500만 반려가구와 함께

# 댕묘앞

** 본 프로젝트는 RESTful API 형식으로 제작되었습니다.*


**이메일**

haneul.skye.cheon@gmail.com

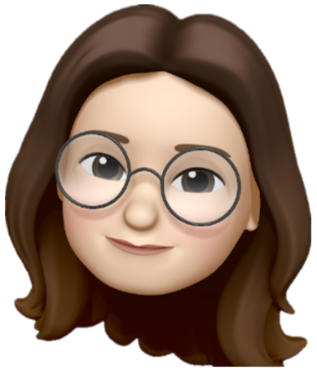
**Git-Hub**

https://github.com/soundspeedskye
https://github.com/myj1217/daengmyoapp


**시연 영상**

https://www.youtube.com/watch?v=HuRwqg0PDUY

ROLE



*상상,*
*그 이상의 개발자*

DAENGMYOAPP

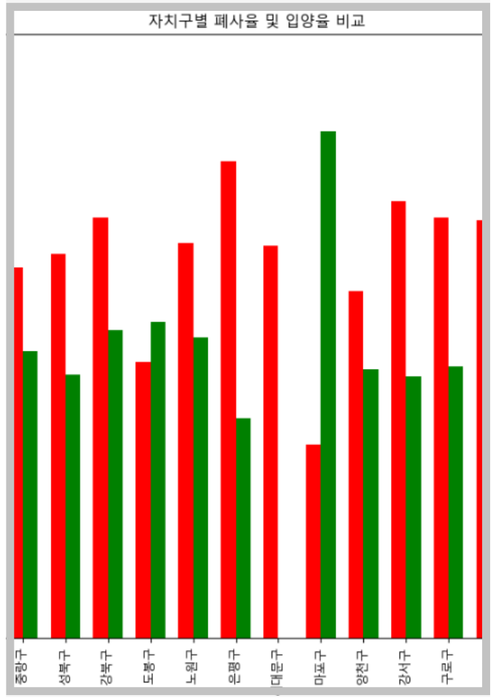
| 01 | 02 | 03 | 04 |
|---|---|---|---|
| 커뮤니티 | 댓글 | 공지사항 | 1:1 이메일 문의 |
| CRUD 페이지<br>CRUD 기능<br>리스트 페이징<br>파일 처리<br>이미지 썸네일 | CRUD 페이지<br>CRUD 기능 | 권한 제한 구현<br>CRUD 페이지<br>CRUD 기능<br>리스트 페이징<br>파일 처리 | 메일 송신 페이지<br>메일 송신 기능 |

BACKGROUND

# 01 프로젝트 기획

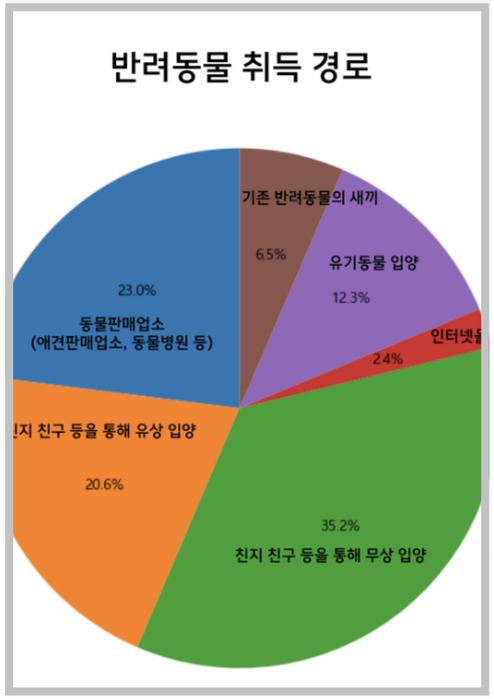
**01 유기동물 폐사율 80%**

## 서울시 유기동물보호 현황 통계




**02 반려동물 입양율 12%**

## 반려동물 취득 경로 통계




**03 반려가구 증가**

## 반려가구 증가하지만 유기동물 수도 많아

반려가구 500만 시대의 그늘... [슬기로운 반려생활 ⑤]

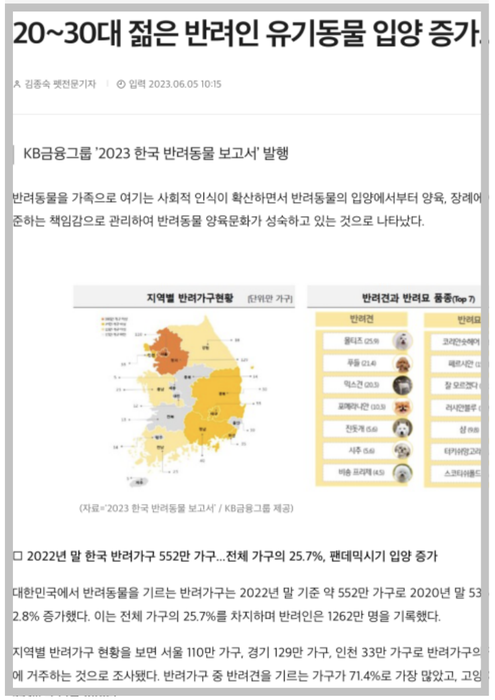
김수영 기자

입력 2024.03.08 06:58 수정 2024.03.08 09:57

전국 유기동물 6년째 10만마리 이상
보호소 1곳당 평균 474마리 관리 '포화'
민간은 운영비 부담 더 커...41.4% '사비'
입양 중요성에도 여전히 '펫숍 구매' 많아
2020년 18.6%→2023년 24%로 증가

**04 20~30대 입양비율 증가**

## 젊은 반려인이 입양 주도



20~30대 젊은 반려인 유기동물 입양 증가

김종숙 펫전문기자 입력 2023.06.05 10:15

KB금융그룹 '2023 한국 반려동물 보고서' 발행

반려동물을 가족으로 여기는 사회적 인식이 확산하면서 반려동물의 입양에서부터 양육, 장례에 준하는 책임감으로 관리하여 반려동물 양육문화가 성숙하고 있는 것으로 나타났다.

지역별 반려가구현황

반려견과 반려묘 품종(Top 7)

(자료=2023 한국 반려동물 보고서 / KB금융그룹 제공)

□ 2022년 말 한국 반려가구 552만 가구...전체 가구의 25.7%, 팬데믹시기 입양 증가

대한민국에서 반려동물을 기르는 반려가구는 2022년 말 기준 약 552만 가구로 2020년 말 53 2.8% 증가했다. 이는 전체 가구의 25.7%를 차지하며 반려인은 1262만 명을 기록했다.

지역별 반려가구 현황을 보면 서울 110만 가구, 경기 129만 가구, 인천 33만 가구로 반려가구의 에 거주하는 것으로 조사됐다. 반려가구 중 반려견을 기르는 가구가 71.4%로 가장 많았고, 고양

# 02

# 참고 레퍼런스

## 01 국가동물보호정보시스템

유기동물 신고, 공고, 동물등록,
농장동물 복지 등 동물보호 업무
전반 통합관리 시스템

## 02 포인핸드

전국 보호소 유기동물에게
가족을 찾아주는
플랫폼 서비스

# 03

# 개발 환경

## 서 버

Apache Tomcat

MySQL

## 개발 툴

IJ

Visual Studio

React

Redux

spring boot

tailwindcss

## 개발 언어

Java

JavaScript

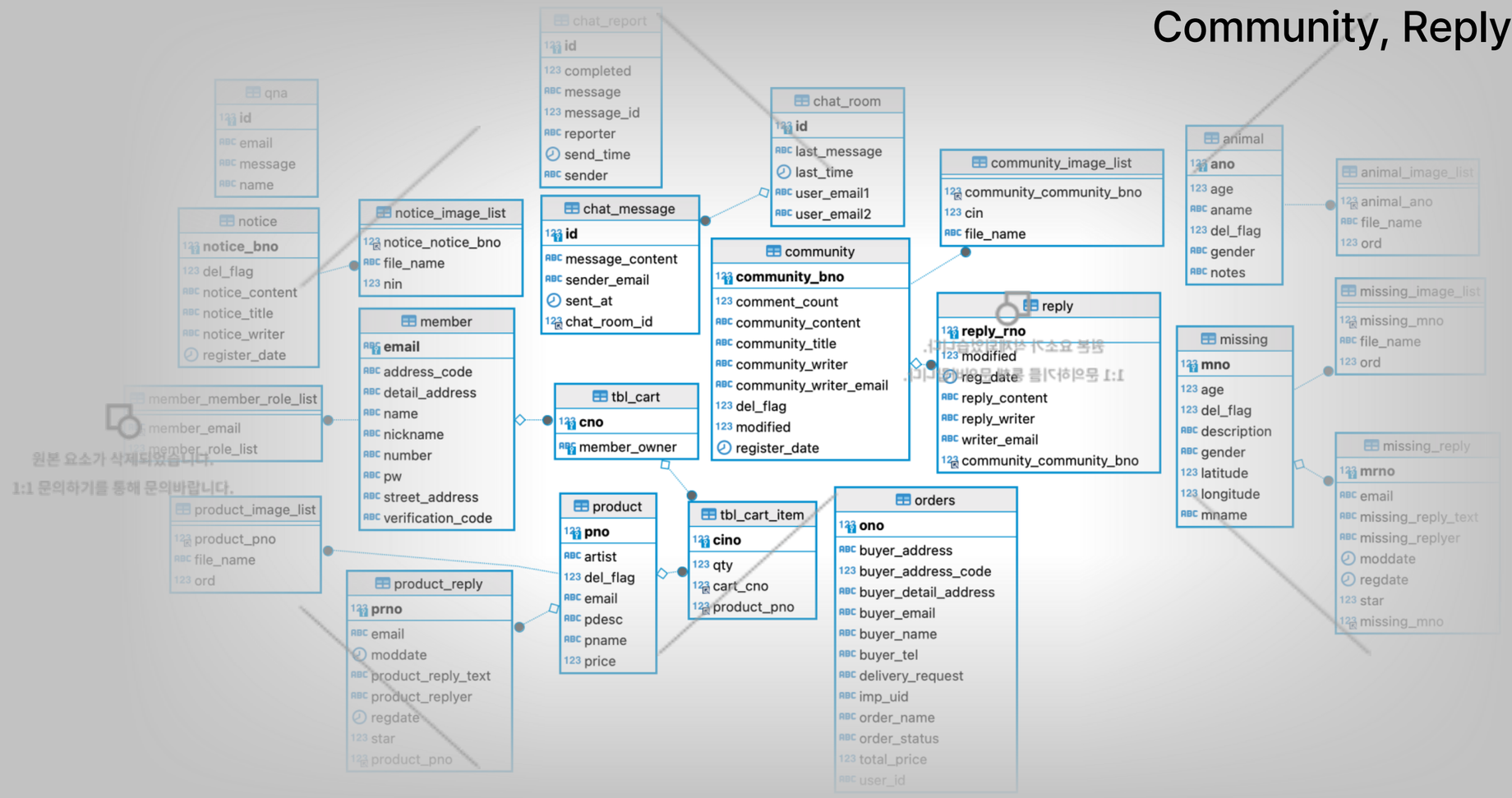
Spring Data JPA

STRUCTURE

# 04 파일 구조

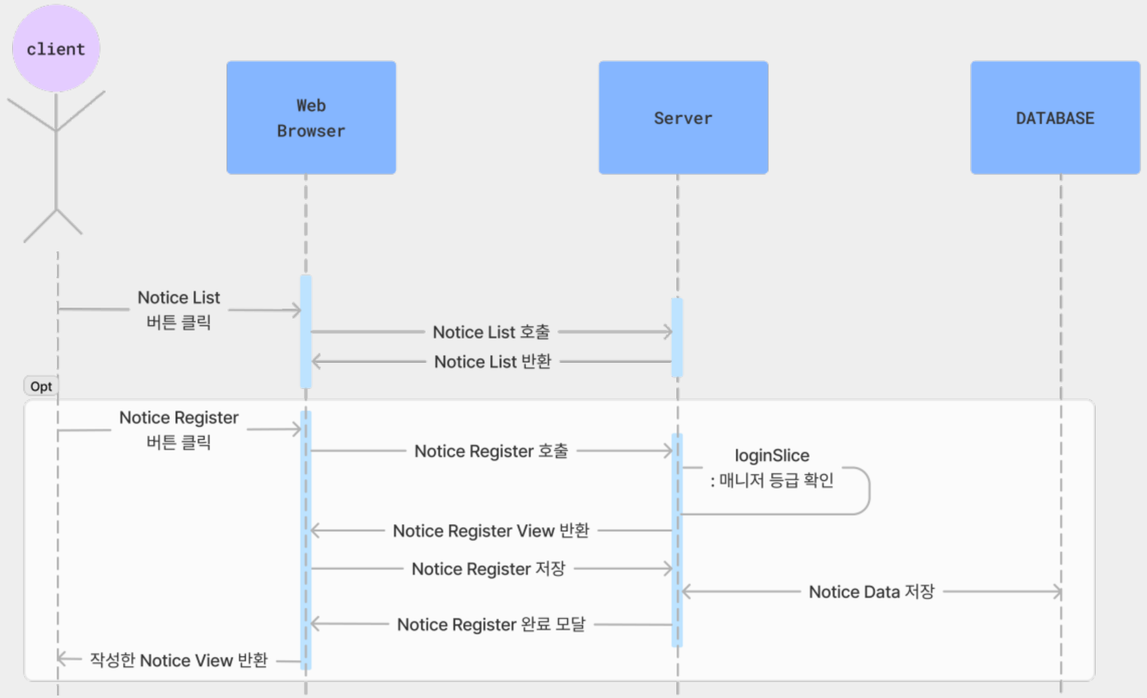
Notice, QnA

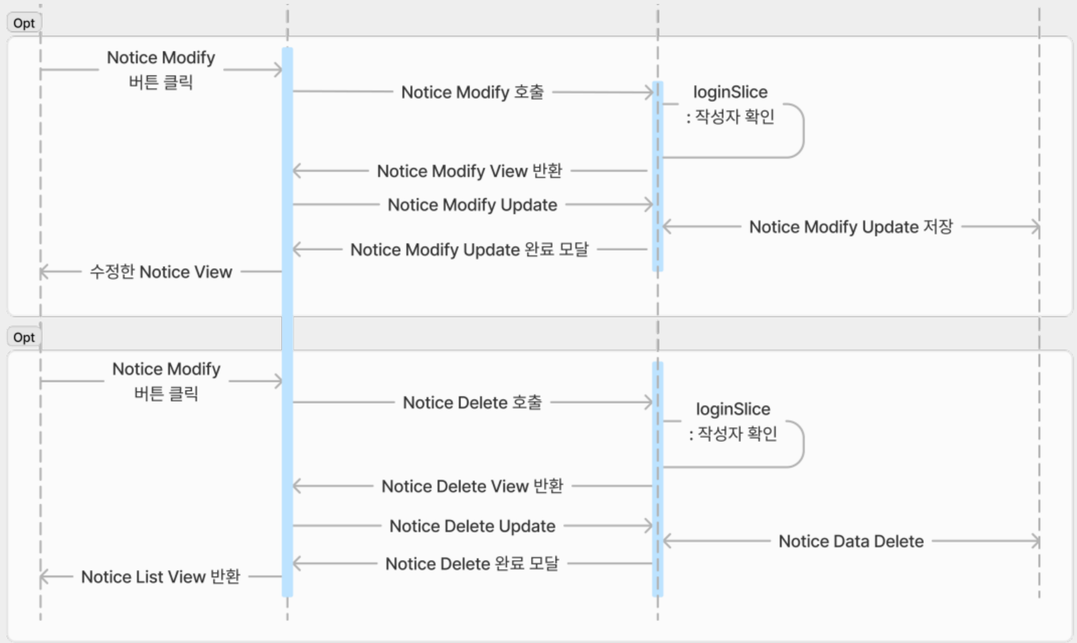
Community, Reply



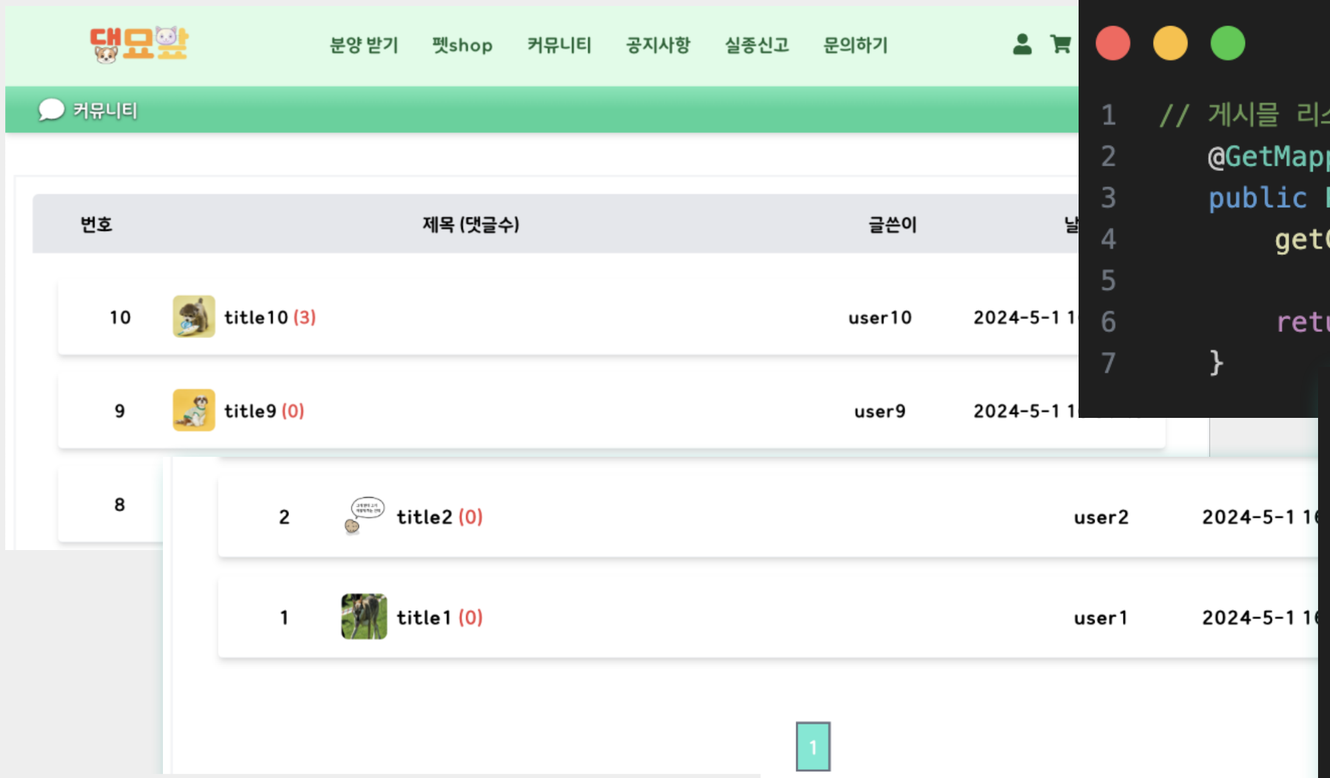
공지사항 UML DIAGRAM

# 05 구현 화면



# 05 구현 화면




BACK

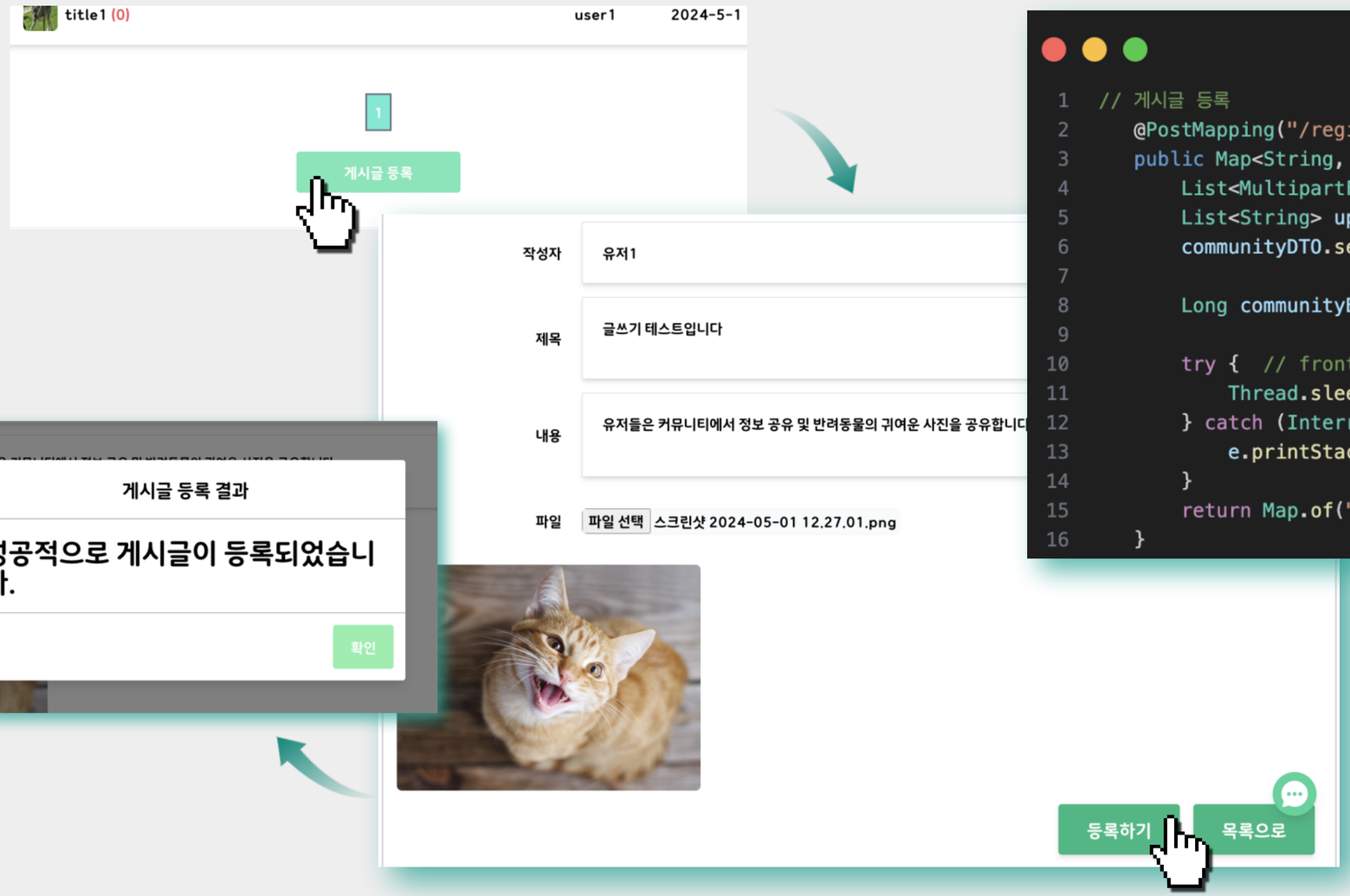
```java
// 게시물 리스트보기
    @GetMapping("/list")
    public PageResponseDTO<CommunityDTO>
        getCommunityList(PageRequestDTO pageRequestDTO) {

        return communityService.getCommunityList(pageRequestDTO);
    }
```

```javascript
const ListCommunityComponent = () => {
  const { exceptionHandle } = useCustomLogin();
  const { page, size, refresh, moveToList, moveToRead } = useCustomMove();
  const navigate = useNavigate();

  const [serverData, setServerData] = useState(initState);

  useEffect(() => {
    communityList({ page, size })
      .then((data) => {
        setServerData(data);
      })
      .catch((err) => exceptionHandle(err));
  }, [page, size, refresh]);

  const handleClickReg = useCallback(() => {
    navigate({ pathname: "../register" });
  });
```

FRONT

1. 최신 등록된 순서대로 커뮤니티 리스트 업데이트
2. 커뮤니티 리스트 하단에 게시글 등록 버튼

# 05 구현 화면

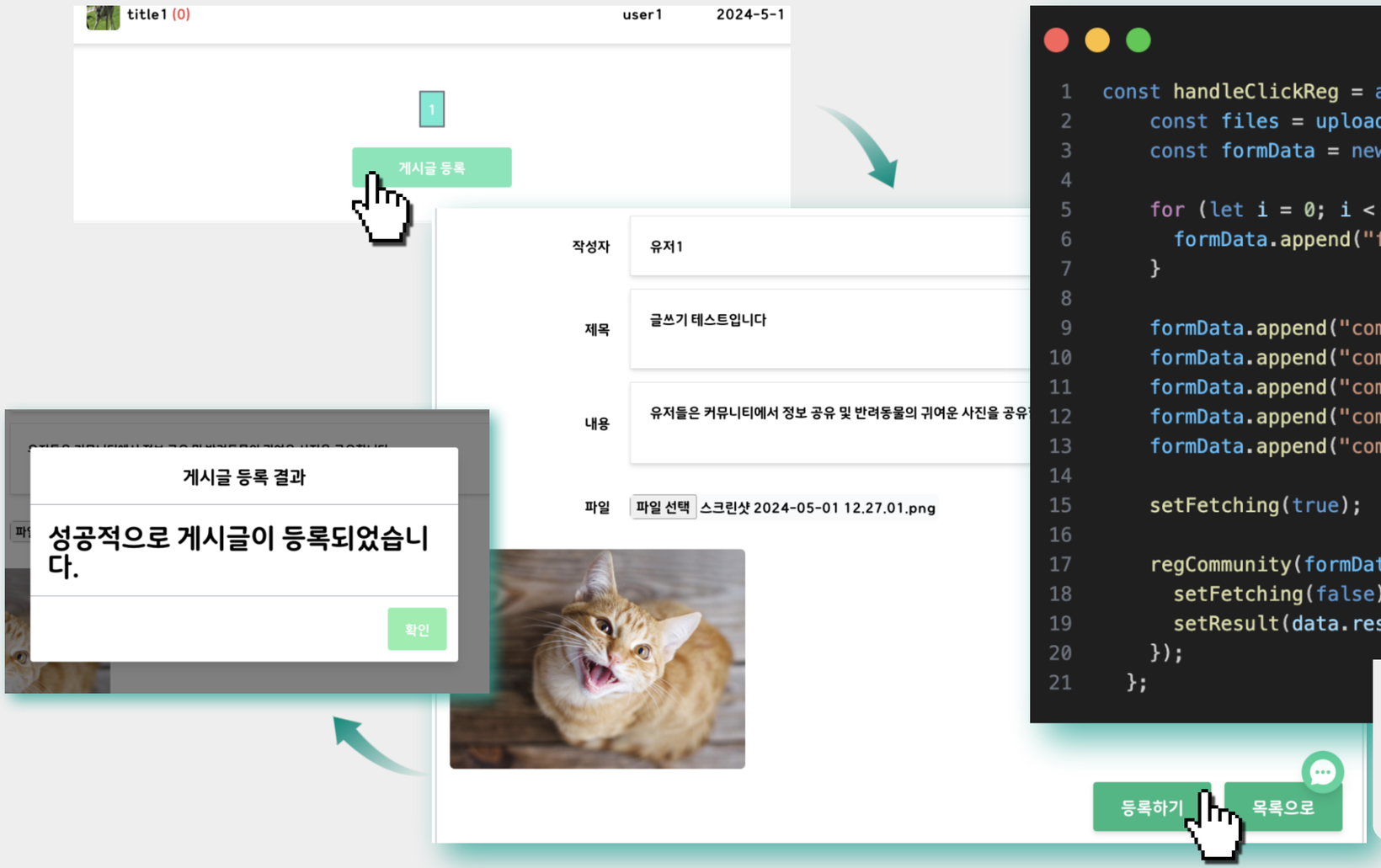



BACK

```java
// 게시글 등록
    @PostMapping("/register")
    public Map<String, Long> regCommunity(CommunityDTO communityDTO) {
        List<MultipartFile> files = communityDTO.getFiles();
        List<String> uploadFileNames = fileUtil.saveFiles(files);
        communityDTO.setUploadFileNames(uploadFileNames);

        Long communityBno = communityService.regCommunity(communityDTO);

        try {  // front의 fetching 진행모달창 1초동안 보이기
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return Map.of("result", communityBno);
    }
```

1. 게시글 등록 시 파일 포함 가능
2. 등록 완료 시 모달창 확인

# 05 구현 화면

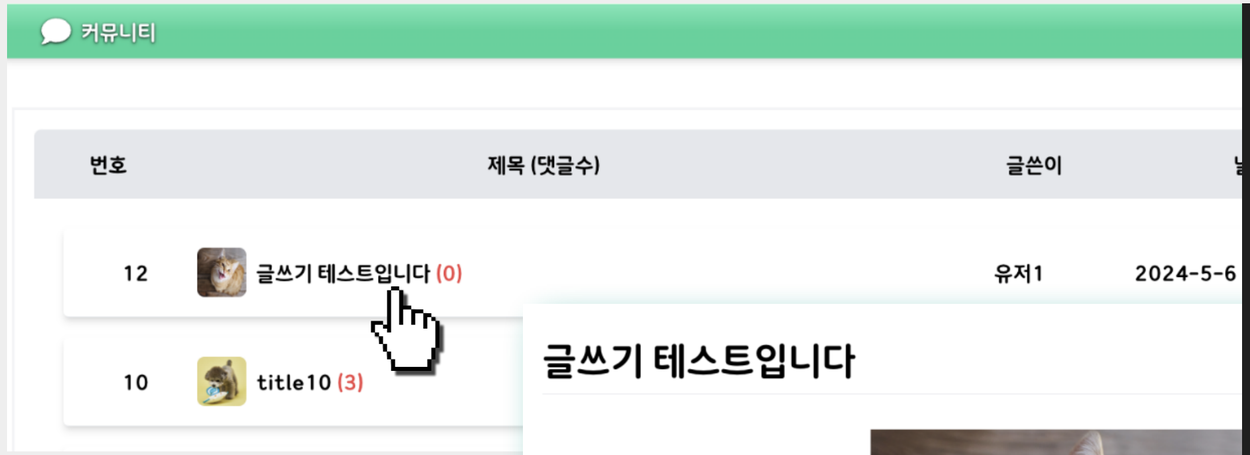



FRONT

```
const handleClickReg = async (e) => {
    const files = uploadRef.current.files;
    const formData = new FormData();

    for (let i = 0; i < files.length; i++) {
      formData.append("files", files[i]);
    }

    formData.append("communityTitle", community.communityTitle);
    formData.append("communityContent", community.communityContent);
    formData.append("communityWriter", loginState.nickname);
    formData.append("communityWriterEmail", loginState.email);
    formData.append("commentCount", 0);

    setFetching(true);

    regCommunity(formData).then((data) => {
      setFetching(false);
      setResult(data.result);
    });
  };
```

1. 게시글 등록 시 첨부 이미지 미리보기 가능
2. loginState로 작성자 식별

# 05 구현 화면



BACK

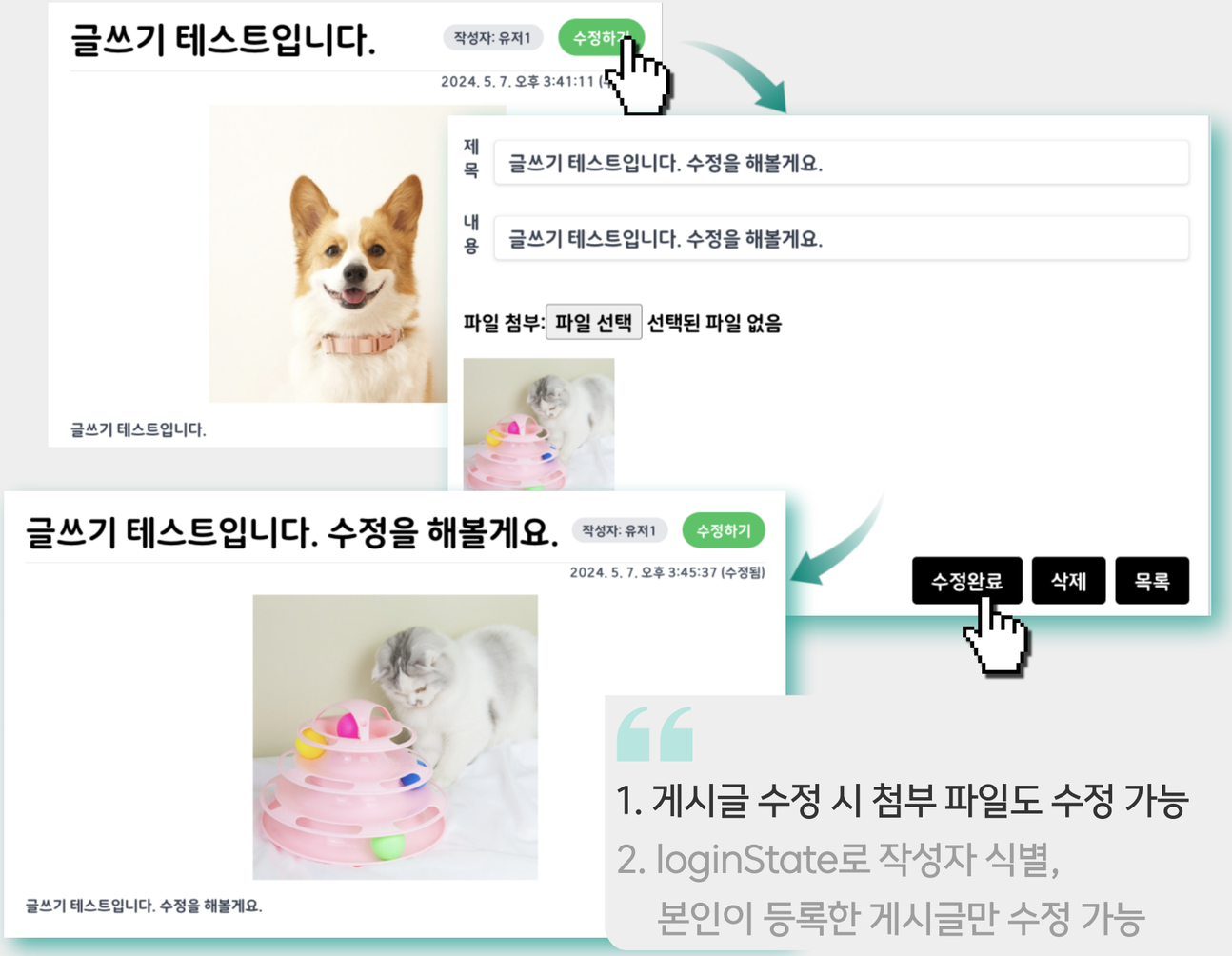
```java
// 게시글 상세보기
    @GetMapping("/read/{communityBno}")
    public CommunityDTO getCommunity
        (@PathVariable(name = "communityBno") Long communityBno) {
        return communityService.getCommunity(communityBno);
    }

    @GetMapping("/view/{fileName}")
    public ResponseEntity<Resource> viewFileGET
        (@PathVariable String fileName) {
        return fileUtil.getFile(fileName);
    }

```

FRONT

```java
useEffect(() => {
  setFetching(true);
  getCommunity(communityBno)
    .then((data) => {
      setCommunity(data);
      setFetching(false);
    })
    .catch((error) => {
      console.error("Error fetching community:", error);
      setFetching(false);
    });
}, [communityBno]);
```

1. PathVariable로 게시글 번호 식별
2. 파일을 포함한 게시글 상세보기

# 05 구현 화면




1. 게시글 수정 시 첨부 파일도 수정 가능
2. loginState로 작성자 식별,
   본인이 등록한 게시글만 수정 가능

BACK

```java
@PutMapping("/{communityBno}")
    public Map<String, String> modCommunity
    (@PathVariable(name = "communityBno")
            Long communityBno, CommunityDTO communityDTO) {
        communityDTO.setCommunityBno(communityBno);
        CommunityDTO oldCommunityDTO = communityService.getCommunity(communityBno);
        //기존의 파일들 (데이터베이스에 존재하는 파일들 - 수정 과정에서 삭제되었을 수 있음)
        List<String> oldFileNames = oldCommunityDTO.getUploadFileNames();
        //새로 업로드 해야 하는 파일들
        List<MultipartFile> files = communityDTO.getFiles();
        //새로 업로드되어서 만들어진 파일 이름들
        List<String> currentUploadFileNames = fileUtil.saveFiles(files);
        //화면에서 변화 없이 계속 유지된 파일들
        List<String> uploadedFileNames = communityDTO.getUploadFileNames();

        //유지되는 파일들  + 새로 업로드된 파일 이름들이 저장해야 하는 파일 목록이 됨
        if (currentUploadFileNames != null && currentUploadFileNames.size() > 0) {
            uploadedFileNames.addAll(currentUploadFileNames);
        }
        communityService.modCommunity(communityDTO);

        if (oldFileNames != null && oldFileNames.size() > 0) {
            //지워야 하는 파일 목록 찾기
            //예전 파일들 중에서 지워져야 하는 파일이름들
            List<String> removeFiles = oldFileNames
                    .stream()
                    .filter(fileName ->
                    uploadedFileNames.indexOf(fileName) == -1)
                            .collect(Collectors.toList());
            //실제 파일 삭제
            fileUtil.deleteFiles(removeFiles);
        }
        return Map.of("RESULT", "SUCCESS");
    }
```

# 05 구현 화면

FRONT

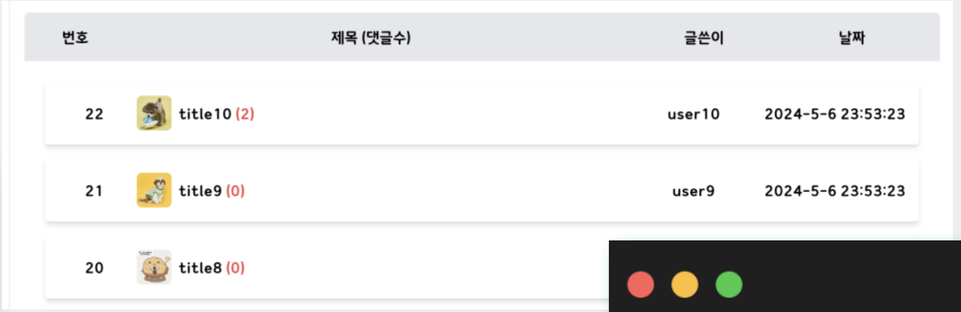
```
const ModCommunityPage = () => {
  const { communityBno } = useParams();
  const loginState = useSelector((state) => state.loginSlice);
  if (!loginState.email) {
    return <Navigate to="/member/login" />;
  }
  return (
    <div className="w-full bg-white">
      <BasicMenu />
      <ModCommunityComponent communityBno=
    </div>
  );
};
```

FRONT

```
const ModCommunityComponent = ({ communityBno }) => {
  const [community, setCommunity] = useState(initState);
  const [result, setResult] = useState(null);
  const { moveToRead, moveToList } = useCustomMove();
  const navigate = useNavigate(); // useNavigate 훅 추가

  const [fetching, setFetching] = useState(false);
  const uploadRef = useRef();

  useEffect(() => {
    setFetching(true);
    getCommunity(communityBno).then((data) => {
      setCommunity(data);
      setFetching(false);
    });
  }, [communityBno]);

  const handleChangeCommunity = (e) => {
    setCommunity({ ...community, [e.target.name]: e.target.value });
  };

  const deleteOldImages = (imageName) => {
    setCommunity({
      ...community,
      uploadFileNames: community.uploadFileNames.filter(
        (fileName) => fileName !== imageName
      ),
```

1. 게시글 수정 시 첨부 파일도 수정 가능
2. loginState로 작성자 식별,
   본인이 등록한 게시글만 수정 가능

# 05 구현 화면




1. 게시글 수정 페이지에서 삭제 가능
2. 삭제 완료 후 커뮤니티 리스트로 이동

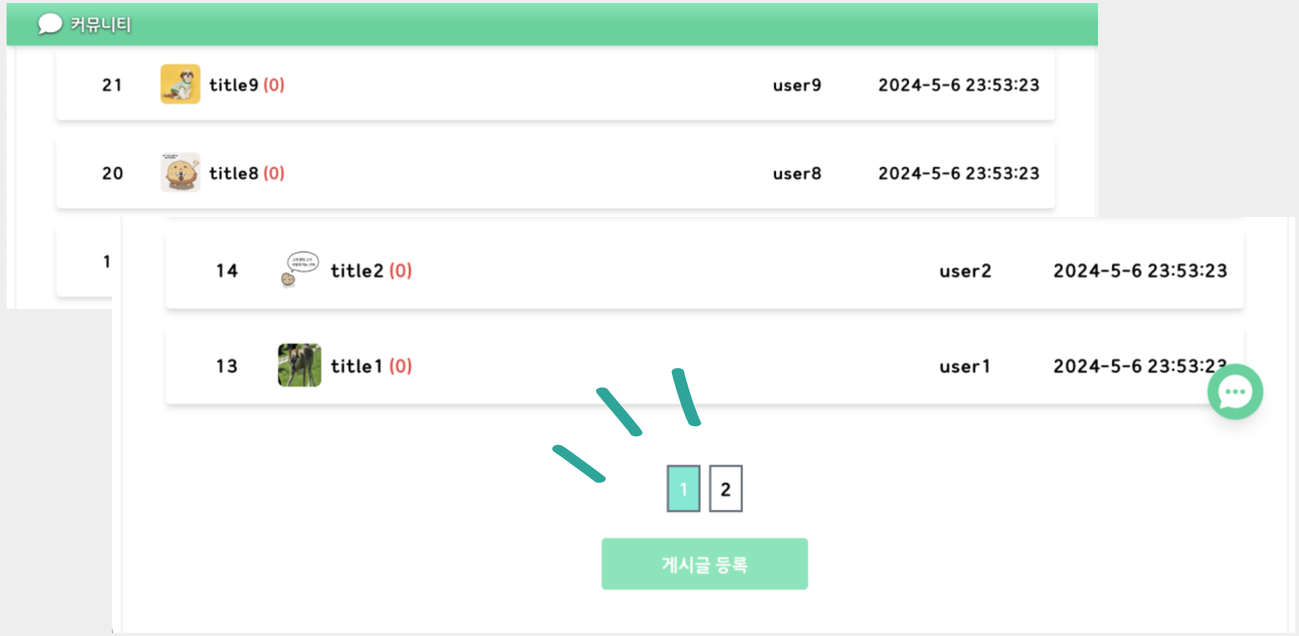
FRONT

```
// 삭제
const handleClickDelete = () => {
  setFetching(true);
  delCommunity(communityBno).then(() => {
    setResult("Deleted");
    setFetching(false);
```

BACK

```
// 게시글 삭제
    @DeleteMapping("/{communityBno}")
    public Map<String, String> delCommunity
        (@PathVariable(name = "communityBno") Long communityBno) {

        //삭제해야할 파일들 알아내기
        List<String> oldFileNames = communityService
            .getCommunity(communityBno).getUploadFileNames();
        communityService.delCommunity(communityBno);
        fileUtil.deleteFiles(oldFileNames);

        return Map.of("RESULT", "SUCCESS");
    }
```

# 05 구현 화면

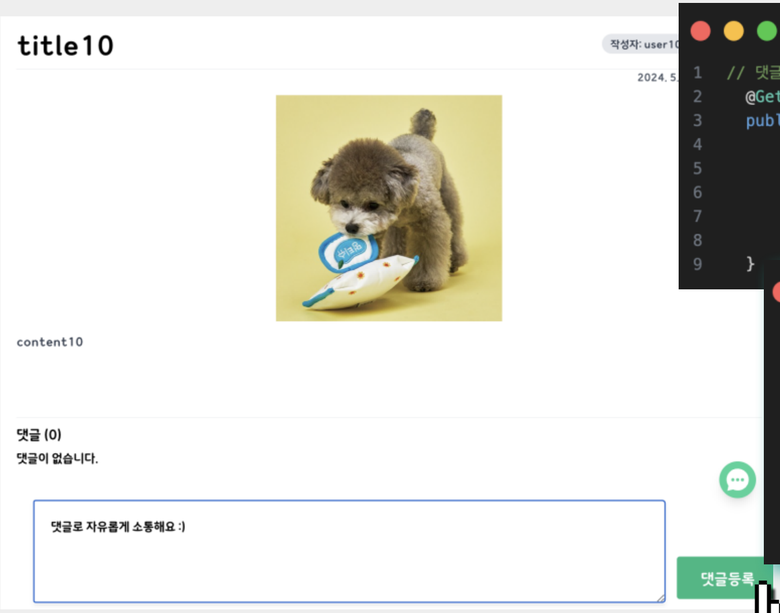



1. 게시글 10개마다 리스트 생성

BACK

```java
// 커뮤니티 페이징
    @Override
    public PageResponseDTO<CommunityDTO>
        getCommunityList(PageRequestDTO pageRequestDTO) {

        Pageable pageable = PageRequest.of(
                pageRequestDTO.getPage() - 1,  //페이지 시작 번호가 0부터 시작하므로
                pageRequestDTO.getSize(),
                Sort.by("communityBno").descending());

        Page<Object[]> result = communityRepository.selectList(pageable);

        List<CommunityDTO> dtoList = result.get().map(arr -> {
            Community community = (Community) arr[0];
            CommunityImage communityImage = (CommunityImage) arr[1];

            CommunityDTO communityDTO = CommunityDTO.builder()
                    .communityBno(community.getCommunityBno())
                    .communityTitle(community.getCommunityTitle())
                    .communityContent(community.getCommunityContent())
                    .communityWriter(community.getCommunityWriter())
                    .communityWriterEmail(community.getCommunityWriterEmail())
                    .commentCount(community.getCommentCount())
                    .regDate(community.getRegDate())
                    .delFlag(community.isDelFlag())
                    .modified(community.isModified())
                    .build();

            // CommunityImage 객체가 null이 아닌 경우에만 이미지 파일명을 설정합니다.
            if (communityImage != null) {
                String imageStr = communityImage.getFileName();
                communityDTO.setUploadFileNames(List.of(imageStr));
            }
            return communityDTO;
        }).collect(Collectors.toList());

        long totalCount = result.getTotalElements();

```

# 05 구현 화면

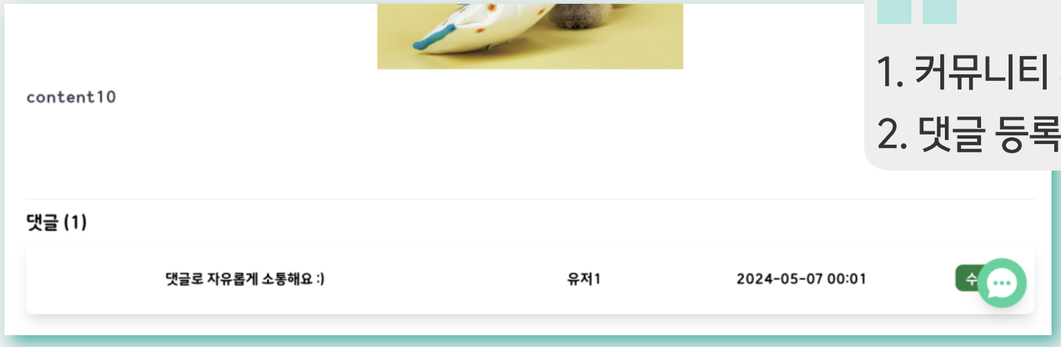



BACK

```python
// 댓글 리스트
@GetMapping("/list/{communityBno}")
public PageResponseDTO<ReplyDTO> getReplyListDto
    (@PathVariable("communityBno")Long communityBno, PageRequestDTO pageRequestDTO) {
    PageResponseDTO<ReplyDTO> responseDTO =
            replyService.getReplyList(communityBno, pageRequestDTO);

    return responseDTO;
}
```

```python
// 댓글 등록
@PostMapping(value = "/register",
        consumes = MediaType.APPLICATION_JSON_VALUE)
public Map<String, Long> regReply(@Valid @RequestBody ReplyDTO replyDTO)
    Map<String, Long> resultMap = new HashMap<>();
    Long replyRno = replyService.regReply(replyDTO);
    resultMap.put("replyRno", replyRno);
    return resultMap;
}
```

BACK

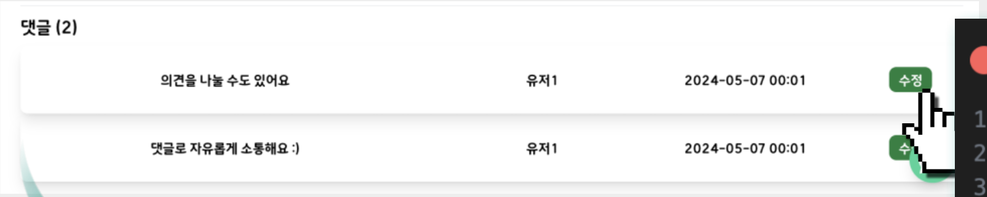


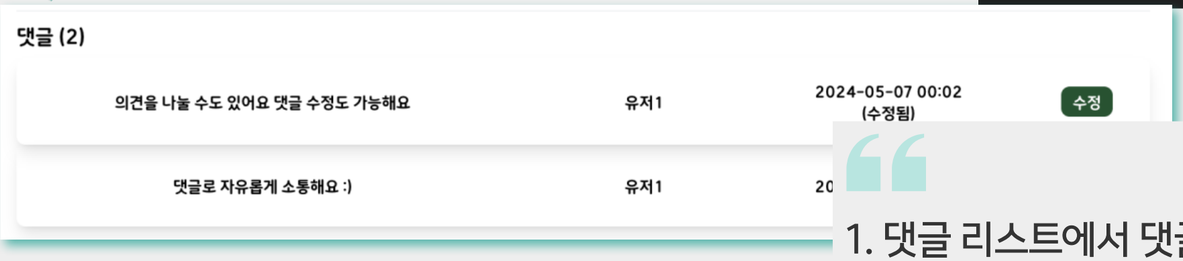

1. 커뮤니티 게시글 하단에서 댓글 작성
2. 댓글 등록 완료 후 댓글 리스트에서 확인

```python
const ReplyListComponent = ({ communityBno }) => {
  const { exceptionHandle } = useCustomLogin();
  const { refresh, moveToList } = useCustomMove();
  const [deletingReplyRno, setDeletingReplyRno] = useState(null);
  const {isLogin} = useCustomLogin();
  // 댓글 목록 및 기타 데이터 저장
  const [serverData, setServerData] = useState(initState);

  useEffect(() => {
    listReply(communityBno)
      .then((data) => {
        console.log(data);
        setServerData(data);
      })
      .catch((err) => exceptionHandle(err));
  }, [refresh]);

  // 댓글 등록
  const handleReplySubmit = (replyContent) => {
    const replyData = {
      communityBno: communityBno,
      replyContent: replyContent,
    };

    regReply(replyData)
      .then(() => {
        // 댓글 작성 후, 댓글 목록 다시 불러오기
        listReply(communityBno).then((data) => {
          setServerData(data);
        });
      })
      .catch((err) => {
        exceptionHandle(err);
      });
  };
```

FRONT

# 05 구현 화면






BACK

```java
// 댓글 수정
@PutMapping("/{replyRno}")
public void modReply(@PathVariable Long replyRno,
                     @Valid @RequestBody ReplyDTO reply
    replyDTO.setReplyRno(replyRno);
    replyService.modReply(replyDTO);
}
```

FRONT

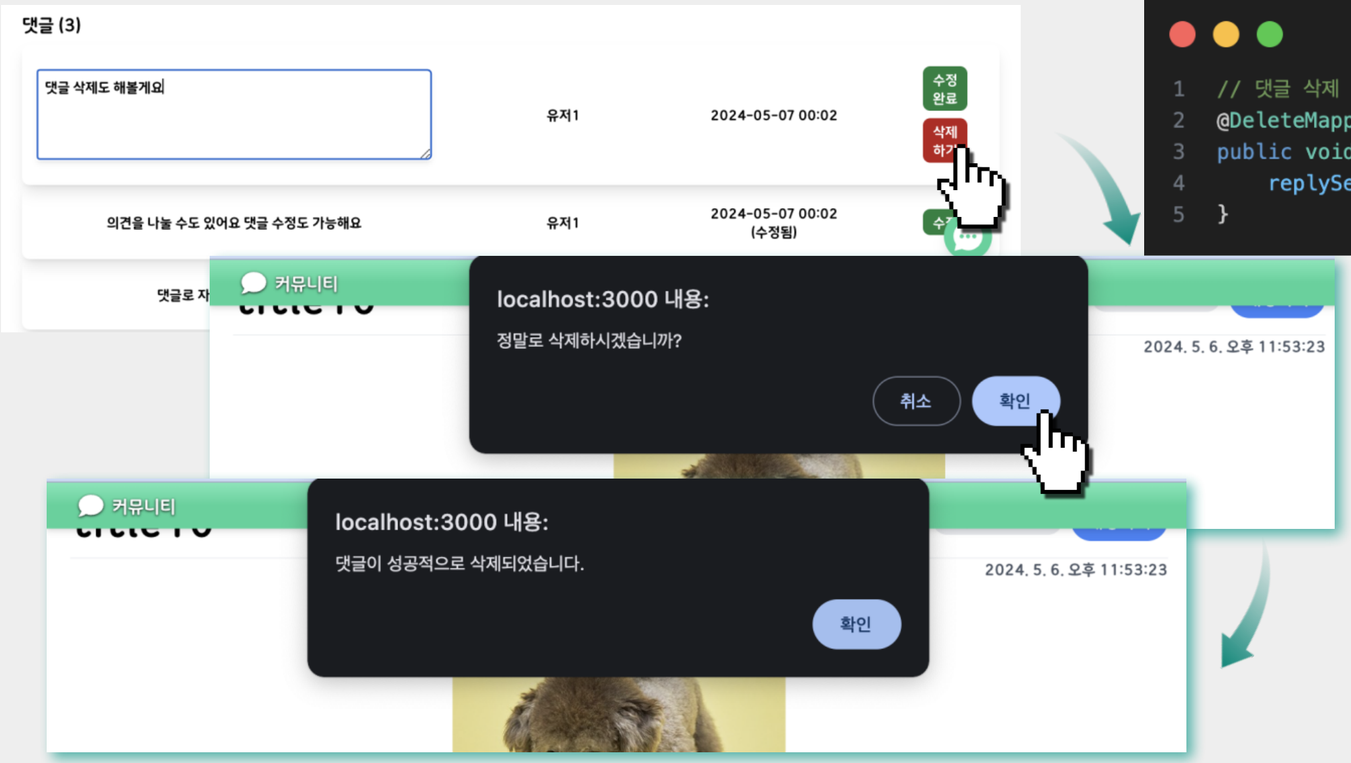
```
writerEmail === loginState.email || isAdmin ?
  <button
      className="bg-green-700 hover:bg-green-9
       text-white w-12 rounded-lg"
    onClick={modReplyHandler}
    disabled={fetching}
  >
    수정
  </button>
) : (
```

FRONT

```jsx
// 수정 버튼 함수
const handleClickModify = () => {
  const formData = new FormData();

  formData.append("communityBno", communityBno);
  formData.append("replyContent", content);
  formData.append("replyWriter", loginState.nickname);
  formData.append("writerEmail", loginState.email);
  // formData.append("regDate", regDate);

  if (!content) {
    window.alert("내용을 입력해주세요.");
    return;
  }

  setFetching(true);

  modReply(replyRno, formData)
    .then((data) => {
      listReplyRedirect();
    })
    .catch((error) => {
      console.error("수정 중 오류 발생:", error);
    })
    .finally(() => {
      setFetching(false);
    });
  window.alert("댓글이 성공적으로 수정되었습니다.");

  setModifyMode(false); // 읽기 모드로 전환
};
```

1. 댓글 리스트에서 댓글 수정 기능
2. 사용자 식별을 통해 본인이 작성한 댓글만 수정 가능

# 05 구현 화면




BACK

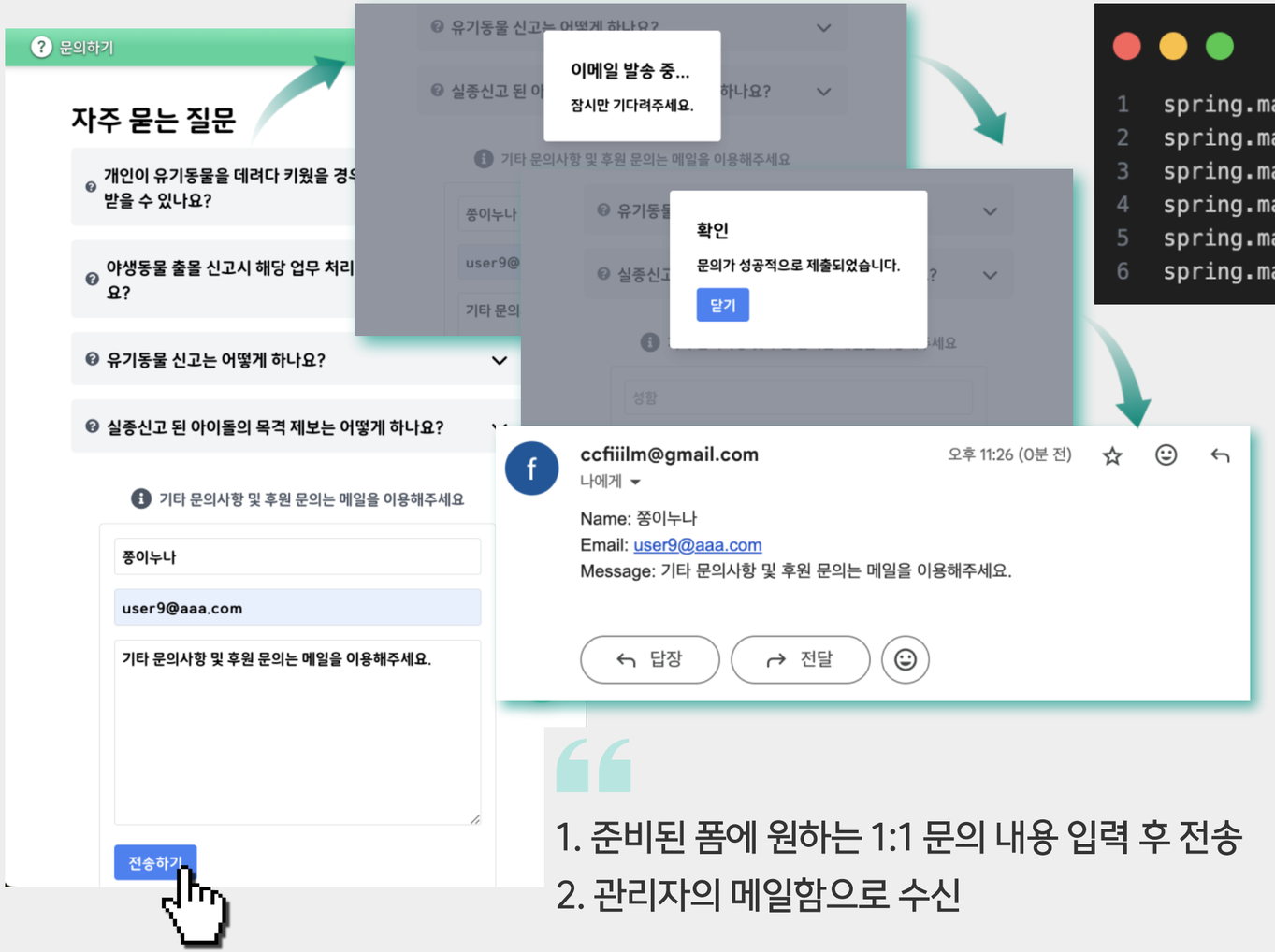
```java
// 댓글 삭제
@DeleteMapping("/{replyRno}")
public void delReply(@PathVariable Long replyRno) {
    replyService.delReply(replyRno);
}
```

FRONT

```fortran
// 삭제 핸들러
const handleClickDelete = () => {
  if (window.confirm("정말로 삭제하시겠습니까?") === false) {
    return;
  }

  setFetching(true);

  delReply(replyRno)
    .then((data) => {
      listReplyRedirect();
    })
    .catch((error) => {
      console.error("삭제 중 오류 발생:", error);
    })
    .finally(() => {
      setFetching(false);
    });

  window.alert("댓글이 성공적으로 삭제되었습니다.");
};
```

1. 댓글 수정 단계에서 삭제 가능
2. 삭제 후 댓글리스트로 이동

# 05 구현 화면




BACK

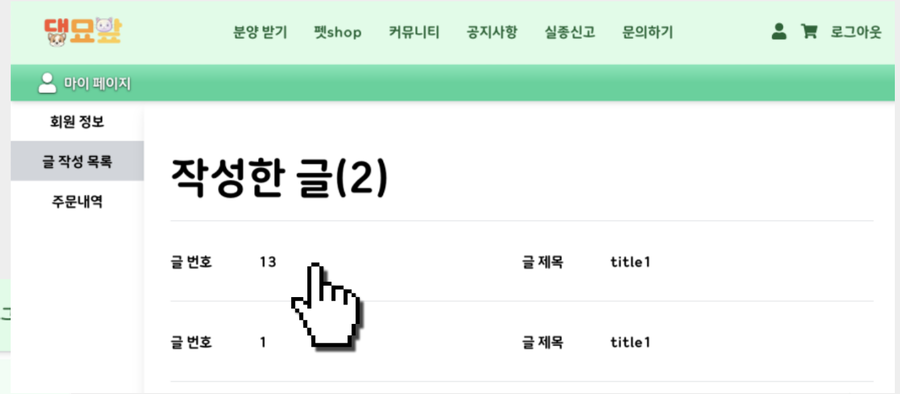
```
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=ccfiiilm@gmail.com
spring.mail.
spring.mail.
spring.mail.
```

BACK

```java
@Autowired
    private JavaMailSender mailSender;

    @Override
    public void sendEmail(String name, String email, String message)
        throws MailException {
        MimeMessage mimeMessage = mailSender.createMimeMessage();
        MimeMessageHelper helper = new MimeMessageHelper(mimeMessage);
        try {
            helper.setTo("ccfiiilm@gmail.com");
            helper.setSubject("New form submission");
            helper.setText("Name: " + name
                + "\nEmail: " + email + "\nMessage: " + message);
        } catch (MessagingException ex) {
            // 예외 처리
            ex.printStackTrace();
            // 또는 예외를 다시 던지지 않고 직접 처리
        }
        mailSender.send(mimeMessage);
    }
```

1. 준비된 폼에 원하는 1:1 문의 내용 입력 후 전송
2. 관리자의 메일함으로 수신

# 05 구현 화면



BACK

```java
// 마이페이지에서 내 글 보기
    @GetMapping("/myList")
    public PageResponseDTO<CommunityDTO> getMyList
        (PageRequestDTO pageRequestDTO, @RequestParam(name="email") String email) {
        return communityService.getMyList(pageRequestDTO, email);
    }
```

```typescript
const MyPageComponent = () => {
  const [selectedTab, setSelectedTab] = useState("profile"); // 선택된 탭
  const loginInfo = useSelector((state) => state.loginSlice);
  const {isAdmin} = useCustomLogin();

  useEffect(() => {
    // URL에서 쿼리 매개변수 가져오기
    const urlParams = new URLSearchParams(window.location.search);
    // "order" 쿼리 매개변수가 있는지 확인하고, 있다면 해당하는 탭으로 설정
    if (urlParams.has("write")) {
      setSelectedTab("write");
    }
    if (urlParams.has("order")) {
      setSelectedTab("orders");
    }
    if (urlParams.has("admin")) {
      setSelectedTab("admin");
    }
  }, []);

  // 탭을 클릭했을 때 호출되는 함수
  const handleTabClick = (tab) => {
    setSelectedTab(tab);
  };
```

FRONT

1. 메인페이지의 글 작성 목록 혹은 마이페이지 아이콘 클릭
2. 글 작성 목록에서 내가 작성한 글 확인 가능
3. 행을 클릭하면 해당 글로 이동

# 05 구현 화면

FRONT

```python
import { API_SERVER_HOST } from "./rootApi";
import jwtAxios from "../utils/jwtUtil";
import axios from "axios";

const host = `${API_SERVER_HOST}/notice`;

// 공지사항 리스트
export const noticeList = async (pageParam) => {
  const { page, size } = pageParam;
  const res = await axios.get(`${host}/list`, {
    params: { page: page, size: size },
  });

  return res.data;
};
```

BACK

```python
@RestController
@Log4j2
@RequiredArgsConstructor
@RequestMapping(value = "/notice")

public class NoticeController {

    private final NoticeService noticeService;
    private final CustomFileUtil fileUtil;
```

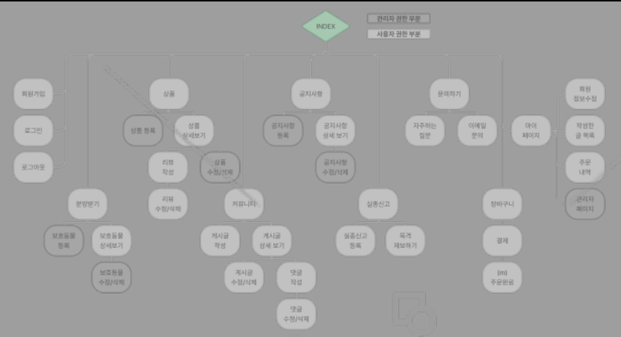
1. Springboot 백엔드 서버에 API 구현
2. React에서 백엔드의 API Url 정의 후
   Axios 라이브러리를 통한 데이터 요청 수행
3. react-router-dom 라이브러리를 통해 프론트 서버 내
   페이지 구성

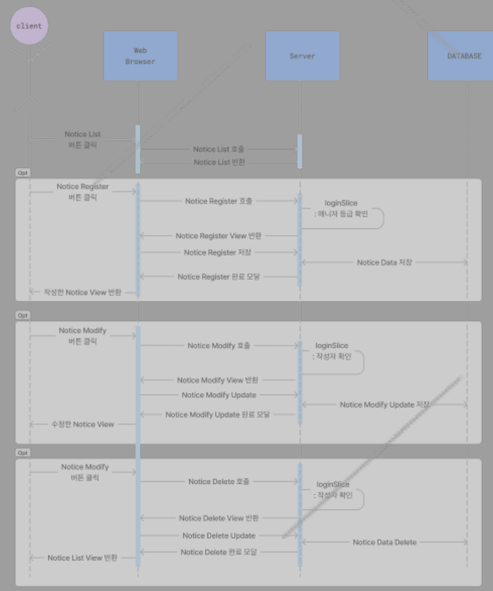
FRONT

```python
const noticeRouter = () => {
  return [
    {
      path: "list",
      element: (
        <Suspense fallback={<LoadingSpinner />}>
          <NoticeList />
        </Suspense>
      ),
    },
    {
      path: "",
      element: <Navigate replace to="/notice/list" />,
    },

    {
      path: "register",
      element: (
        <Suspense fallback={<LoadingSpinner />}>
          <RegNotice />
        </Suspense>
      ),
    },
    {
      path: "read/:noticeBno",
      element: (
        <Suspense fallback={<LoadingSpinner />}>
          <ReadNotice />
        </Suspense>
      ),
    },
    {
      path: "modify/:noticeBno",
      element: (
        <Suspense fallback={<LoadingSpinner />}>
          <ModNotice />
        </Suspense>
      ),
```

SOLUTION

# 06 마주한 어려움




## ☑ 구조 설계

웹 애플리케이션에 추가된 기능이 많아진 만큼
더 **체계적인 구조화**와 **효율적인 상태관리**를 위해
**Redux Toolkit을 사용**하고
**RESTful API 방식**으로 제작하였습니다.

**customHook, reducer, slice를 이용**하여
구조를 더 디테일하게 설계할 수 있었으며,
**권한 별 게시판 등록 기능**을 구분하고
**UML 다이어그램**을 통해
혼동을 극복하였습니다.

SOLUTION

# 06 마주한 어려움

```
const communityRouter = () => {
  return [
    {
      path: "list",
      element: (
        <Suspense fallback={<LoadingSpinner />}>
          <CommunityList />
        </Suspense>
      ),
    },
    {
      path: "",
      element: <Navigate replace to="/community/list" />,
    },

    {
      path: "register",
      element: (
        <Suspense fallback={<LoadingSpinner />}>
          <RegCommunity />
        </Suspense>
      ),
    },
```

```
import axios from "axios";
import { API_SERVER_HOST } from "./rootApi";
import jwtAxios from "../utils/jwtUtil";

const host = `${API_SERVER_HOST}/community/reply`;

//******** Reply  *************/

// 댓글 리스트
export const listReply = async (communityBno) => {
  const res = await axios.get(`${host}/list/${communityBno}`);
  return res.data;
};

// 댓글 등록
export const regReply = async (reply) => {
  const header = { headers: { "Content-Type": "application/json" } };
  const res = await jwtAxios.post(`${host}/register`, reply, header);
  return res.data;
};
```

## ☑ React와 Springboot 연결

프론트엔드와 백엔드를
각각 **React와 Springboot를 사용**하여
프로젝트를 제작했습니다.

**API와 API엔드포인트의 연결**이
undefined 되는 케이스가 수차례 발생하여
**UML 다이어그램 제작**을 통해
사용된 클래스와 객체들을 자세히 관찰하였고,
관련 문제를 해결할 수 있었습니다.

# 07

# 느낀점 / 아쉬운점

## 느낀점

### RESTful API 형식 경험

React-Springboot 연결을 위해
API를 설계하고 API 엔드포인를
설정한 경험을 축적하며
프로세스 이해도를 높였습니다.

또한 오류를 마주했을 때,
해결시간이 단축되는 효과를 체감,
개인적인 만족감을 느꼈습니다.

## 아쉬운점

### 관리자 메일 생성

퍼스널 문의사항과 유기동물 후원에
1:1 이메일 문의 기능을 제공하지만
지속 사용 가능한 도메인 구매에
어려움이 있었습니다.
사이트 내 관리자 메일이 아닌
개인 메일주소를 사용함으로써,
사용자에게 공개되는 부분은
아니지만 개인정보 보호 관련으로
다소 아쉬웠습니다.

Thank you!

# 감사합니다.

상상, 그 이상의 개발자 천하늘

https://github.com/soundspeedskye