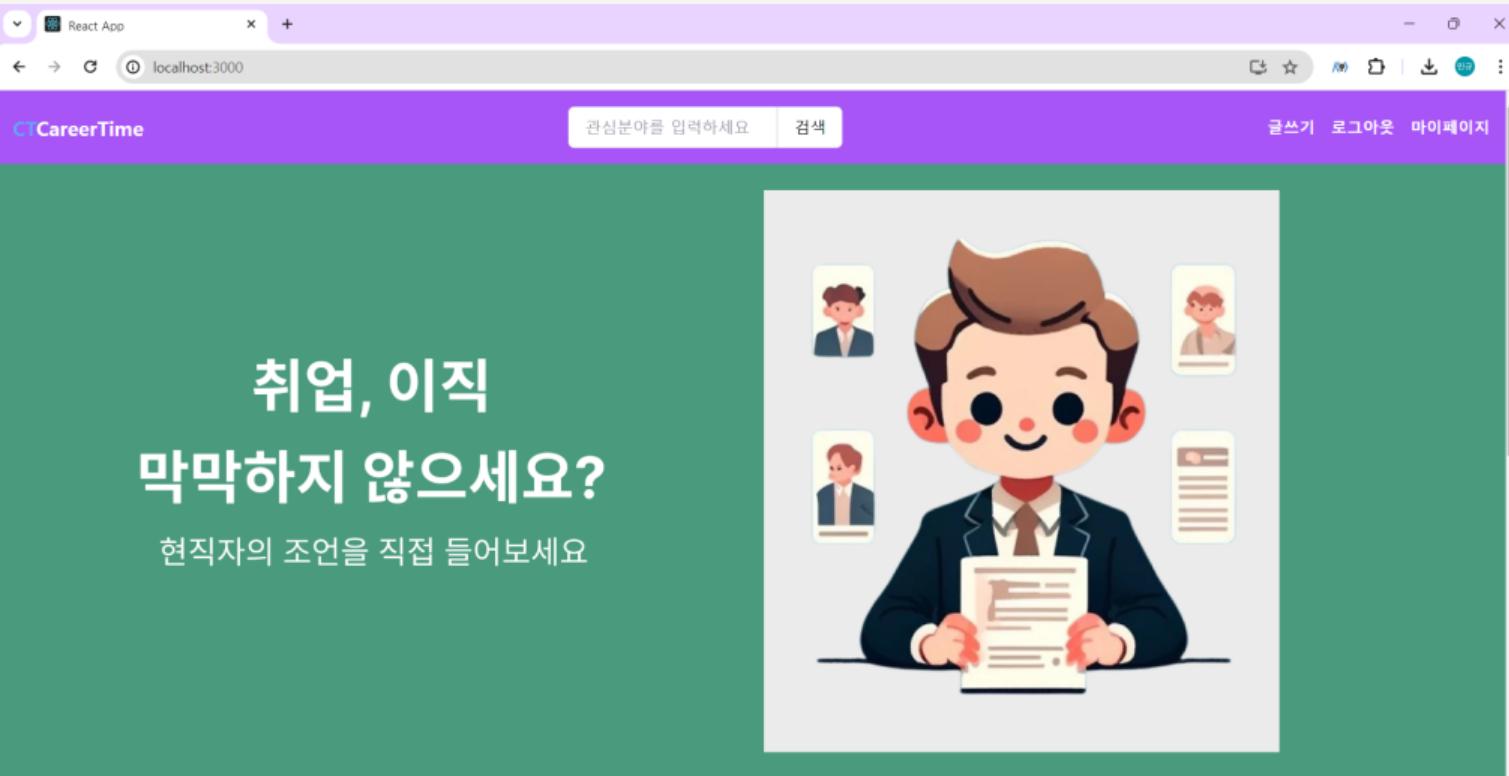
The screenshot shows the search results for the hashtag '#고민'. The title is '#고민 검색결과'. It lists one result: '자바 고민' by '자바 고민' (작성자: dmstj-1121, 작성일: 2024-06-10). The result includes the hashtags '#자바' and '#고민'.

This screenshot shows the search results for the word '자바'. The title is '자바 검색결과'. It lists two results: '자바 백엔드 개발자 상담' by '자바 백엔드 개발자 상담' (작성자: parkm2ngyu00, 작성일: 2024-06-06) and '자바 고민' by '자바 고민' (작성자: dmstj-1121, 작성일: 2024-06-10). Both results include the hashtags '#자바' and '#고민'.

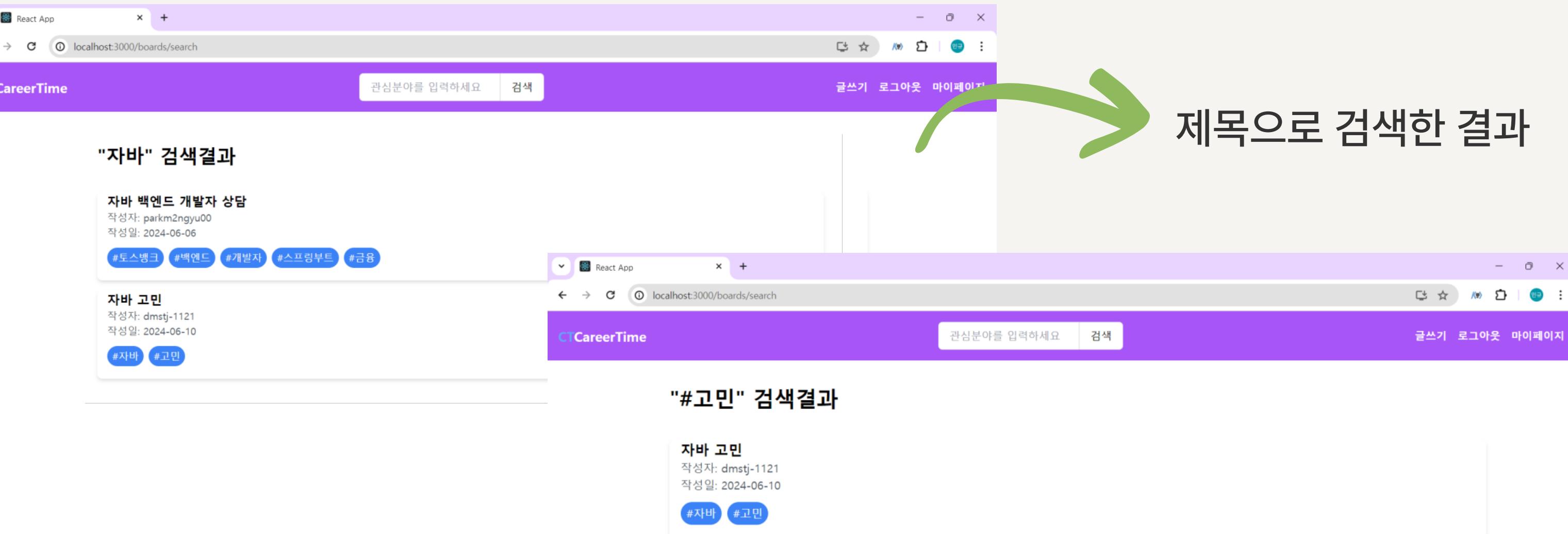
문제해결을 위한 자바 활용

웹 구성

1. 사이트 접속 시 첫 화면



웹 구성



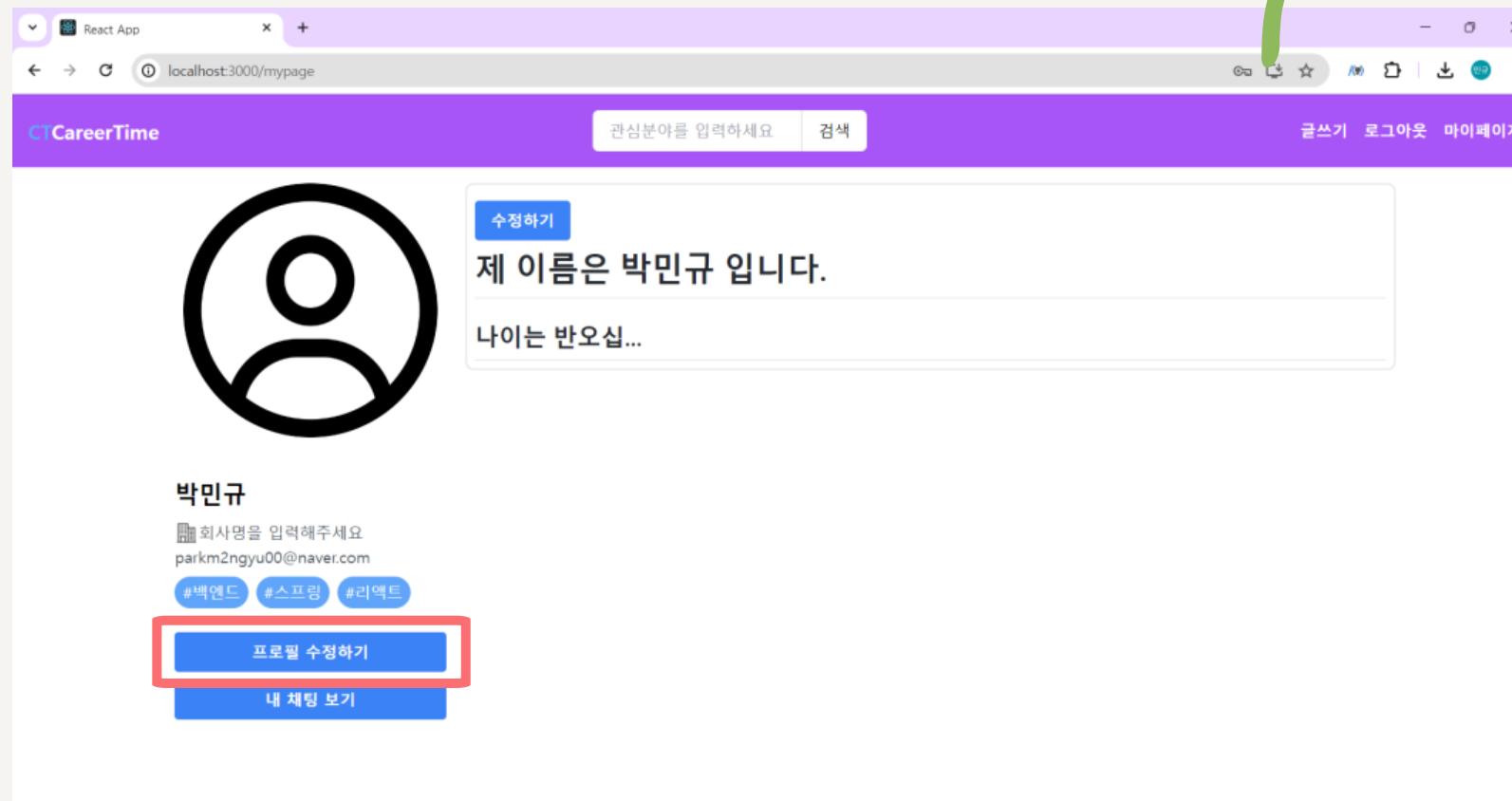
해시태그로 검색한 결과

웹 구성

2. 마이페이지 - 수정

커리어타임

2. 마이페이지



```

<div align="center">
<img alt="Profile picture of a person standing next to a tall vertical plaque." data-bbox="495 170 588 324"/>
<div>
  박민규
  토스뱅크
  parkm2ngyu00@naver.com
  #해시태그
  #백엔드 x #스프링 x
  #리액트 x #프론트엔드 x
</div>
<button>저장하기</button>

```

2. 마이페이지 - 수정 결과



문제해결을 위한 자바 활용

웹 구성

2. 마이페이지 - 내 채팅 보기

This screenshot shows the 'MyPage' section of the application. At the top, there's a purple header bar with the 'CTCareerTime' logo, a search bar, and navigation links for '글쓰기', '로그아웃', and '마이페이지'. Below the header is a large placeholder for a user profile picture. To its right, a blue button labeled '수정하기' (Edit) is visible. Below the button, the text '제 이름은 박민규입니다.' (My name is Park Min-gyu) is displayed. Underneath this, there's a placeholder for age information. On the left side of the main content area, there's a user profile section for '박민규' (Park Min-gyu). It includes a placeholder for company name and email (parkm2ngyu00@naver.com), and a list of hashtags: '#백엔드', '#스프링', and '#리액트'. Below this is a blue button labeled '프로필 수정하기' (Edit Profile) and a red-bordered blue button labeled '내 채팅 보기' (View Chats). A large green arrow points from this screen to the next one.

1:1 채팅 목록

This screenshot shows the '1:1 Chat List' section. It has a similar purple header bar with the 'CTCareerTime' logo and navigation links. The main content area displays a list of recent chats under a blue header '채팅 목록' (Chat List). The first item in the list is for '서영빈' (Seo Young-bin), with a placeholder for company name and email (parkm2ngyu00@naver.com), and a timestamp (2024-06-06T22:43:47.541). The second item is for '이은서' (Lee Eun-seo), with a placeholder for company name and email (parkm2ngyu00@naver.com), and a timestamp (2024-06-06T12:18:23.033). A large green arrow points from the previous screen to this one.

1:1 채팅 화면

This screenshot shows the '1:1 Chat Room' interface. The top bar is purple with the 'CTCareerTime' logo and navigation links. The main area is titled '서영빈' (Seo Young-bin). It shows a conversation history with messages from both users. The messages are as follows:

- 박민규: 안녕하세요 (Hello) - 오후 12:16
- 서영빈: 제 이름은 영빈이에요 (My name is Youngbin) - 오후 12:16
- 박민규: 제 나이는 22살이구요 (I am 22 years old) - 오후 10:38
- 서영빈: 고민이 있어요.... (I have a problem...) - 오후 10:38
- 박민규: 그게... (That's...) - 오후 10:42
- 서영빈: 고민이 뭐예요? (What's your problem?) - 오후 10:40
- 박민규: 뭐라고! (What!) - 오후 10:47
- 서영빈: 뭐라고? (What!) - 오후 10:43

At the bottom of the screen, there's a message input field with the placeholder '메시지를 입력하세요' (Enter message) and a blue '전송' (Send) button.

웹 구성

3. 게시글 작성

커리어타임

문제 해결을 위한 자바 활용

본문 내용

마크다운 형식을 사용해서
다양하게 사용자가 직접 꾸밀 수 있음

외부 이미지도 불러올 수 있음

```
![img]
(data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAQ0AAC7CAMAAABIKDvmaAAAgVBMVEX//9tsz9psTlrsjz+//1lrzL1+vFosTbd
7dNkry7F4LSaynv4/PWVx3elz41sjsrl4bzq15y72qj22KF5uk3U6Mjk8dzv9urA3a7r9eXQ5sNxtUTI4blhriqRxXGfzlWJwWaBvVvh79h6uVGLwm
mezIKs0SDvl9aqxtSpwPj7d0yXZV8AAAN0k1EQVR4nO1d54KjvA4FFwwJwSF0SCghkLnf+z/gpZjOFGYTsmF9fuzO7FLsgyRLtiwLAGchBwc
HBwcHBwcHBwcHBwcHBwcHBwcHB8ffCaRKnu7rqf6f3KNI0yp6dYteBiSftdAUlYIxIRgrGJr36GQ5r27XCyD0T2gBECxB0gJEdP92fm3ROR
6SjCh4iwgUMxl/mf4QHIWgE+oYIQQMT6qvVsk72WtTJkjRD4FRcVKl6tTj5O+Qsb/EQ4ugi+paICEMNGlpz7XXppq58EOfmBXDQgwakWD/0i
Wi9u+BOA8uCHglEDkn0pHh7NoPbqtj8czl75uWA04iELapiqYaB+/y3gn1JBg1qOnrH75w/NiYqsgpWU5G6ZFdigHFCbalKlfzSxfjc1y0QkIQIGz
JY7fTX6hJCWVfjSs+3pCqSPHvyICXqDafDtVf3IXHAE3xr8igpHVCb8lmQpfT7ySDAL99hPuxlVjSptf6GZVgXA529wz7cnpdBx4JFP5CNCA2T
33dkJLDNRlt5wMqlj5KFDLzU2oipQs9TQAFlNvLAInsokx1IAWUoHvxoyrZRNj/bY/HD8XDUDgBVkiqWbOTGSjdr930J2D3FRKFAQQXUArQIA3
z3fVTW5ltl5F+09sKCwFQUzv4SHTcsP1ravnfNld4/L+ZtQTZ30NogSh4V+IBaOm9eF/f9Fjt2MTw6xctt5S+Xe24D/NaMohGjyzx+gNqzJzvaUF
q4INRkrCrzEC7gQBLkZYdQ4fHdvVA5GZABxmvdgtJher7dfR/DHZNBFglHErq3pRFH67oPKKJYHqf39PX1InTygPFh4818HFTA1TPFCyRCinl
4Z4vWhbVsdozAelSj+GEe7+8PxuDARYvXvz2PvouxtuzfbZAObCMcEzw76L9v5sHpsKEunveP/DcxM/u5soKhjAwbLFoik8GNAH8reffY
L5Z0VJdkjVxKFI6G3pYbpu6+3nTo28HnJjd79QoaiiKW3d/fMz91c1jia3gJHrPnkcNDm/YCyJ3nZS6Q8zMI7Eo+Ze3nxj1zCaGBT9fOkQnDC+
TFIX84/jYtr0A7crSz9mwE6UYjcdXo5S+uxEtAhW8A0jIOJfmr1ws4/djWi5BtvajZ94osi6XyDE0zk/pL2/Si8hHtDB/xBPG5rhIhAcaf/4wXi0vj3b0
Q7w/Gtv4FsTSn0CidzDnh+eft5wBJ2wEYVEn3dHTkSMRThZT/nf3viZUZg3hDNFA8QIVUz02UgggsRm7OdRnslef+VthJ2k8uifJblaLv7QckVil
wO84GZi8k2RKMYZFvLMc05kDxfSwmuss8Bjj9JSvAVER/7fhkDL2XDymoGNnnPA4UXJNFLun5kx5bAb28/6pjg3MTyBKx3nSCJM/SDYpm
Ki1CGDx4H9mZH1Mwdsvs3VA+4YOQDXPs07RLcBKszcDEwSGp9mKqg5pfTtV1L68FjCPSCkdChwt7sPYGleDPvTsRf5CYZwG0iOLWRly
9S2ZH8jvalBfFESzf+cCgGdD+VYs4kcpz78AEJYDpJNwnkhFOne/YoJ6aqluFQxnG/BCx1gp7DVfkzTOHdmE/vKoEk23KjIFQ+iijDbfhA7ik
HiSdtKAjyU+Glh3CMMyiSNN13WUwdG1/S0x12v2yONqKpzGAW3uTxslGABTcQemjtLG9xDmcbVAYSvxX9mu4MvsdKDE2ZzMaIMvyoyW5t
MTcvbrNT4Sj4wV/kQHrbktM1A+um/DC1GGLR3ajJ6lCMn22XB0G2ccGo4Rjmd/IBMyM2MAv6l0inm/l5gEBC0/wLJ3V7QFaekikCnySTUqKY2f
H9i5GWQr3qhzQow1oAKAXI3zBlb7m/JPd8U5Cuvqrh8MtPhyi/HSU/7GKPTNACKlq8cc/TwQHBwcHBwcHBwcHB8ebwLbkTa4o/QZqbslgfvftj
Y9CnFΔRlk1lhPweXi?nii_eWnfΔ7+px6F3i77snpa\z7OTa!6fl!HOPdv2RrCf2Y95Wai li7E2e7a_E II I03nksHRwchBpna+atEo7 Inidcu1z1eRnk
```

게시글 등록

본문 내용

마크다운 형식을 사용해서

다양하게 사용자가 직접 꾸밀 수 있음

외부 이미지도 불러올 수 있음



게시글 등록

게시글 등록 후, 등록된 모습

자바 백엔드 개발 도와드립니다.

#자바 #백엔드 #스프링부트

상세 설명 후기

마크다운 형식을 사용해서

다양하게 사용자가 직접 꾸밀 수 있음

외부 이미지도 불러올 수 있음



수정 삭제



parkm2ngyu00

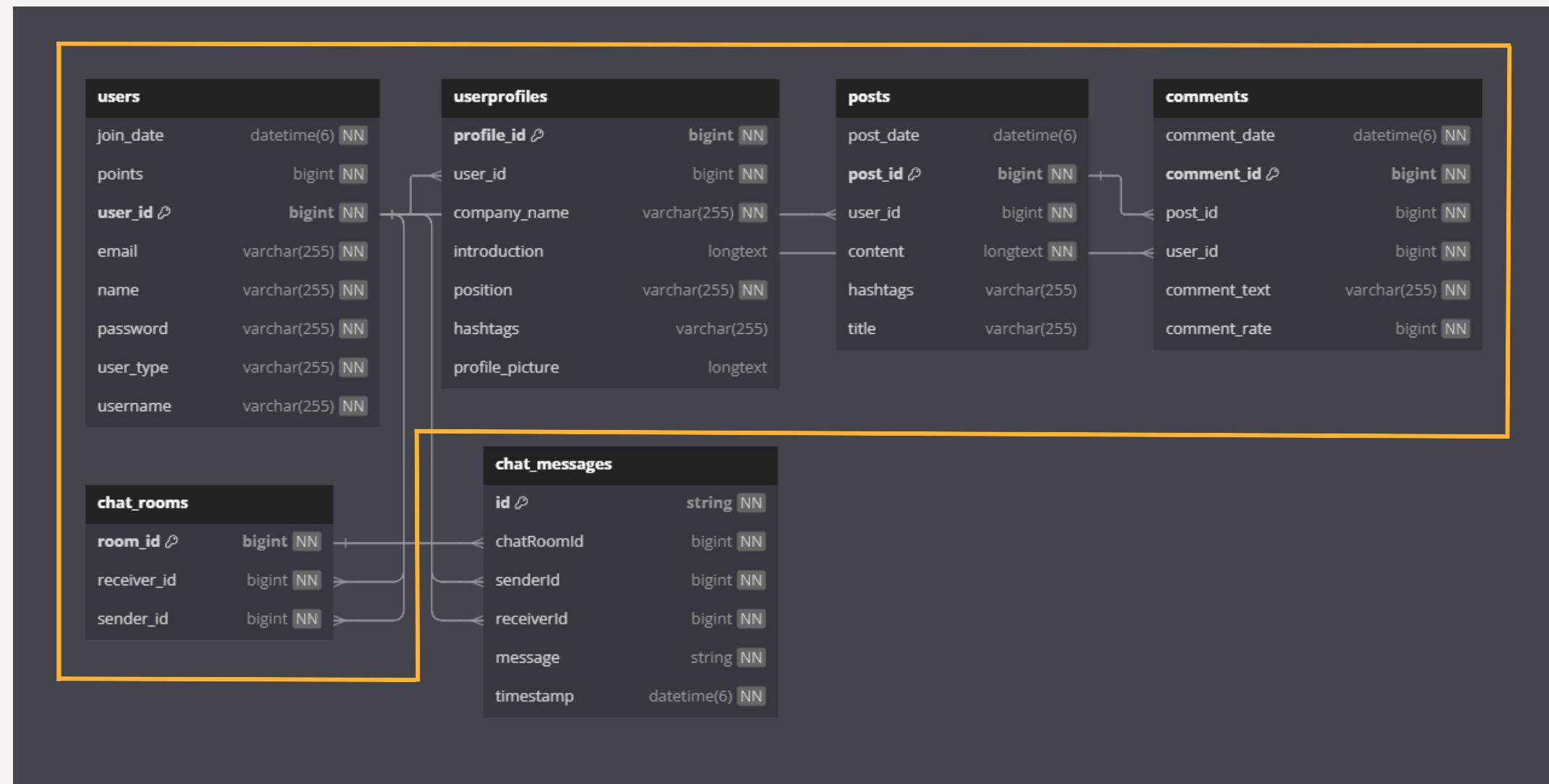
회사명을 입력해주세요
parkm2ngyu00@naver.com

#백엔드 #스프링 #리액트

프로필 보러가기

채팅하기

최종 결과물 설명 - DBMS (MySQL)



DBMS를 두개로 분리

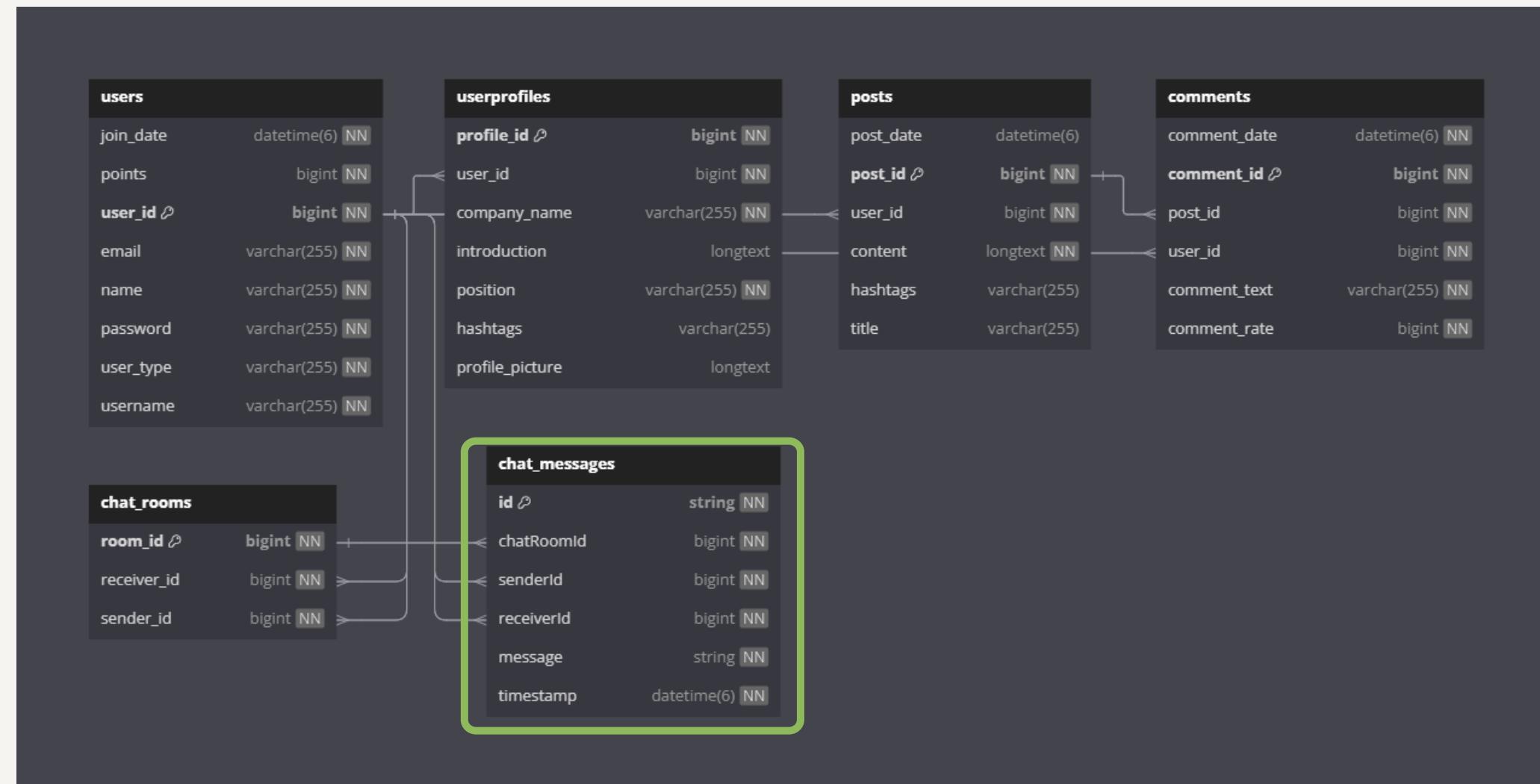
<MySQL>

-조인 연산을 통해 데이터를 자주 조회해야 하는 값들에 대해서는 RDBMS인 MySQL 사용

프로젝트에서 사용하는 db table

->users, userprofiles, posts, comments, chat_rooms

최종 결과물 설명 -DBMS (mongoDB)



DBMS를 두개로 분리

<mongoDB>

- 채팅 메세지를 담는 부분은 NoSQL인 mongoDB를 사용
- mongoDB는 고속 쓰기 작업에 최적화되어있어 채팅 메시지와 같은 빈번한 데이터 삽입 작업에서 높은 성능을 제공

프로젝트에서 사용하는 mongoDB table
->chat_messages

API 명세서

기능	Method.	Description	URL
BoardController.java	POST	새 게시판 생성	/api/boards/{userId}
BoardController.java	GET	게시판 검색	/api/boards/search
BoardController.java	GET	모든 게시판 목록 조회	/api/boards
BoardController.java	GET	특정 ID 게시판 상세 정보 조회	/api/boards/{boardId}
BoardController.java	PUT	특정 ID 게시판 정보 수정	/api/boards/{boardId}
BoardController.java	DELETE	특정 ID 게시판 삭제	/api/boards/{boardId}
ChatController.java	GET	특정 사용자의 모든 채팅 조회	/api/chats/{userId}
ChatController.java	POST	두 사용자 간 채팅방 생성or조회	/api/chats/{userId}/{yourId}
CommentController.java	POST	게시판에 댓글 생성	/api/boards/{boardId}/comments
CommentController.java	GET	특정 게시판의 모든 댓글 조회	/api/boards/{boardId}/comments
CommentController.java	PUT	특정 게시판의 댓글 수정	/api/boards/{boardId}/comments
CommentController.java	DELETE	특정 게시판의 댓글 삭제	/api/boards/{boardId}/comments
CommentController.java	GET	특정 게시판의 댓글 요약 정보 조회	/api/boards/{boardId}/comments/summary
ProfileController.java	POST	사용자 ID 기반 새 프로필 생성	/api/profiles/{userId}
ProfileController.java	GET	사용자 ID로 프로필 조회	/api/profiles/{userId}
ProfileController.java	PUT	사용자 ID로 프로필 수정	/api/profiles/{userId}
ProfileController.java	DELETE	사용자 ID로 프로필 삭제	/api/profiles/{userId}
RecommendationController.java	GET	사용자 ID 기반 게시판 추천 목록 조회	/api/boards/recommend
UserController.java	POST	사용자 등록	/api/users/register
UserController.java	POST	사용자 로그인&인증	/api/users/login

1. 로그인 & 회원가입

최종 결과물 설명



- Spring Security를 활용하여, 로그인 및 회원가입 기능을 구현

- 회원가입 시, 아이디/비밀번호/성함/이메일/유저타입(취준생or현직자)에 대한 값을 사용자로부터 받아서 DB에 저장

- 아래 테이블에 값이 들어가고, 기본키인 user_id는 정수값으로 그 값이 할당될 때마다 1씩 증가하며, 자동으로 할당

```
mysql> desc users;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+
| join_date | datetime(6) | NO |     | NULL    |                |
| points | bigint | NO |     | NULL    |                |
| user_id | bigint | NO | PRI | NULL    | auto_increment |
| email | varchar(255) | NO |     | NULL    |                |
| name | varchar(255) | NO |     | NULL    |                |
| password | varchar(255) | NO |     | NULL    |                |
| user_type | varchar(255) | NO |     | NULL    |                |
| username | varchar(255) | NO | UNI | NULL    |                |
+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

최종 결과물 설명

1. 로그인 & 회원가입

회원가입

POST /api/users/register

Request Body

```
{
  "username": "exampleUser",
  "password": "examplePassword",
  "name": "Example Name",
  "email": "example@example.com",
  "user_type": "USER"
}
```



Response Body

```
{
  "user_id": 1,
  "username": "exampleUser",
  "name": "Example Name",
  "email": "example@example.com",
  "user_type": "USER",
  "points": 0,
  "join_date": "2024-05-27 04:01"
}
```

<회원가입>

-사용자로부터 비밀번호를 받아 저장할 때, 그대로 DB에 저장하게 되면 큰 보안상의 문제가 생김. 그래서 Spring Security에서 제공하는 해시 함수를 통해 사용자가 입력한 비밀번호를 암호화하여 DB에 저장하는 형태로 구현

로그인

POST /api/users/login

Request Body

```
{
  "username": "exampleUser",
  "password": "examplePassword"
}
```



Response Body

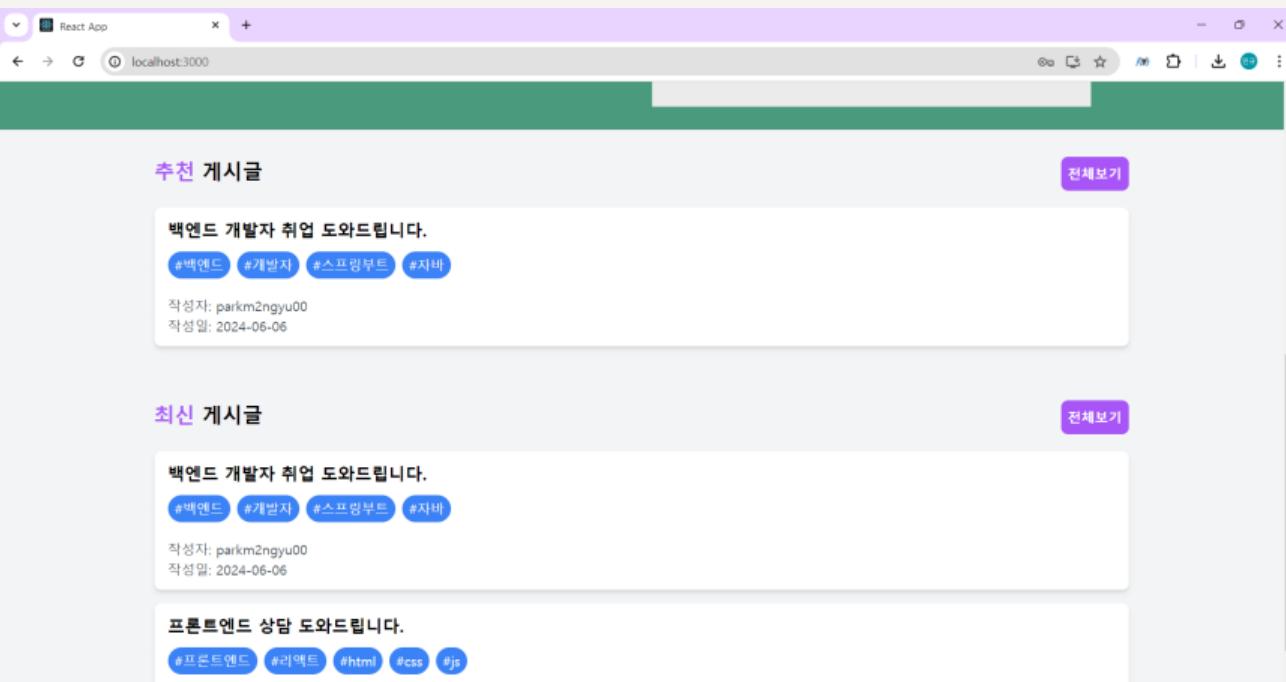
```
{
  "user_id": 1,
  "username": "exampleUser",
  "name": "Example Name",
  "email": "example@example.com",
  "user_type": "USER",
  "points": 0,
  "join_date": "2024-05-27 04:01"
}
```

<로그인>

-사용자가 입력한 값을 해쉬함수로 한번 암호화하고, 그 암호화한 값과 DB에 저장된 이미 암호화된 값을 비교하는 형태로 구현.
(이때, 보안상의 이유로 GET요청 대신 POST 요청 사용)

최종 결과물 설명

2. 추천



`GET /api/boards/recommend?userId=1`

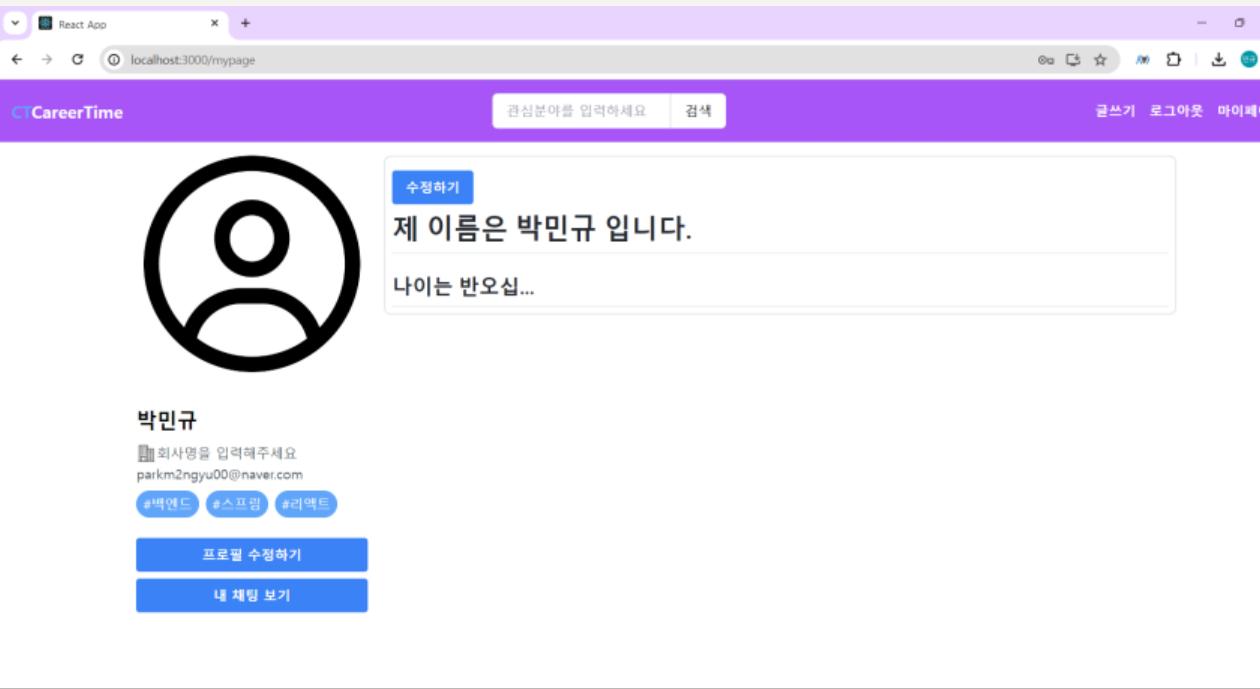
Response Body

```
[
  {
    "post_id": 1,
    "title": "Example Title",
    "hashtags": ["#example", "#board"],
    "content": "Example content for the board.",
    "postdate": "2024-05-27",
    "userinfo": {
      "user_id": 1,
      "username": "exampleUser",
      "usercompany": "Example Company",
      "useremail": "example@example.com",
      "userimage": "data:image/png;base64",
      "userinterest": ["#java", "#spring", "#backend"]
    }
  },
  {
    "post_id": 2,
    "title": "Example Title2",
    "hashtags": ["#example", "#board"],
    "content": "This is an example content.",
    "postdate": "2024-05-27",
    "userinfo": {
      "user_id": 3,
      "username": "exampleUser3",
      "usercompany": "Example Company3",
      "useremail": "example3@example.com",
      "userimage": "data:image3/png;base64",
      "userinterest": ["#python", "#django", "#backend"]
    }
  }
]
```

사용자의 프로필에 기반하여 게시물을 추천하는 기능을 제공.
주요 작업은 사용자 프로필을 가져와 사용자가 관심 있는 해시태그를 찾고, 해당 해시태그를 포함하는 게시물을 찾고, 이를 사용자에게 추천함.

최종 결과물 설명

3.유저 프로필



CRUD 기능

mysql> desc userprofiles;						
Field	Type	Null	Key	Default	Extra	
profile_id	bigint	NO	PRI	NULL		auto_increment
user_id	bigint	NO	UNI	NULL		
company_name	varchar(255)	NO		NULL		
introduction	longtext	YES		NULL		
position	varchar(255)	NO		NULL		
hashtags	varchar(255)	YES		NULL		
profile_picture	longtext	YES		NULL		

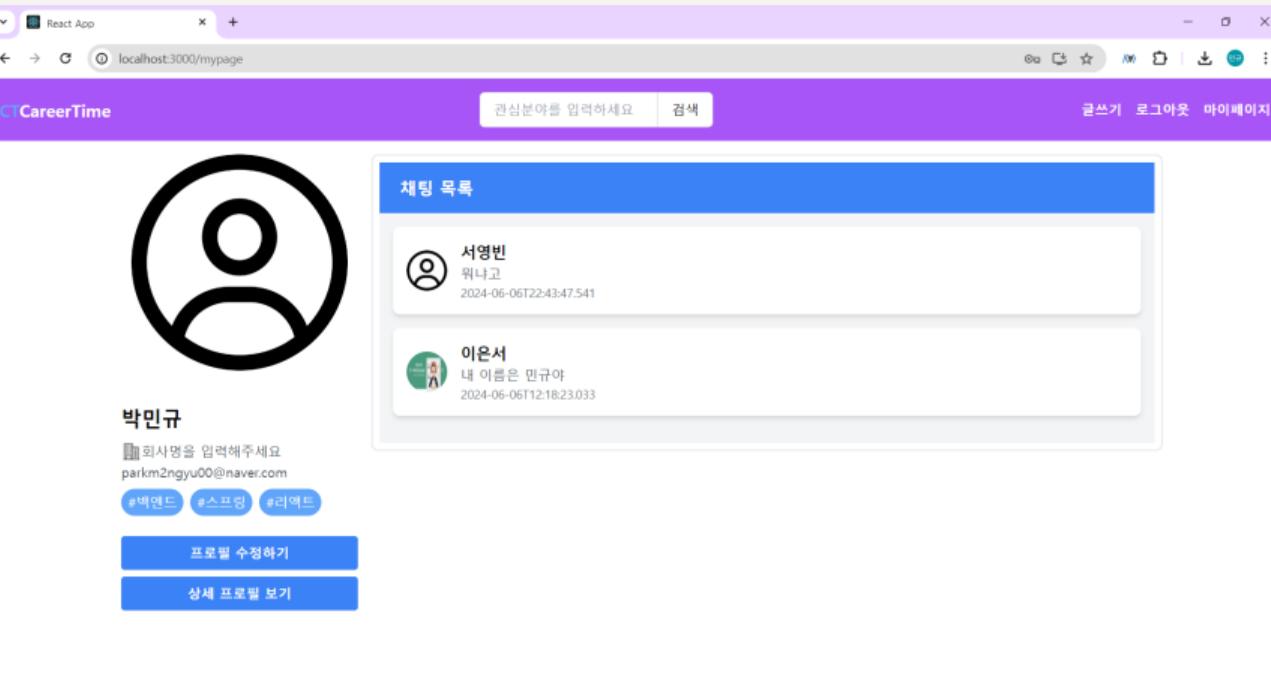
7 rows in set (0.00 sec)

사진 저장:

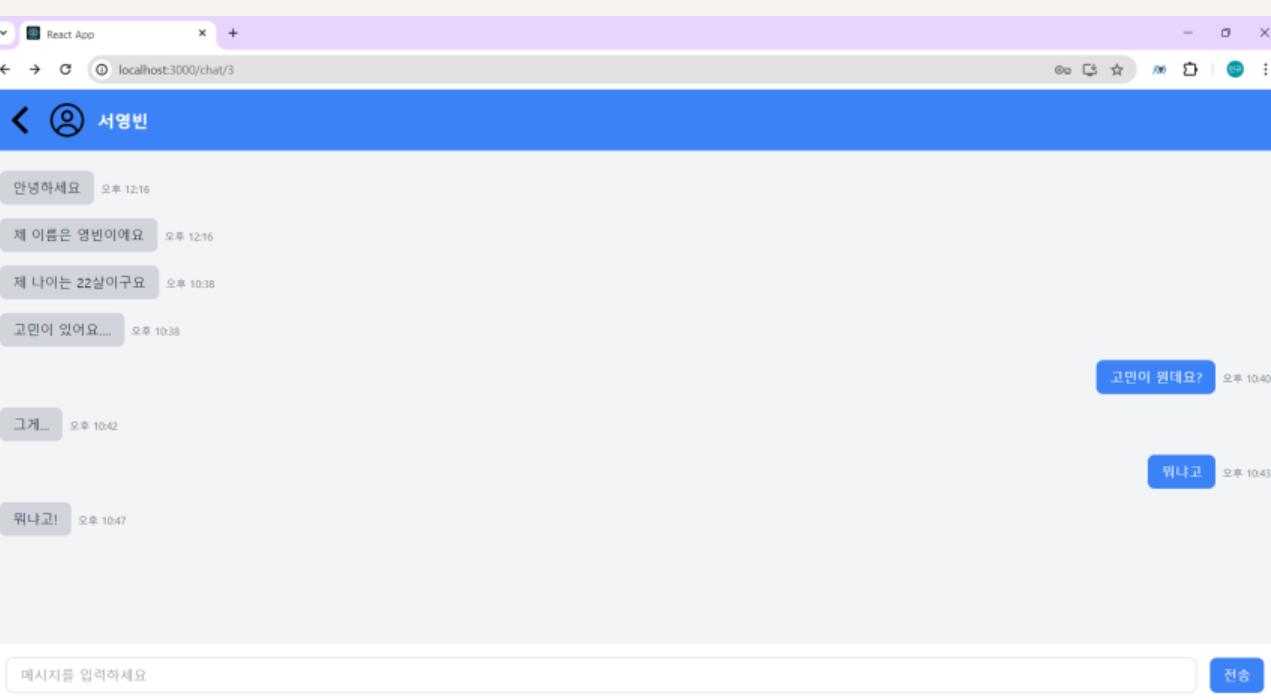
1. 프론트 단에서 프로필 사진을 base64로 인코딩 (이미지 → 문자열)
2. 그 문자열을 DB(MySQL)에 저장
3. 프로필 사진 조회하면 프론트로 인코딩된 문자열을 보냄
4. 프론트 단에서 문자열을 이미지로 디코딩하여 사용

최종 결과물 설명

4. 1:1 채팅



- OSI 7계층에 위치한 웹소켓 프로토콜을 통해 일대일 채팅 기능을 구현
- Spring Boot에서 제공하는 메세지 브로커를 사용하여 연결 안정성을 높임



두 유저가 나누는 대화는 바로 DB(몽고DB)에 저장되므로,
서로 오고간 대화를 영구적으로 조회할 수 있음.

MySQL 상에서 두 유저가 나누는 채팅 방에 대한 정보만 저장되어 있는 테이블

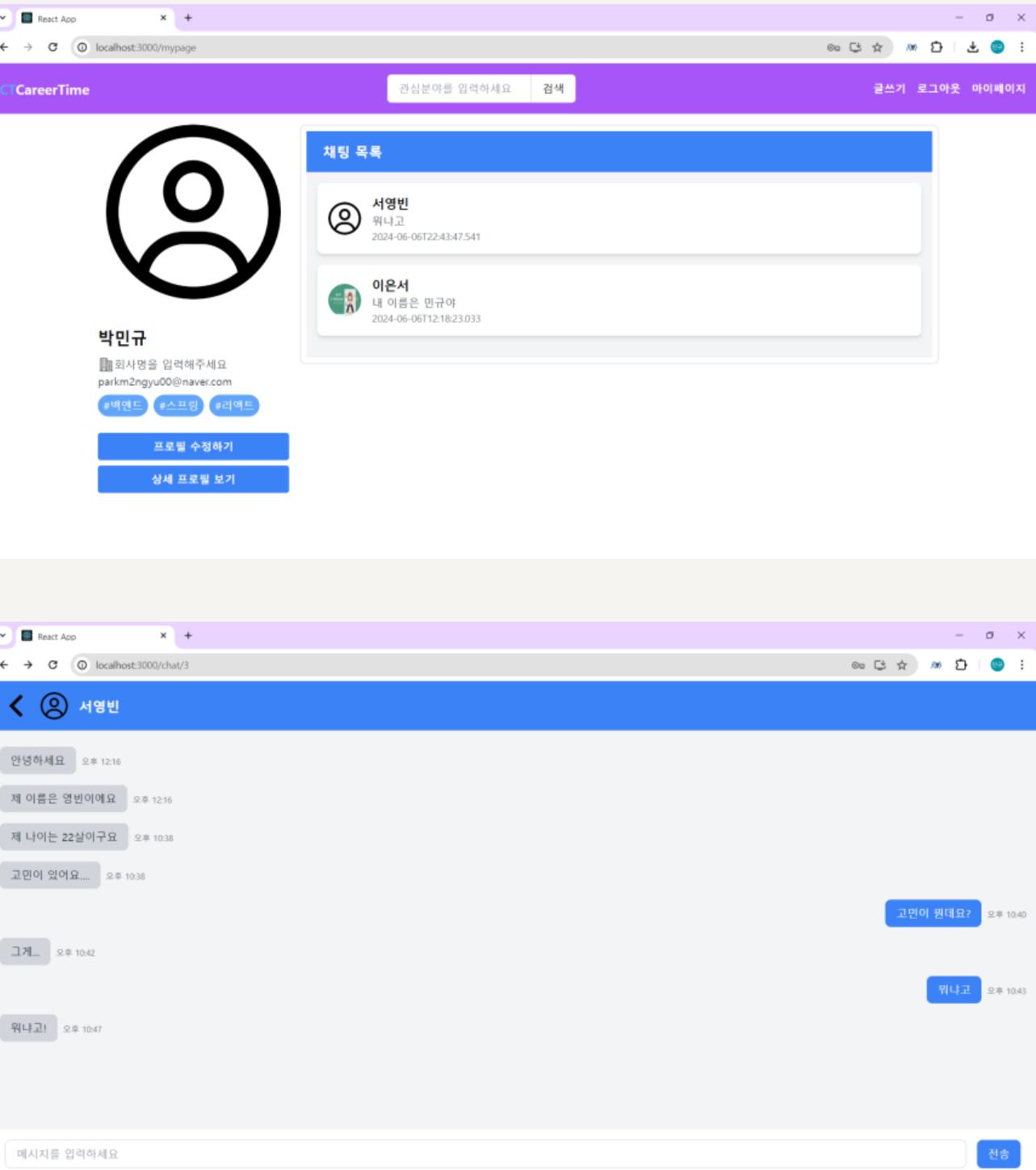
mysql> desc chat_rooms;						
Field	Type	Null	Key	Default	Extra	
room_id	bigint	NO	PRI	NULL	auto_increment	
receiver_id	bigint	NO	MUL	NULL		
sender_id	bigint	NO	MUL	NULL		

3 rows in set (0.00 sec)

최종 결과물 설명

4. 1:1 채팅

커리어타임



두 유저가 주고 받은 채팅 내용은 MongoDB에 저장

```
_id: ObjectId('6664b12064529653243a73cb')
chatRoomId: 7
senderId: 170
receiverId: 171
message: "안녕하세요"
timestamp: 2024-06-08T19:29:36.172+00:00
_class: "kr.ac.dankook.ace.careertime.domain.ChatMessage"
```

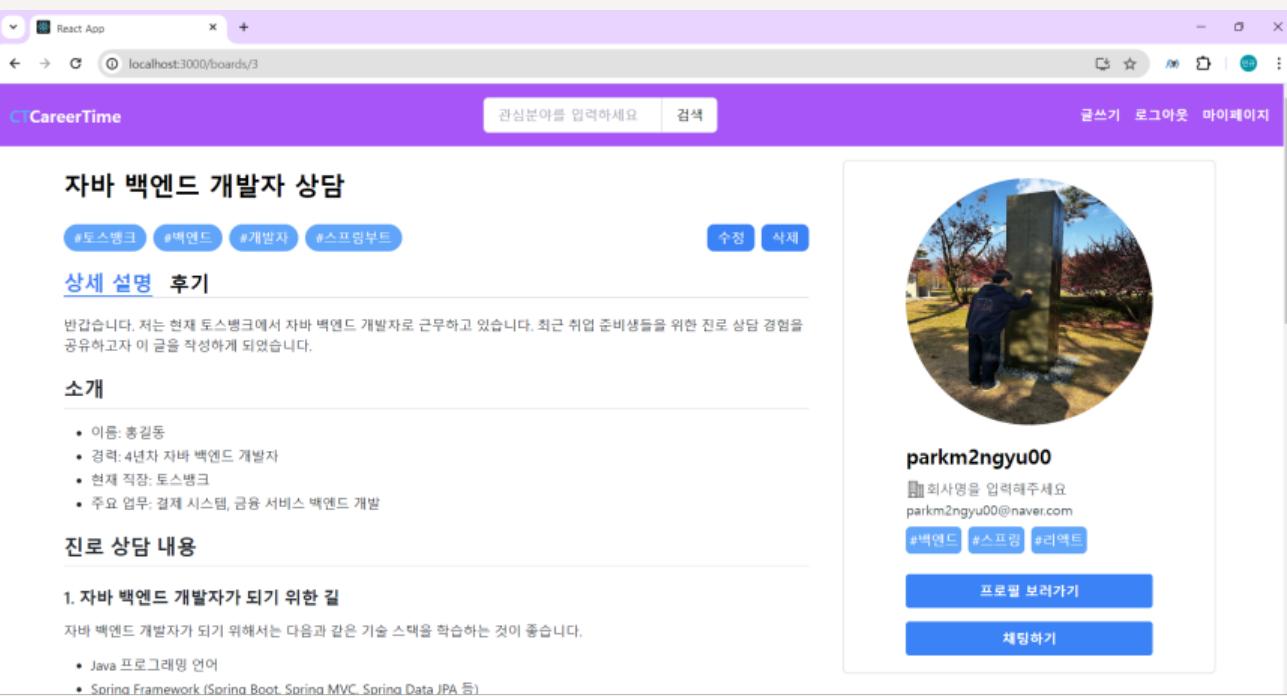
```
_id: ObjectId('6664b12564529653243a73cc')
chatRoomId: 8
senderId: 171
receiverId: 170
message: "반갑습니다"
timestamp: 2024-06-08T19:29:41.467+00:00
_class: "kr.ac.dankook.ace.careertime.domain.ChatMessage"
```

```
GET /api/boards/recommend?userId=1
Response Body
[
  {
    "post_id": 1,
    "title": "Example Title",
    "hashtags": ["#example", "#board"],
    "content": "Example content for the board.",
    "postdate": "2024-05-27",
    "userinfo": {
      "user_id": 1,
      "username": "exampleUser",
      "usercompany": "Example Company",
      "useremail": "example@example.com",
      "userimage": "data:image/png;base64",
      "userinterest": ["#java", "#spring", "#backend"]
    }
  },
  {
    "post_id": 2,
    "title": "Example Title2",
    "hashtags": ["#example", "#board"],
    "content": "This is an example content.",
    "postdate": "2024-05-27",
    "userinfo": {
      "user_id": 3,
      "username": "exampleUser3",
      "usercompany": "Example Company3",
      "useremail": "example3@example.com",
      "userimage": "data:image3/png;base64",
      "userinterest": ["#python", "#django", "#backend"]
    }
  }
]
```

최종 결과물 설명

5. 게시글

커리어타임



DB 테이블

```
mysql> desc posts;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| post_date | datetime(6) | YES | NULL | NULL | auto_increment |
| post_id | bigint | NO | PRI | NULL | |
| user_id | bigint | NO | MUL | NULL | |
| content | longtext | NO | NULL | NULL | |
| hashtags | varchar(255) | YES | NULL | NULL | |
| title | varchar(255) | YES | NULL | NULL | |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

최종 결과물 설명

6. 댓글(후기)

자바 백엔드 개발자 상담

#도스뱅크 #백엔드 #개발자 #스프링부트

상세 설명 후기

상품 후기
총 0개의 후기

0 ⭐⭐⭐⭐⭐

후기 남기기

★★★★★

후기 내용을 작성해주세요.

제출하기

상세 설명 후기

상품 후기
총 2개의 후기

3 ⭐⭐⭐⭐⭐

후기 남기기

★★★★★

후기 내용을 작성해주세요.

제출하기

박민규 ★★★★★
정말 많은 도움이 됩니다.

박민규 ★★★★★
별로임 ㅋ

DB 테이블

```
mysql> desc comments;
```

Field	Type	Null	Key	Default	Extra
comment_date	datetime(6)	NO		NULL	
comment_id	bigint	NO	PRI	NULL	auto_increment
post_id	bigint	NO	MUL	NULL	
user_id	bigint	NO	MUL	NULL	
comment_text	varchar(255)	NO		NULL	
comment_rate	bigint	NO		NULL	

6 rows in set (0.00 sec)

후기에서 별점의 평균을 계산하여 제공하는 API

개시글의 댓글 요약 정보 조회

```
GET /api/boards/{boardId}/comments/summary
```

Response Body

```
{
  "reviewCount": 1,
  "averageRating": 5.0,
  "reviewList": [
    {
      "comment_id": 1,
      "username": "exampleUser",
      "useremail": "example@example.com",
      "name": "Example Name",
      "user_type": "USER",
      "user_id": 1,
      "rating": 5,
      "comment": "This is a sample comment."
    }
  ]
}
```

감사합니다.