

KG아이티뱅크 자바반

PET TAMING

팀원: 이한얼, 김영래



목차

1. 개요

2. 개발 환경

3. DB ERD 와 JPA

4. 개발 일정

5. 팀원 역할

6. 패키지 구성도

7. SECURITY 구성도

8. CRUD 구성도

9. 환경 설정 구성

10. PAGE LAYOUT

11. 영상 시연

12. 코드 리뷰

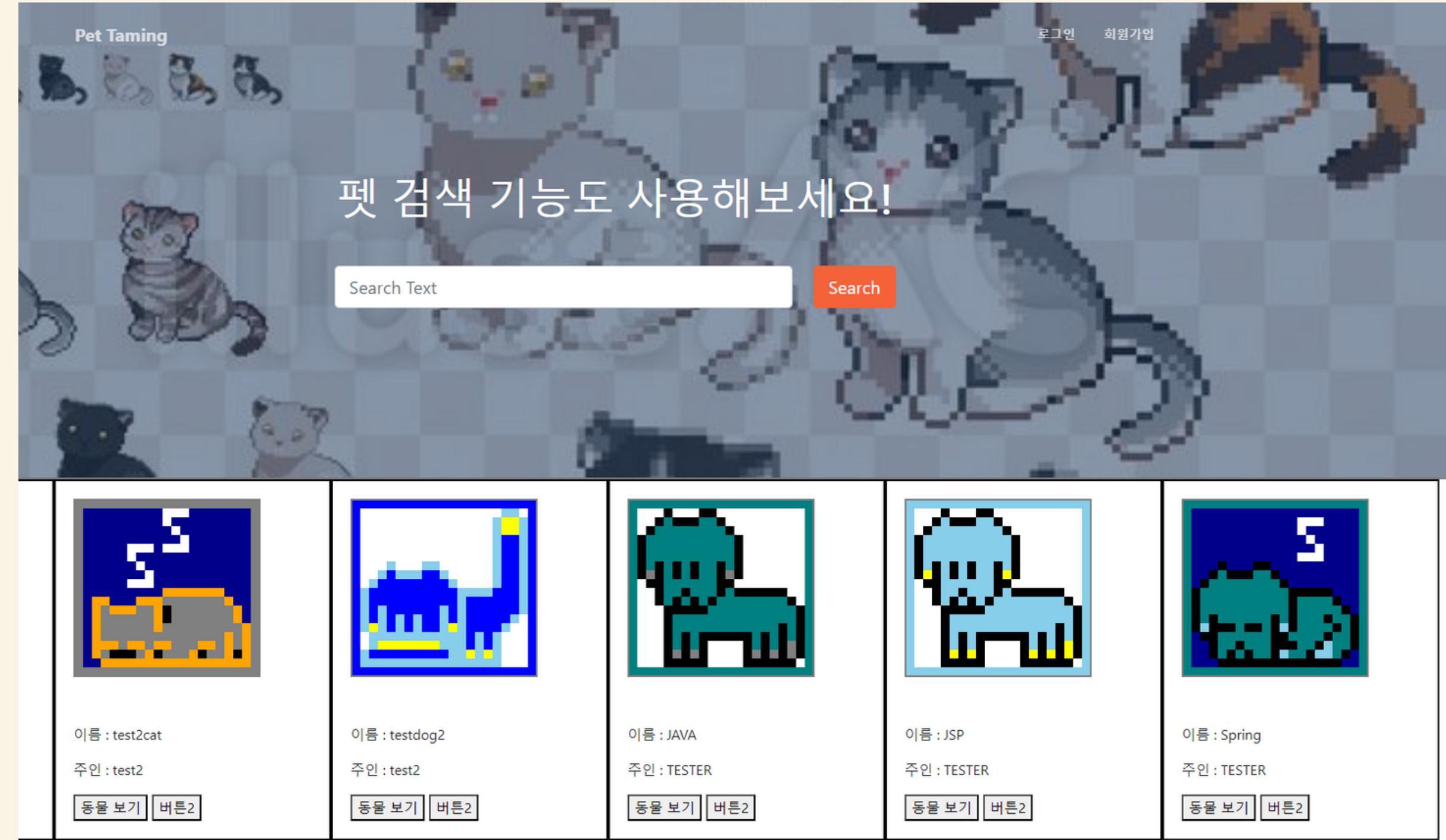
제작 의도

1. 기능 의도

CRUD 기능이 적용된 E-commers는 매우 훌륭한 프로젝트이지만 이러한 기능들을 좀 더 창의적으로 이용해볼 수 없을까 하는 고민중 시도하게 되었습니다.

2. 기획 의도

웹에서 실시간으로 동물을 키우는 사이트는 흔치 않다는 데에서 영감을 얻었으며 특히나 동물키우기 게임들이 오픈되어 보여지는 사이트가 없으니 만들어보기 좋다는 생각에 기획하게 되었습니다.

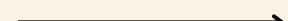


MAIN PAGE



개발 환경

1. 운영 체제: Windows 10
2. Tool : Spring Tools Suite 4.17.2
3. Web Server : Apache Tomcat 9.0
4. JDK : Open JDK 15 version
5. BootStrap : 5.3.1 version
6. 라이브 러리: ORM(JPA,Hibernate)
6. Jquery : 3.6.1 version
7. Database : Mysql Workbench 8.0 CE
8. 개발 도구: Spring Boot
9. View template: Thymeleaf
10. Bulid Tool: Apache Maven 3.6.1
12. 버전 관리 시스템(vcs): Git, Github



PET TAMING

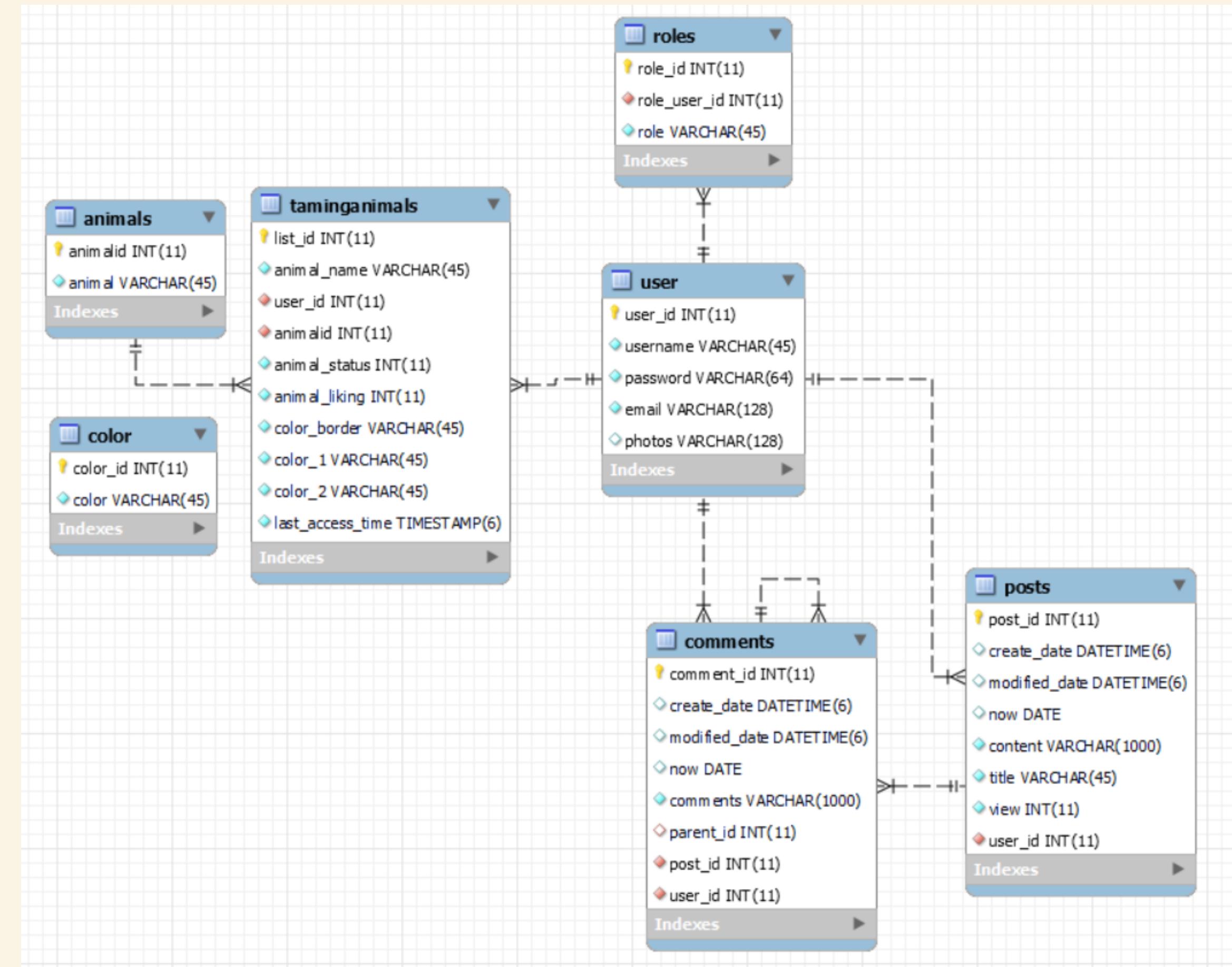
DB ERD 구성도

주요 테이블 = User, TamingAnimals

권한 및 Role 담당 Security 테이블 =
Role 테이블에 JoinColumn을 이용하여
User 테이블의 User_id와 연관 관계 매팅

웹의 서비스 담당 테이블=
Posts 테이블과 Comment 테이블은
OneToMany AND ManyToOne JPA
연관 관계 매팅

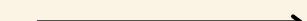
TamingAnimals 테이블은
OneToOne JPA 연관 관계 매팅



JPA 구성도

```
1 @Entity  
2 @Table(name="roles")  
3 public class Roles {  
  
4     @Id  
5     @GeneratedValue(strategy = GenerationType.IDENTITY)  
6     private Integer role_id;  
  
7     @Column(nullable = false)  
8     @JoinColumn(name = "user_id")  
9     private Integer role_user_id;  
  
10    @Column(length = 45, nullable = false)  
11    private String role;  
  
12    public Roles() {  
13    }
```

```
1 import java.sql.Timestamp;  
2  
3 @Entity  
4 @Table(name="taminganimals")  
5 public class TamingAnimals {  
6     @Id  
7     @GeneratedValue(strategy = GenerationType.IDENTITY)  
8     private Integer list_id;  
9  
10    @Column(length = 45, nullable = false)  
11    private String animal_name;  
12  
13    @OneToOne  
14    @JoinColumn(name = "user_id")  
15    private User user_id;  
16  
17    @OneToOne  
18    @JoinColumn(name = "animalid")  
19    private Animals animalid;
```



JPA 구성도

```

// 자유게시판 및 Q&A 게시판 Entity
@Table(name = "posts")
@Entity
public class Posts extends BaseTime{

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer post_id;

    @Column(length = 45, nullable = false)
    private String title;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id", nullable = false)
    private User user_id;

    // 게시글 내용
    @Column(length = 1000, nullable = false)
    private String content;

    @OneToMany(mappedBy = "posts", fetch = FetchType.LAZY,
               cascade = CascadeType.REMOVE)
    @OrderBy("comment_id DESC")
    private List<Comment> comments;

    @Column(columnDefinition = "integer default 0", nullable = false)
    private Integer view;

    public Posts() {
    }
}

```

```

// 댓글 Entity
@Entity
@Table(name = "Comments")
public class Comment extends BaseTime{

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer comment_id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "post_id", nullable = false)
    private Posts posts;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id", nullable = false)
    private User user;

    @Column(length = 1000, nullable = false)
    private String comments;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "parent_id")
    private Comment parent;

    @OneToMany(mappedBy = "parent", orphanRemoval = true)
    private List<Comment> childrens = new ArrayList<>();
}

```

PET TAMING

PET ACTION DB 구성도

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree shows the 'animal_action_db' schema selected. Under 'Tables', the 'cat_1' table is expanded, showing its columns: Y, x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15, x16, x17, x18, x19, and x20. A SQL query window at the top displays the command: '1 • SELECT * FROM animal_action_db.cat_1;'. The main area shows the 'Result Grid' with 20 rows of data. The columns contain various values such as 'color_2', 'white', and 'Pink'. The bottom left pane shows the table structure: 'Table: cat_1' and 'Columns: Y int(11) AI PK, x0 varchar(30), x1 varchar(30)'. The bottom right pane shows the column definitions: 'Y int(11) AI PK', 'x0 varchar(30)', and 'x1 varchar(30)'.

Y	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16	x17	x18	x19	x20
1	color_2	color_2	colo...	color_2	color_2	co...	color_2	col...	color_2	color_2	color_2										
2	color_2	white	colo...	color_bo...	white	w...	white	col...	color_b...	color_border	color_border	white	white	white	color_border	color_border	color_border	white	white	white	color_2
3	color_2	white	colo...	color_2	color_...	co...	color_b...	col...	color_2	color_border	white	white	white	white	color_border	color_1	color_border	white	white	white	color_2
4	color_2	white	colo...	color_2	color_2	co...	color_2	col...	color_2	color_border	white	white	white	white	color_border	color_1	color_border	white	white	white	color_2
5	color_2	white	colo...	color_2	color_2	co...	color_2	col...	color_2	color_border	white	white	white	white	color_border	color_2	color_border	white	white	white	color_2
6	color_2	white	colo...	color_2	color_...	co...	color_b...	col...	color_2	color_border	white	white	white	white	color_border	color_2	color_border	white	white	white	color_2
7	color_2	white	colo...	color_2	color_...	co...	color_b...	col...	color_2	color_border	white	white	white	white	color_border	color_2	color_border	white	white	white	color_2
8	color_2	white	colo...	color_2	color_...	co...	color_b...	col...	color_2	color_border	white	white	white	white	color_border	color_2	color_border	white	white	white	color_2
9	color_2	white	colo...	color_2	color_1	co...	color_1	col...	color_2	color_border	color_border	color_border	color_border	color_border	color_2	color_border	color_2	color_border	white	white	color_2
10	color_2	white	colo...	color_1	Pink	Pink	Pink	col...	color_1	color_2	color_2	white	color_2								
11	color_2	white	white	color_bo...	color_1	co...	color_1	col...	color_1	color_2	color_1	color_2	white	white	color_2						
12	color_2	white	white	color_bo...	color_1	co...	color_1	col...	color_2	color_2	color_1	color_2	white	white	color_2						
13	color_2	white	white	color_bo...	color_1	co...	color_1	col...	color_2	color_2	color_1	color_2	white	white	color_2						
14	color_2	white	white	color_bo...	color_1	co...	color_1	col...	color_2	white	white	color_2									
15	color_2	white	white	color_bo...	color_2	co...	color_1	col...	color_2	white	white	color_2									
16	color_2	white	white	color_bo...	color_2	co...	color_b...	col...	color_2	color_border	color_border	color_border	color_border	color_border	color_2	color_2	color_2	color_2	white	white	color_2
17	color_2	white	white	color_bo...	color_2	co...	white	col...	color_2	color_border	white	white	color_border	color_2	color_2	color_2	color_2	color_2	white	white	color_2
18	color_2	white	white	color_bo...	color_1	co...	white	col...	color_1	color_border	white	white	color_border	color_1	color_border	color_1	color_border	white	white	white	color_2
19	color_2	white	white	color_bo...	color_...	co...	white	col...	color_border	color_border	white	white	color_border	color_border	color_border	color_border	color_border	white	white	white	color_2
20	color_2	color_2	color_2	colo...	color_2	color_2	co...	color_2	col...	color_2	color_2	color_2	color_2								

Pet의 움직임을 담당하는 animal_action_db는 JPA가 아닌 SQL을 사용하여 테이블에 데이터를 입력해주었습니다.

개발 일정

PET TAMING

개발 기간:

23.03.02 ~ 23.03.29(총 28일)

데이터 베이스 구축: 7일

1차 기능 적용 및 점검: 7일

2차 기능 적용 및 점검: 5일

최종 기능 적용 및 점검: 2일

PPT 작성 및 이슈 체크: 3일

웹 디자인: 5일

웹 디자인

17.2%

DB 구축

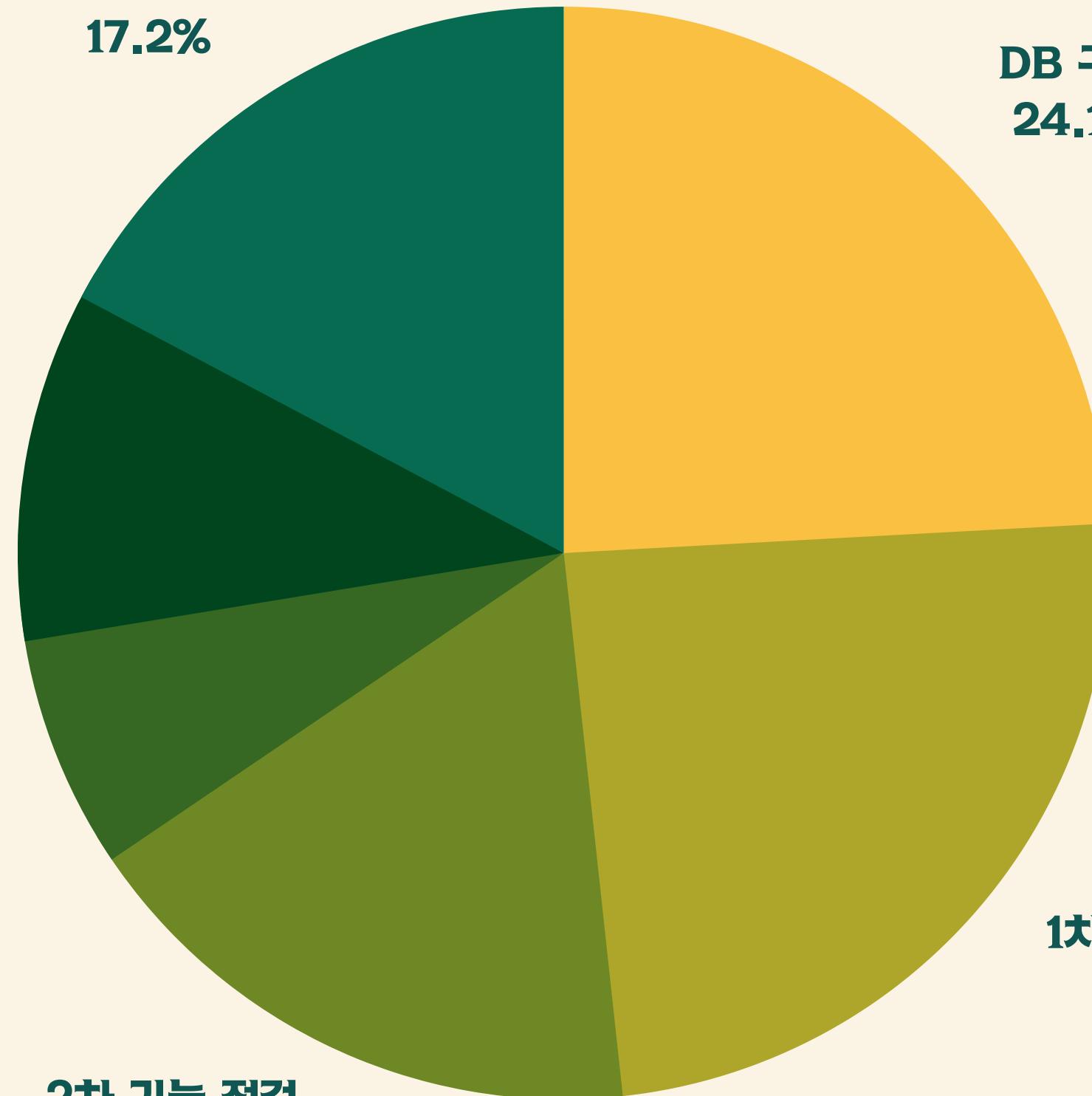
24.1%

1차 기능 점검

24.1%

2차 기능 점검

17.2%



팀원 역할 및 담당 파트

이한얼(User)

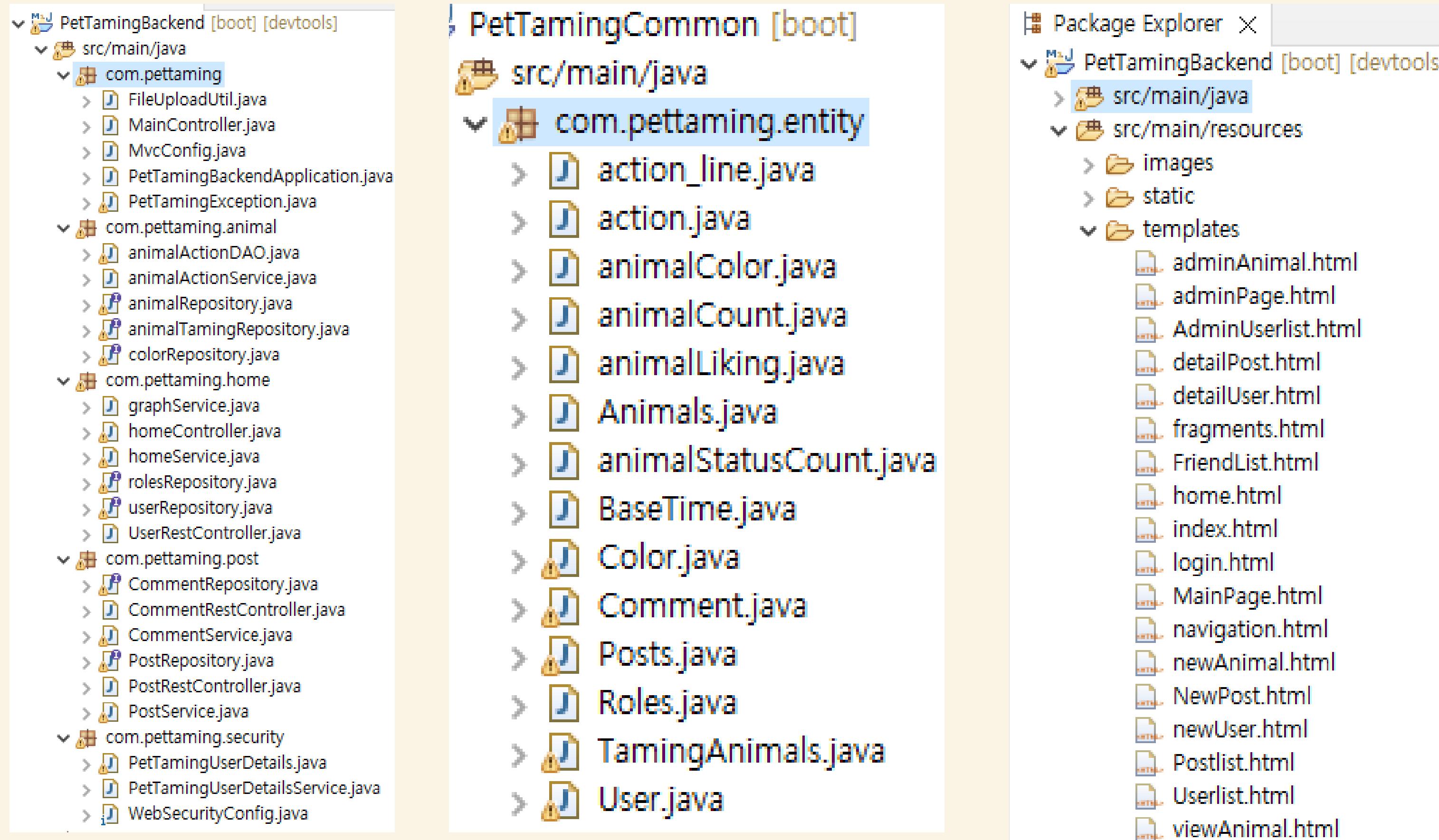
- 회원 가입 및 로그인(100%)
- 회원 정보 수정 및 회원 탈퇴(100%)
- 유저리스트 출력(100%)
- 유저 마이 페이지(80%)
- 파일 업로드(100%)
- SECURITY 설정(50%)
- 게시글 및 댓글 작성(90%)
- 게시글 상세보기(100%)
- 게시글 수정 및 삭제(100%)
- PAGING 및 SORTING 처리(100%)
- 전반적인 웹 디자인(80%)
- 패스워드 암호화(100%)
- 이메일 중복 체크(100%)
- 검색 기능(100%)
- 카카오 로그인(100%)

김영래(Pet)

- 메인 페이지 리스트 출력(100%)
- 펫 만들기(100%)
- 펫 상세 보기(100%)
- 펫의 액션 설정(100%)
- 펫 돌보기 기능(100%)
- 펫 호감도 기능(100%)
- 친구 추가 및 삭제(100%)
- 관리자 페이지(60%)
- 친구 리스트 출력(100%)
- SECURITY 설정(50%)
- 게시글 댓글 작성(10%)

→

프로젝트 패키지 구성도



SECURITY 구성도

- Security 로그인을 위하여 LoginPage 매핑값을 입력 후 로그인 완료 시 MainPage로 이동 설정
- 권한에 영향을 받지 않아야 하는 부분들은 ignoring 처리
- 권한에 따른 Url 매핑 권한 설정
- MainPage 및 Home과 회원가입은 모든 유저가 접속 가능하게 하기위해 permitAll 설정

```

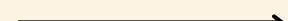
@Override
protected void configure(HttpSecurity http) throws Exception{
    http.authorizeRequests()
        .antMatchers("/MainPage/Admin/**").hasAuthority("ROLE_ADMIN")
        .antMatchers("/MainPage/admin/**").hasAuthority("ROLE_ADMIN")
        .antMatchers("/MainPage/User/new/**").permitAll()
        .antMatchers("/MainPage/Main/**").permitAll()
        .antMatchers("/MainPage/home/**").permitAll()
        .anyRequest().authenticated()
        .and()
        .formLogin()
            .loginPage("/login")
            .defaultSuccessUrl("/MainPage/Main")
            .usernameParameter("email")
            .permitAll();
    http.logout().permitAll();
}

@Override
public void configure(WebSecurity web) throws Exception {
    web.ignoring().antMatchers("/", "/css/**", "/images/**", "/assets/**", "/webjars/**")
        .requestMatchers(PathRequest.toStaticResources().atCommonLocations());
}

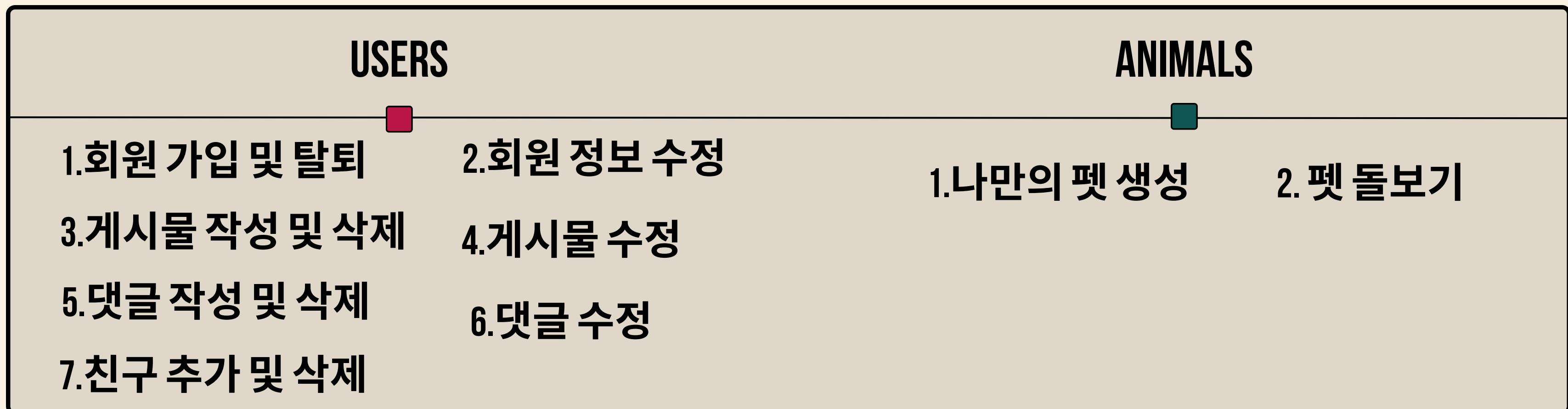
```

NAVIGATION 구성도

```
<nav class="navbar navbar-expand-lg navbar-light fixed-top py-3" id="mainNav" th:fragment="menu">
    <div class="container px-4 px-lg-5">
        <a class="navbar-brand" th:href="@{/MainPage/home}">Pet Taming</a>
        <button class="navbar-toggler navbar-toggler-right" type="button" data-bs-toggle="collapse" data-bs-target="#navbarResponsive"
            aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarResponsive">
            <ul class="navbar-nav ms-auto my-2 my-lg-0">
                <li sec:authorize="isAnonymous()" class="nav-item"><a class="nav-link" th:href="@{/login}">로그인</a></li>
                <li sec:authorize="isAnonymous()" class="nav-item"><a class="nav-link" th:href="@{/MainPage/User/new}">회원가입</a></li>
                <li sec:authorize="isAuthenticated()" class = "nav-item"><a class = "nav-link" th:href="@{/MainPage/Animal/new}">펫 만들기</a></li>
                <li sec:authorize="isAuthenticated()" class="nav-item"><a class="nav-link" th:href="@{/MainPage/Post}">자유 게시판</a></li>
                <li sec:authorize="isAuthenticated()" class="nav-item"><a class="nav-link" th:href="@{/MainPage/User/Mypage}">마이 페이지</a></li>
                <li sec:authorize="isAuthenticated()" class="nav-item"><a class="nav-link" th:href="@{/MainPage/User}">유저 리스트</a></li>
                <li sec:authorize="hasRole('ROLE_ADMIN')" class="nav-item"><a class="nav-link" th:href="@{/MainPage/admin/Graph}">관리자 페이지</a></li>
                <li sec:authorize="isAuthenticated()" class="nav-item"><a th:href = "@{/MainPage/User/Mypage}" class="nav-link" sec:authentication="principal.name"></a></li>
            <form th:action="@{/logout}" method="post" th:hidden="true" name="logoutForm">
                <input type="submit"/>
            </form>
            <li sec:authorize="isAuthenticated()" class="nav-item" id ="logoutLink"><a class="nav-link" th:href="@{/logout}">로그아웃</a></li>
        </ul>
    </div>
</div>
</nav>
```



CRUD 구성도



환경 설정

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:sch
1 ⊕<project xmlns="http://maven.apache.org/POM/4.0.0"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
4     <modelVersion>4.0.0</modelVersion>
5     <groupId>com.pettaming</groupId>
6     <artifactId>PetTamingProject</artifactId>
7     <version>0.0.1-SNAPSHOT</version>
8     <packaging>pom</packaging>
9     <name>PetTamingProject</name>
10    <description>PetTamingProject</description>
11
12 ⊕<modules>
13   <module>PetTamingCommon</module>
14   <module>PetTamingWebParent</module>
15 </modules>
16 </project>
```

프로젝트 모듈화 진행중 최상위 프로젝트
인 PetTamingProject에 Module 버전
과 groupId로 패키지 네이밍 규칙을 설정

pakaging 기준은 pom.xml로 설정해주
고 modules에 하위에 추가되어있는 프로
젝트들을 입력해주었습니다.

환경 설정

```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
1 <?xml version="1.0" encoding="UTF-8"?>
2 ⊕<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://
4   <modelVersion>4.0.0</modelVersion>
5 ⊕<parent>
6   <groupId>org.springframework.boot</groupId>
7   <artifactId>spring-boot-starter-parent</artifactId>
8   <version>2.4.1</version>
9   <relativePath/> <!-- lookup parent from repository -->
10 </parent>
11 <groupId>com.pettaming</groupId>
12 <artifactId>PetTamingCommon</artifactId>
13 <version>0.0.1-SNAPSHOT</version>
14 <name>PetTamingCommon</name>
15 <description>common library</description>
16 ⊕<properties>
17   <java.version>15</java.version>
18 </properties>
19 ⊕<dependencies>
20 ⊕  <dependency>
21    <groupId>org.springframework.boot</groupId>
22    <artifactId>spring-boot-starter-data-jpa</artifactId>
23  </dependency>
24 </dependencies>
25 </project>
```

PetTamingProject의 pom.xml에 추가
해준 모듈 프로젝트중 하나인
PetTamingCommon Project의
pom.xml 구성도입니다.

동일하게 groupId로 패키지 네이밍 규칙
을 설정해주고 주로 Entity 부분을 컨트롤
하게될 프로젝트이기에 dependency로
JPA를 입력해주 PetTamingCommon
프로젝트에 JPA에 대한 의존성을 주입해
주었습니다.

환경 설정

```

3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4   <modelVersion>4.0.0</modelVersion>
5   <parent>
6     <groupId>org.springframework.boot</groupId>
7     <artifactId>spring-boot-starter-parent</artifactId>
8     <version>2.4.1</version>
9     <relativePath/> 
10    </parent>
11    <groupId>com.pettaming</groupId>
12    <artifactId>PetTamingWebParent</artifactId>
13    <version>0.0.1-SNAPSHOT</version>
14    <packaging>pom</packaging>
15    <name>PetTamingWebParent</name>
16    <description>parent web project</description>
17
18    <properties>
19      <java.version>15</java.version>
20    </properties>
21    <dependencies>
22      <dependency>
23        <groupId>org.springframework.boot</groupId>
24        <artifactId>spring-boot-starter-web</artifactId>
25      </dependency>
26
27      <dependency>
28        <groupId>org.springframework.boot</groupId>
29        <artifactId>spring-boot-starter-security</artifactId>
30      </dependency>
31
32      <dependency>
33        <groupId>org.springframework.boot</groupId>
34        <artifactId>spring-boot-starter-thymeleaf</artifactId>
35      </dependency>
36
37      <dependency>
38        <groupId>org.thymeleaf.extras</groupId>
39        <artifactId>thymeleaf-extras-springsecurity5</artifactId>
40      </dependency>
41
42      <dependency>
43        <groupId>org.springframework.boot</groupId>
44        <artifactId>spring-boot-starter-test</artifactId>
45        <scope>test</scope>
46      </dependency>
47
48      <dependency>
49        <groupId>mysql</groupId>
50        <artifactId>mysql-connector-java</artifactId>
51        <scope>runtime</scope>
52      </dependency>
53
54      <dependency>
55        <groupId>org.springframework.boot</groupId>
56        <artifactId>spring-boot-starter-data-jpa</artifactId>
57      </dependency>

```

→

```

</dependency>

<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>jquery</artifactId>
  <version>3.4.1</version>
</dependency>

<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>webjars-locator-core</artifactId>
</dependency>

<dependency>
  <groupId>org.webjars</groupId>
  <artifactId>bootstrap</artifactId>
  <version>5.2.3</version>
</dependency>

<dependency>
  <groupId>com.pettaming</groupId>
  <artifactId>PetTamingCommon</artifactId>
  <version>0.0.1-SNAPSHOT</version>
</dependency>

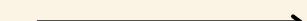
<dependencies>
  <dependency>
    <groupId>
      <plugins>
        <plugin>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
      </plugins>
    </dependency>
    <dependency>
      <module>PetTamingBackend</module>
    </dependency>
  </modules>
</dependencies>

```

환경 설정

앞의 환경 설정 구성도는 PetTamingWebParent 프로젝트의 pom.xml 구성도입니다.

Backend 개발에 쓰이게될 대부분 기능(thymeleaf, security, test,jquery등) 을 dependency로 입력하여 의존성을 주입해주고 Entity를 기반으로 데이터를 요청하게 되기 때문에 PetTamingCommon 프로젝트도 의존성을 주입해주고 프로젝트 관리도구를 Maven으로 입력하기 위해 plugin에 Maven plugin도 추가해주었습니다.

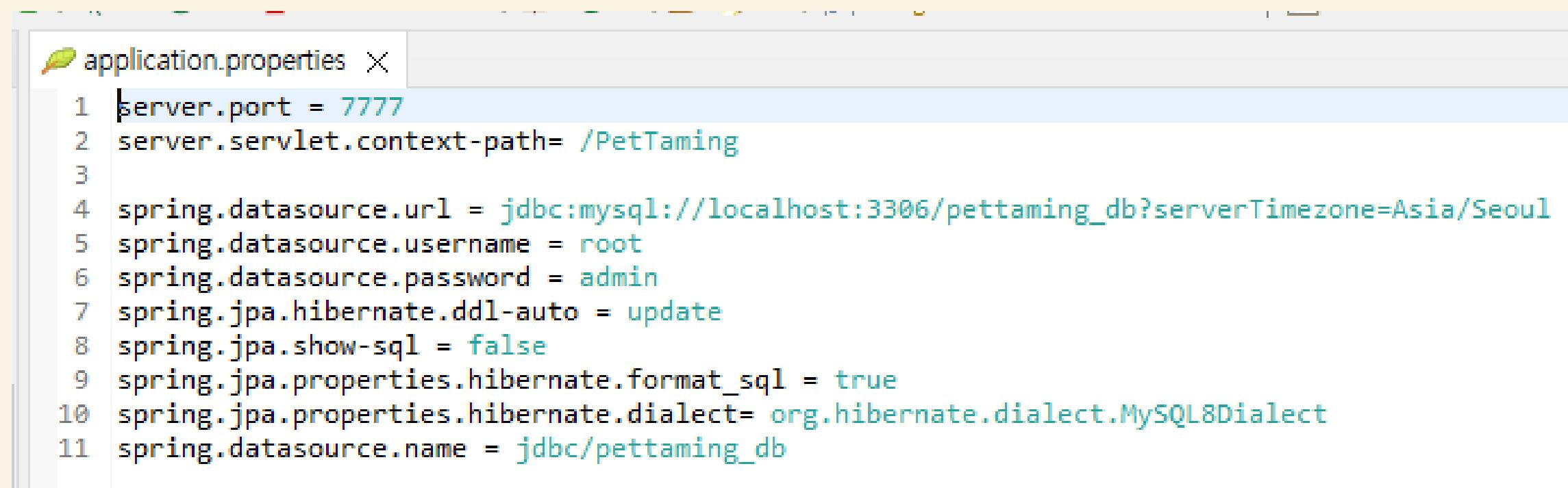


환경 설정

```
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.
4   <modelVersion>4.0.0</modelVersion>
5   <parent>
6     <groupId>com.pettaming</groupId>
7     <artifactId>PetTamingWebParent</artifactId>
8     <version>0.0.1-SNAPSHOT</version>
9     <relativePath/> <!-- lookup parent from repository --&gt;
10    &lt;/parent&gt;
11
12    &lt;artifactId&gt;PetTamingBackend&lt;/artifactId&gt;
13    &lt;name&gt;PetTamingBackend&lt;/name&gt;
14    &lt;description&gt;parent web project&lt;/description&gt;
15
16    &lt;properties&gt;
17      &lt;java.version&gt;15&lt;/java.version&gt;
18    &lt;/properties&gt;
19
20    &lt;dependencies&gt;
21      &lt;dependency&gt;
22        &lt;groupId&gt;com.pettaming&lt;/groupId&gt;
23        &lt;artifactId&gt;PetTamingCommon&lt;/artifactId&gt;
24        &lt;version&gt;0.0.1-SNAPSHOT&lt;/version&gt;
25      &lt;/dependency&gt;
26
27      &lt;dependency&gt;
28        &lt;groupId&gt;org.springframework.boot&lt;/groupId&gt;
29        &lt;artifactId&gt;spring-boot-configuration-processor&lt;/artifactId&gt;
30        &lt;optional&gt;true&lt;/optional&gt;
31      &lt;/dependency&gt;
32
33      &lt;dependency&gt;
34        &lt;groupId&gt;org.springframework.boot&lt;/groupId&gt;
35        &lt;artifactId&gt;spring-boot-devtools&lt;/artifactId&gt;
36      &lt;/dependency&gt;
37    &lt;/dependencies&gt;
38
39    &lt;build&gt;
40      &lt;plugins&gt;
41        &lt;plugin&gt;
42          &lt;groupId&gt;org.springframework.boot&lt;/groupId&gt;
43          &lt;artifactId&gt;spring-boot-maven-plugin&lt;/artifactId&gt;
44        &lt;/plugin&gt;
45      &lt;/plugins&gt;
46    &lt;/build&gt;
47</pre>
```

PetTamingWebParent 하위에 생성된 PetTamingBackend 프로젝트에는 WebParent에 대부분의 기능에 대한 의존성을 주입되어 PetTamingBackend 프로젝트에도 의존성이 주입되어있기에 개발 속도 향상을 위한 devtools 설정 및 어노테이션 기능을 위한 configuration을 dependency로 추가 후 Entity에서 데이터를 주고받기에 PetTamingCommon 프로젝트를 dependency에 추가해주었습니다.

환경 설정

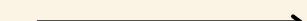


The screenshot shows a code editor window with the file 'application.properties' open. The code defines various application settings:

```
server.port = 7777
server.servlet.context-path= /PetTaming
spring.datasource.url = jdbc:mysql://localhost:3306/pettaming_db?serverTimezone=Asia/Seoul
spring.datasource.username = root
spring.datasource.password = admin
spring.jpa.hibernate.ddl-auto = update
spring.jpa.show-sql = false
spring.jpa.properties.hibernate.format_sql = true
spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MySQL8Dialect
spring.datasource.name = jdbc/pettaming_db
```

application.properties 파일에는 포트번호와 path를 입력하여 기본 url 값을 입력해주고 Mysql 경로 및 계정을 입력하여 Mysql에 있는 DB와 연결해주었습니다.

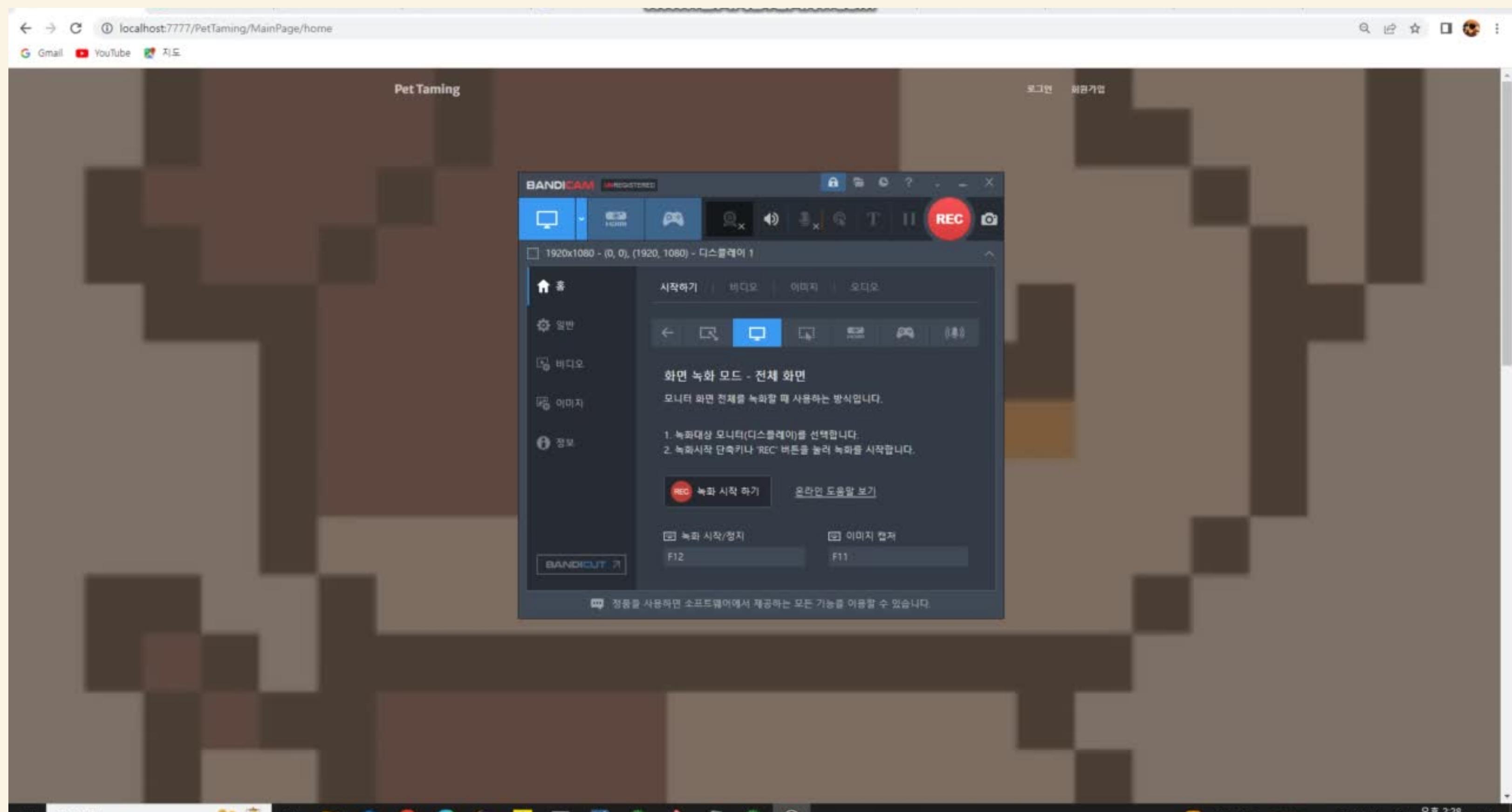
Show sql 부분은 false로하여 프로젝트 실행 시 console에 쿼리문이 보이지 않도록 설정해주었습니다.



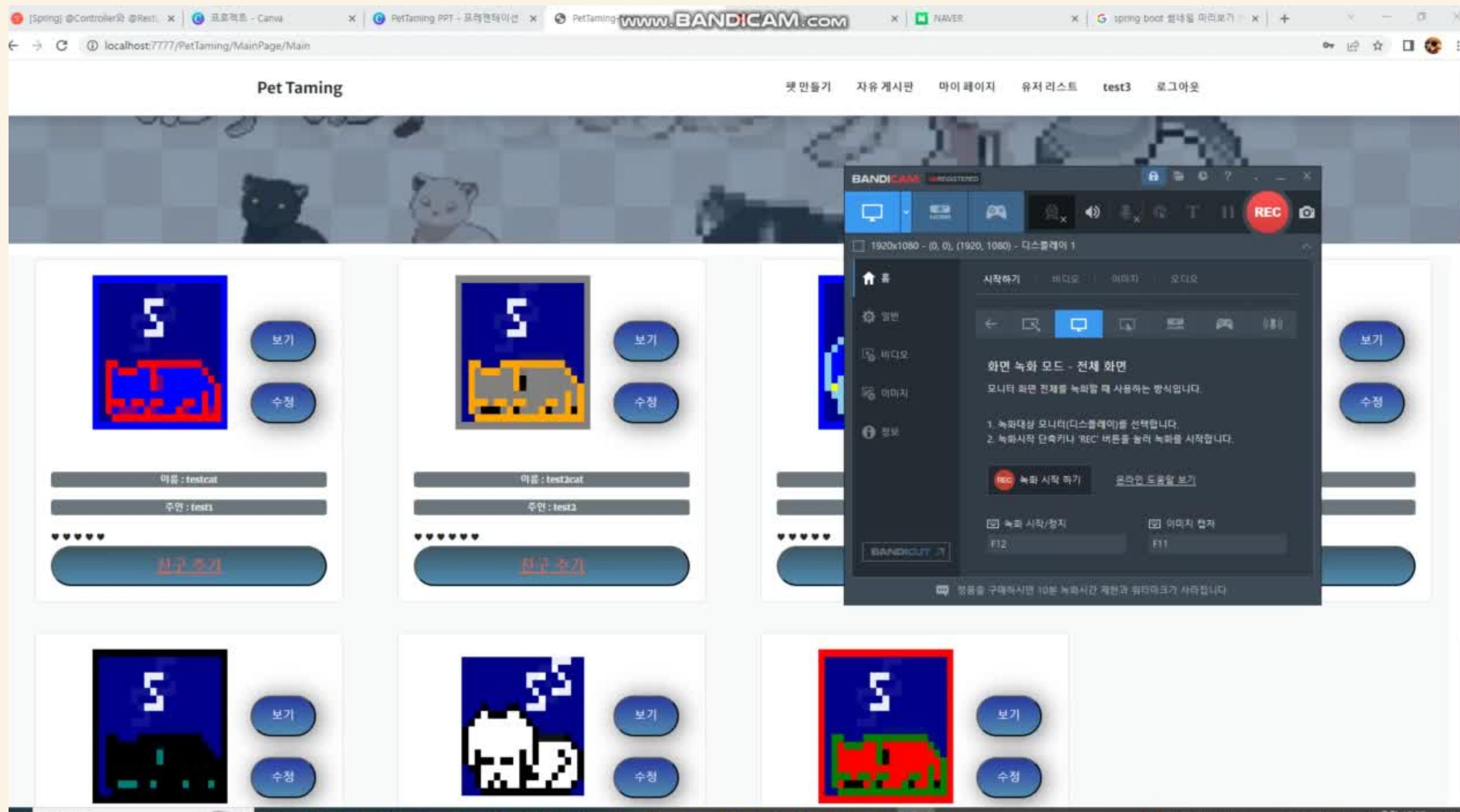
영상 시연

- 1.회원가입 및 로그인
 - 2.유저 마이페이지 및 친구리스트
 - 3.유저 정보 수정
 - 4.나만의 펫 만들기
 - 5.펫 돌보기
 - 6.펫 돌보기 친구 가능
 - 7.게시글 및 댓글 작성
 - 8..관리자 페이지
-

PET TAMING



PET TAMING



PET TAMING

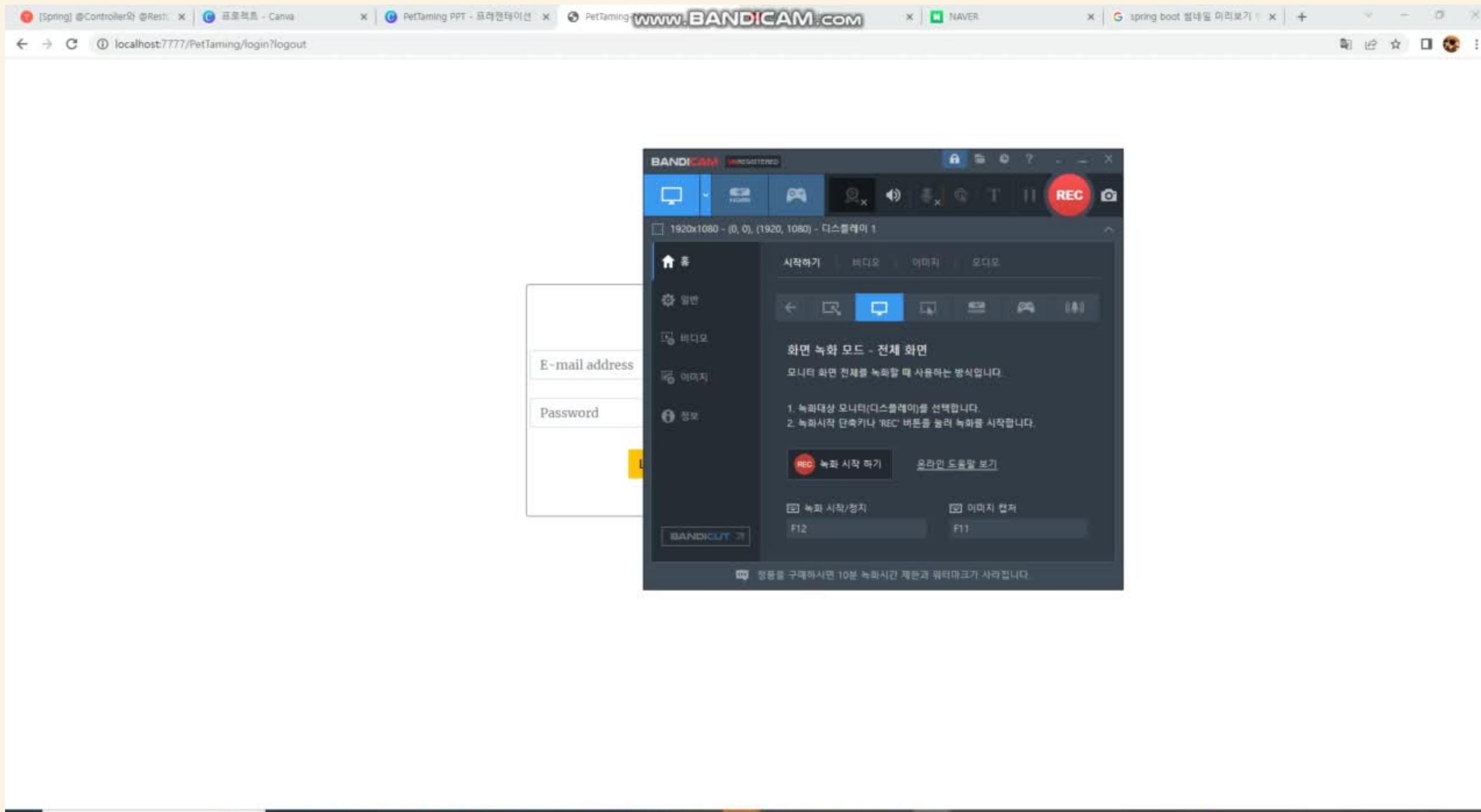
The screenshot shows a web browser window with multiple tabs open. The active tab displays a website titled "Pet Taming". The page features a header with navigation links: "펫 만들기", "자유 게시판", "마이 페이지", "유저 리스트", "관리자 페이지", "test4", and "로그아웃". Below the header is a search bar labeled "Search Text". The main content area contains a grid of four cat illustrations. At the bottom, there is a table listing posts:

No	제목	작성자	등록일	조회수
23	네번째 글	test5	2023-03-30T01:19:24+09:00	9
21	두번째 글입니다.	test3	2023-03-30T01:06:50.592256	13
20	첫글입니다.	test3	2023-03-30T00:57:33.990969	17
22	세번째 글!	test4	2023-03-30T01:12:40.962616	6

At the bottom of the page, there are navigation buttons: "»First", "Previous", "1", "Next", and "Last«". A message below the buttons says "Showing posts # 1 to 4 of 4".

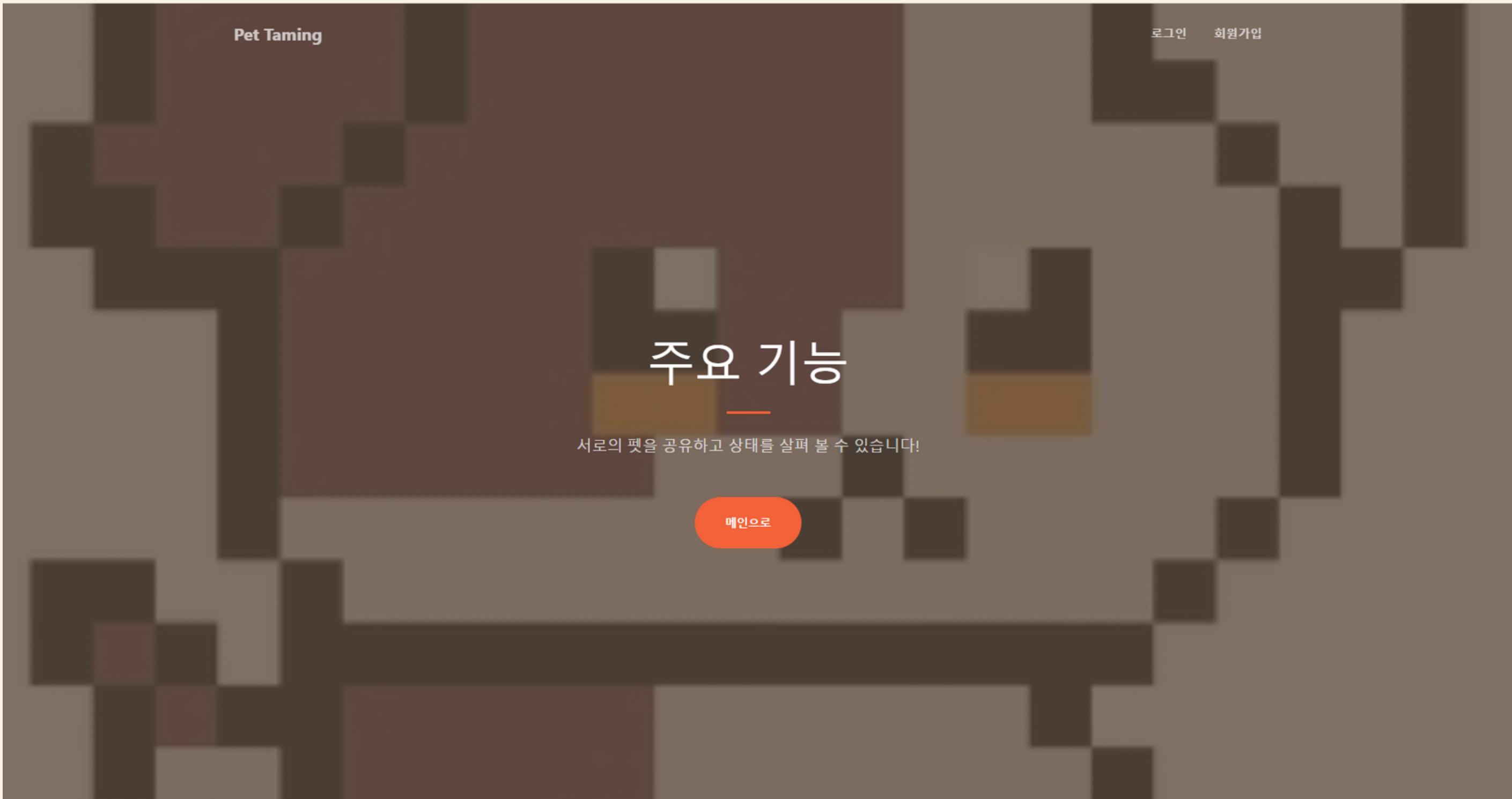
A Bandicam recording overlay is visible, covering the right side of the page. The overlay has a dark theme with white text and icons. It includes a "REC" button, a "Search Text" input field, and various recording options like "화면 녹화 모드 - 전체 화면" and "녹화 시작/정지".

PET TAMING



PET TAMING

PAGE LAYOUT



PAGE LAYOUT

Pet Taming

로그인 회원가입

펫 키우기 주요 서비스



시간별로 변경되는 나만의 펫 상태

유저들의 펫을 같이 보살펴 주어봐요!

도움이 필요한 펫이나 이쁜 펫 등 게시판에 공유해보아요!

친구의 펫을 들봐주거나 친구로 추가된 유저는 나의 펫을 들봐줄 수 있어요!



서로의 동물 공유



자유 게시판



친구 추가된 유저와 함께
돌보기

PET TAMING

PAGE LAYOUT

Pet Taming

로그인 회원가입

기쁨, 슬픔, 배고픔 등 다양한 감정을
펫이 느끼고 있어요!

판에 공유해보아요!

드립니다!

동물 보기

상태

이름 : testcat
주연 : testt

친구 추가

동물 보기

상태

이름 : MyPet
주연 : 오늘의자바

친구 삭제

동물 보기

상태

이름 : 화난고양이
주연 : 오늘의자바

친구 삭제

동물 보기

상태

이름 : 귀여워
주연 : 오늘의자바

친구 삭제

동물 보기

상태

이름 : 날만고양이
주연 : 오늘의자바

친구 삭제

동물 보기

상태

이름 : 한국고양이
주연 : 오늘의자바

친구 삭제

동물 보기

상태

이름 : 복수의 고양이
주연 : 오늘의자바

친구 삭제

동물 보기

상태

이름 : Spring
주연 : 오늘의자바

친구 삭제

동물 보기

상태

이름 : 친환경 강아지
주연 : 오늘의자바

친구 삭제

동물 보기

상태

이름 : 초코나무숲
주연 : 오늘의자바

친구 삭제

동물 보기

상태

이름 : 피카츄
주연 : 오늘의자바

친구 삭제

동물 보기

상태

이름 : 정재
주연 : 오늘의자바

친구 삭제

→

PAGE LAYOUT

The screenshot shows the homepage of a web application titled "Pet Taming". The background features a large, pixelated image of a cat's face and several smaller cat icons. At the top, there is a navigation bar with links for "Pet Taming", "로그인" (Login), and "회원가입" (Sign Up). A search bar with the placeholder "검색 기능도 사용해보세요!" (Use the search function too!) and a "Search" button are also present. Below the search bar, there are four cards, each representing a pet:

- Card 1:** Features a blue and red cat icon. Buttons for "보기" (View) and "상태" (Status) are shown. Below the icon, the name "이름 : testcat" and owner "주인 : test1" are listed. The card has a rating of 5 stars and a "친구 추가" (Add Friend) button at the bottom.
- Card 2:** Features a yellow and black cat icon. Buttons for "보기" (View) and "상태" (Status) are shown. Below the icon, the name "이름 : MyPet" and owner "주인 : 오늘의자바" are listed. The card has a rating of 5 stars and a "친구 추가" (Add Friend) button at the bottom.
- Card 3:** Features a multi-colored cat icon. Buttons for "보기" (View) and "상태" (Status) are shown. Below the icon, the name "이름 : 화난고양이" and owner "주인 : 오늘의자바" are listed. The card has a rating of 5 stars and a "친구 추가" (Add Friend) button at the bottom.
- Card 4:** Features a red and black cat icon. Buttons for "보기" (View) and "상태" (Status) are shown. Below the icon, the name "이름 : 귀여워" and owner "주인 : 오늘의자바" are listed. The card has a rating of 5 stars and a "친구 추가" (Add Friend) button at the bottom.

PET TAMING

펫만들기



펫 이름 : 나만의 펫 만들기 Cancel

동물 선택 cat dog

기본색 : Brown

White Black Red Teal Blue Brown Gray Yellow Green Purple Orange Skyblue

보조색 : White

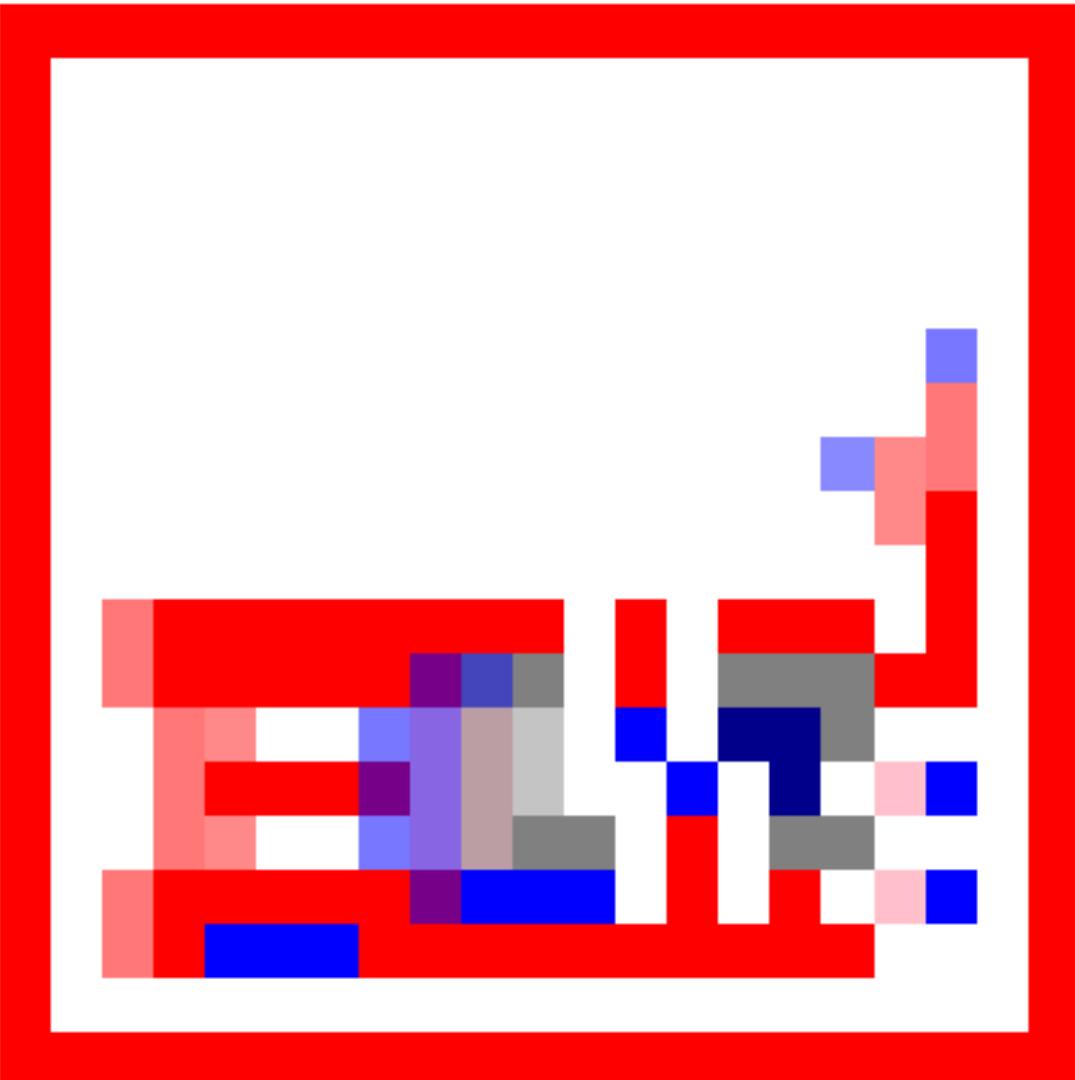
White Black Red Teal Blue Brown Gray Yellow Green Purple Orange Skyblue

태두리색 : Black

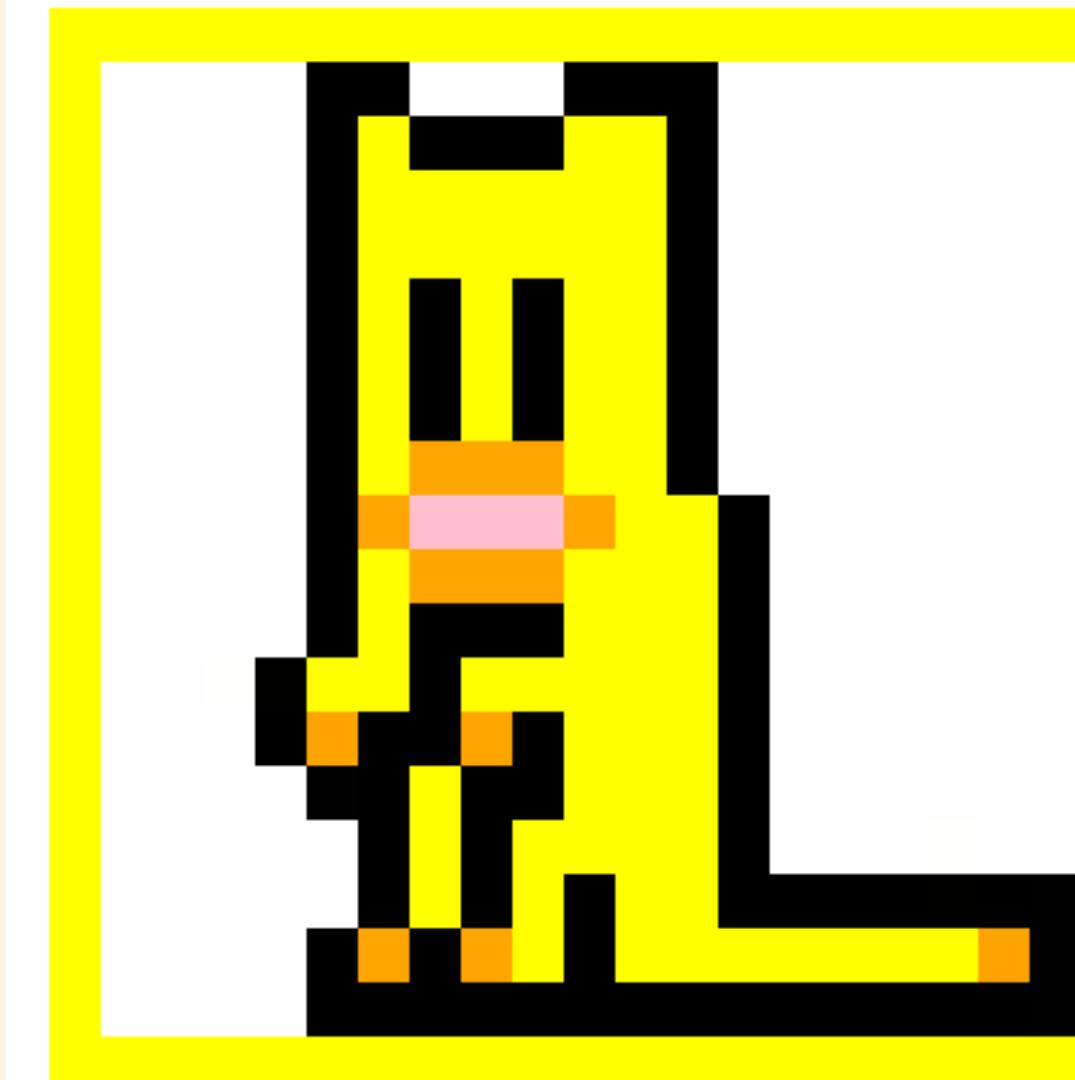
White Black Red Teal Blue Brown Gray Yellow Green Purple Orange Skyblue

PET TAMING

상세보기,돌보기



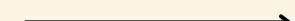
이름 : 한국고양이
주인 : 오늘의자바
♥ ♥ ♥ ♥ ♥
[친구 삭제](#)
[밥먹이기](#)
[놀아주기](#)
[간식주기](#)
[빗겨주기](#)



이름 : 피카츄
주인 : 오늘의자바
♥ ♥ ♥ ♥ ♥
[친구 추가](#)

PAGE LAYOUT

1. Web의 전반적인 틀 및 디자인 : boot Strap
2. 테이블 정렬 및 색상: .css 에 입력하여 적용
3. Pet의 다양한 액션 : Java script로 입력



PET TAMING

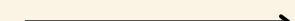
번호	사진	이름	이메일	수정	삭제
3		admin	admin@email.com	UPDATE	DELETE
5		test1	test1@email.com	UPDATE	DELETE
9		test2	test2@email.com	UPDATE	DELETE
10		test3	test3@email.com	UPDATE	DELETE

• [First](#) →

유저 주요 코드

1. Userlist Paging 및 Sorting 그리고 검색

2. User의 전반적인 CRUD 및 마이 페이지



USER SERVICE와 REPOSITORY

Paging, Sorting, Searching

```
    return userRepo.findAll(keyword, pageable);
}

//유저 리스트 페이징, Sorting, 검색 처리 메서드
public Page<User> Pagelist(String keyword,int pageNum,String sortField,String sortDir){

    Sort sort = Sort.by(sortField);

    sort = sortDir.equals("asc") ? sort.ascending() : sort.descending();

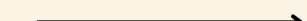
    Pageable pageable = PageRequest.of(pageNum - 1, USER_PER_PAGE, sort);

    System.out.println(sort);

    if(keyword != null) {
        return userRepo.findAll(keyword,pageable);
    }

    return userRepo.findAll(pageable);
}
```

```
@Query("SELECT u FROM User u WHERE CONCAT(u.user_id,' ',u.username,' ',u.email) LIKE %?1%")
public Page<User> findAll(String keyword, Pageable pageable);
```



USER CONTROLLER

PET TAMING

Paging, Sorting, Searching

```
//유저 리스트 Paging, Sorting, 검색 처리
@GetMapping("/{authority}/page/{pageNum}")
public String listByPage(@PathVariable(name = "pageNum") int pageNum
    , @PathVariable("authority") String authority
    , @Param("keyword") String keyword
    , Model theModel
    , Authentication auth
    , @Param("sortField") String sortField,
    @Param("sortDir") String sortDir) {

    String reversesortDir = sortDir.equals("asc") ? "desc" : "asc";

    User user = homeservice.findByUserName(auth.getName());
    theModel.addAttribute("currentPage", pageNum);

    theModel.addAttribute("user", user);
    System.out.println("유저: " + sortField);
    System.out.println("유저: " + sortDir);

    if(authority.equals("Friend")) {
        List<Integer> RoleList = homeservice.Role_FindById(user.getUser_id());
        List<User> listUser = homeservice.FriendPagelist(pageNum,RoleList);
        List<User> page = homeservice.FriendPageAll(pageNum,RoleList);

        theModel.addAttribute("Userlist", listUser);
        theModel.addAttribute("totalPages", (int)(Math.ceil((double)(page.size())/4)))
        theModel.addAttribute("totalItems", page.size());

        long startCount = (pageNum -1) * homeservice.USER_PER_PAGE + 1;
        long endCount = startCount + homeservice.USER_PER_PAGE -1;

        if(endCount > page.size()){
            endCount = page.size();
        }
        theModel.addAttribute("startCount", startCount);
        theModel.addAttribute("endCount", endCount);
    } //페이징 처리된 뉴서 리스트 을
    @GetMapping("/User")
    public String findAll(Model theModel,Authentication auth) {

        return listByPage(1,"User",null,theModel, auth, "username", "asc");
    }
}
```

```
else {
    Page<User> page = homeservice.Pagelist(keyword,pageNum,sortField,sortDir)
    List<User> listUser = page.getContent();

    theModel.addAttribute("Userlist", listUser);
    theModel.addAttribute("totalPages", page.getTotalPages());
    theModel.addAttribute("totalItems", page.getTotalElements());

    long startCount = (pageNum -1) * homeservice.USER_PER_PAGE + 1;
    long endCount = startCount + homeservice.USER_PER_PAGE -1;
    if(endCount > page.getTotalElements()){
        endCount = page.getTotalElements();
    }
    if(endCount > page.getTotalElements()){
        endCount = page.getTotalElements();
    }
    theModel.addAttribute("startCount", startCount);
    theModel.addAttribute("endCount", endCount);
}

theModel.addAttribute("keyword",keyword);
theModel.addAttribute("sortField",sortField);
theModel.addAttribute("sortDir",sortDir);
theModel.addAttribute("reversesortDir",reversesortDir);

if(authority.equals("User"))
    return "Userlist";
else if(authority.equals("Admin"))
    return "AdminUserlist";
else if(authority.equals("Friend"))
    return "FriendList";
return "Userlist";

```



PET TAMING

USER SERVICE

User CRUD

```
//유저를 테이블에 저장하기 위한 메서드
public void save(User user) {

    boolean isUpdatingUser = (user.getUser_id() != null);
    if (isUpdatingUser) {
        User existingUser = userRepo.findById(user.getUser_id()).get();

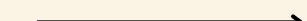
        if (user.getPassword().isEmpty()) {
            user.setPassword(existingUser.getPassword());
        }else {
            encodePassword(user);
        }

        userRepo.save(user);
    }else {
        //패스워드를 암호화하기위한 매서드
        encodePassword(user);
        //유저를 우선 저장 후
        //저장과 동시에 유저의 정보를 기반으로 권한 테이블에 유저 권한을 저장
        userRepo.save(user);
        rolRepo.save(new Roles(userRepo.getUserByName(user.getEmail()).getUser_id(),"ROLE_USER"));
    }

    //패스워드를 암호화하기 위한 매서드
    private void encodePassword(User user) {

        //패스워드 암호화
        String encodedPassword = passwordEncoder.encode(user.getPassword());
        //암호화된 패스워드를 유저에게 저장
        user.setPassword(encodedPassword);
    }
}
```

```
7     //유저리포지토리에서 유저 이름으로 객체를 불러오는 매서드
8     public User findByName(String name) {
9
10         return userRepo.getUserByName(name);
11     }
12     // 유저를 삭제하는 매서드
13     // 유저 삭제 시 유저의 룸과 Role, 게시글, 댓글, 그리고 친구 리스트 모두 지워지도록 설정
14     public void deleteById(Integer id) {
15
16         rolRepo.deleteById(id);
17         commentRepo.deleteByIdCommentUserId(id);
18         rolRepo.deleteByUserId_Friend("ROLE_" + id);
19         aniTamRepo.deleteByIdUserId(id);
20         postRepo.deleteByIdUserId(id);
21         userRepo.deleteById(id);
22     }
23     // 유저를 선택하는 매서드
24     public User findById(Integer id) {
25
26         return userRepo.findById(id).get();
27     }
28
29     public void saveRole(Integer user_id, Integer animalMaster_id) {
30
31         rolRepo.save(new Roles(user_id,"ROLE_" + animalMaster_id));
32     }
33
34     public void deleteRole(Integer user_id, Integer animalMaster_id) {
35
36         rolRepo.delete(user_id,"ROLE_" + animalMaster_id);
37     }
38 }
```



PET TAMING

USER CONTROLLER

User CRUD

```
}

//회원가입 페이지 출력 메서드
@GetMapping("/User/new")
public String newUserPage(Model model) {
    //새 유저를 생성
    User theUser = new User();
    //생성된 유저와 페이지의 이름을 어트리뷰트로 저장
    model.addAttribute("authority", "user");
    model.addAttribute("users", theUser);
    model.addAttribute("pageTitle", "회원 가입");

    return "newUser";
}

//유저 정보 업데이트
@GetMapping("/User/update/{authority}/{id}")
public String updateUser(@PathVariable("id") Integer id
    , @PathVariable("authority") String authority
    , Model theModel, RedirectAttributes RA) throws PetTamingException {
    User user = homeservice.findById(id);

    if(user.getPassword() == null) {
        theModel.addAttribute("check", "new");
    }

    theModel.addAttribute("users", user);
    theModel.addAttribute("getid", id);
    theModel.addAttribute("authority", authority);

    theModel.addAttribute("pageTitle", user.getUsername() + "님 정보 수정 페이지.");

    return "newUser";
}
```

```
//회원가입 페이지에서 저장한 유저 정보를 테이블에 업데이트 하기 위한 메서드
//사진 업로드 시 기존 사진 삭제
@PostMapping("/User/save/{authority}")
public String newUserSave(@ModelAttribute("user")User user
    , @RequestParam("image") MultipartFile multipartFile
    , @PathVariable("authority") String authority
    , Model theModel,
    RedirectAttributes reAt) throws IOException{
    if(!multipartFile.isEmpty()) {
        String filename = StringUtils.cleanPath(multipartFile.getOriginalFilename());
        user.setPhotos(filename);
        homeservice.save(user);
        User saveUser = homeservice.findByUserName(user.getEmail());
        String uploadDir = "User-photos/" + saveUser.getUser_id()///

        FileUploadUtil.cleanDir(uploadDir);
        FileUploadUtil.sendFile(uploadDir, filename, multipartFile);
    }else {
        if(user.getPhotos().isEmpty()) {
            user.setPhotos(null);
        }
        homeservice.save(user);
    }

    if(authority.equals("user"))
        return "redirect:/MainPage/Main";
    else if(authority.equals("admin"))
        return "redirect:/MainPage/admin/User";

    return "redirect:/MainPage/Main";
}

@GetMapping("/User/delete/{authority}/{id}")
public String deleteUser(@PathVariable("id") Integer id
    , @PathVariable("authority") String authority
    , Model theModel) {
    homeservice.deleteById(id);

    if(authority.equals("user"))
        return "redirect:/login?logout";
    else if(authority.equals("admin"))
        return "redirect:/MainPage/admin/User";

    return "redirect:/MainPage/User";
}
```

USER CONTROLLER

User CRUD

```
7 public class PetTamingUserDetails implements UserDetails {  
8  
9  
0@ 1 @Autowired  
2 2 private User user;  
3  
4@ 3 public PetTamingUserDetails(User user) {  
5 4     this.user = user;  
6 }  
7  
8@ 8@Override  
9 9 public Collection<? extends GrantedAuthority> getAuthorities() {  
0 10     Set<Roles> roles = user.getRoles();  
1 11     List<SimpleGrantedAuthority> authorities = new ArrayList<>();  
2 12     for (Roles role : roles) {  
3 13         authorities.add(new SimpleGrantedAuthority(role.getRole()));  
4 14     }  
5 15     return authorities;  
6 }  
7  
2@ 16@Override  
3 17 public String getPassword() {  
4 18     return user.getPassword();  
5 }  
6  
7@ 19@Override  
8 20 public String getUsername() {  
9 21     return user.getEmail();  
0 }  
1  
2@ 22@Override  
3 23 public String getName() {  
4 24     return user.getUsername();  
5 }
```

```
1@ 1@Query("Select u From User u WHERE u.email = :email")  
2 2 User findByEmail(@Param("email") String email);  
3
```

USER CONTROLLER

PET TAMING

User CRUD

```
//Email 중복 체크를 위한 메서드
public boolean isEmailUnique(Integer user_id, String email) {

    User user = userRepo.findByEmail(email);

    if(userRepo.findByEmail(email) == null)
        return true;

    boolean CreateUser = (user_id == null);
    if(CreateUser) {
        if(user != null) {
            return false;
        }
    }else {
        if(user.getUser_id() != user_id) {
            return false;
        }
    }
}
```

```
@RestController
public class UserRestController {

    @Autowired
    private homeService hs;

    @PostMapping("/users/check_email")
    public String checkDuplicateEmail(@Param("user_id") Integer user_id, @Param("email") String email) {
        return hs.isEmailUnique(user_id, email) ? "OK" : "Duplicated";
    }
}
```

PET TAMING

USER CONTROLLER

User CRUD

```
function checkEmailUnique(form){  
    url = "[[@{/users/check_email}]]";  
  
    userid = $("#user_id").val();  
    userEmail = $("#email").val();  
    csrfValue = $("input[name='_csrf']").val();  
    params = {user_id: userid, email : userEmail, _csrf : csrfValue};  
  
    $.post(url, params, function(response){  
        if(response == "OK"){  
            form.submit();  
        }else if(response == "Duplicated"){  
            alert(userEmail + "가 이미 등록된 이메일입니다. 다시 입력해주세요.")  
        }else{  
            alert(userid + userEmail + "서버 연결에 실패하였습니다.")  
        }  
    }).fail(function(){  
        alert("에러2" + "서버 연결에 실패하였습니다.")  
    });  
    return false;  
}  
  
;script>
```

```
</script>  
);  
$(document).ready(function(){  
    $("#buttonCancel").on("click",function(){  
        window.location = "[[@{/MainPage/home}]]";  
    });  
    rawPassworded = "[${users.password}]";  
    if(rawPassworded != null){  
        $("#password").removeAttr("required");  
        console.log(rawPassworded)  
        console.log(rawPassworded != null)  
    }  
  
    $("#fileImage").change(function(){  
        fileSize = this.files[0].size;  
        if(fileSize > 1048576){  
            this.setCustomValidity("1MB를 초과하는 파일은 업로드 할 수 없습니다.");  
            this.reportValidity();  
        }else{  
            this.setCustomValidity("");  
            showImageThumbnail(this);  
        }  
    });  
});  
  
function showImageThumbnail(fileInput){  
    var file = fileInput.files[0];  
    var reader = new FileReader();  
    reader.onload = function(e){  
        $("#thumbnail").attr("src", e.target.result);  
    };  
    reader.readAsDataURL(file);  
}
```

PET TAMING

USER CONTROLLER

User MyPage

```
//Navigation var에 입력될 현재 유저 기반 마이페이지
@GetMapping("/User/Mypage")
public String UserMyPage(Model theModel,
    Authentication auth) throws Exception {

    User user = homeservice.findByUserName(auth.getName());

    List<TamingAnimals> TA = homeservice.tamingAnimal_findByUserId(user.getUser_id());
    animalservice.getAllAnimalAction(TA);

    theModel.addAttribute("MyAccount", user);
    theModel.addAttribute("Users", user);

    theModel.addAttribute("allAnimals", TA);
    theModel.addAttribute("pageTitle", "마이페이지 입니다.");

    return "detailUser";
}
//유저 리스트에 출력될 마이페이지
@GetMapping("/User/Mypage/{User}")
public String UserMyPageViwe (Model theModel
    ,Authentication auth
    ,@PathVariable(name="User") Integer User_id) throws Exception {

    User user = homeservice.findById(User_id);

    List<TamingAnimals> TA = homeservice.tamingAnimal_findByUserId(User_id);
    animalservice.getAllAnimalAction(TA);

    theModel.addAttribute("MyAccount", homeservice.findByUserName(auth.getName()));

    theModel.addAttribute("Users", user);

    theModel.addAttribute("allAnimals", TA);
    theModel.addAttribute("pageTitle", "마이페이지 입니다.");

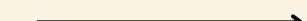
    return "detailUser";
}
```

```
<
class="container mt-5" id = "umyPage">
<div class="row">
<div class="col-lg-8">
<!-- Post content-->
<article>
<!-- Post header-->
<header class="mb-4">
<!-- Post title-->
<h1 class="fw-bolder mb-1">[[ ${Users.username}]]님의 [[ ${pageTitle}]]</h1>
<!-- Post meta content-->
<div class="text-muted fst-italic mb-2"><b>E-mail: [[ ${Users.email}]]</b></div>
</header>
<!-- Preview image figure-->
<figure class="mb-4">
</figure>
<button><a th:href="@{'/MainPage/Friend/page/1' + '?sortField=username&sortDir=' + ${sortDir}}" title = "Friend List">Fri
<button th:if="${Users.user_id == MyAccount.user_id}"><a th:href="@{'/MainPage/User/update/user/' + ${Users.user_id}}" tit
UPDATE</a></button>

<button th:if="${Users.user_id == MyAccount.user_id}"><a class = "fa-solid fa-trash fa-2x icon-silver link-delete" title =
th:href="@{'/MainPage/User/delete/user/' + ${Users.user_id}}" th:userId = "${Users.user_id}">DELETE</a></button>
</div>
</div>
<div style="float:right;" id = "mypagePet">
<div class="card AllAmimalTd" th:each="Animals : ${allAnimals}" style="width: 400px; padding: 20px;">
<div style="position:relative; height: 250px; width: 250px; margin:auto;" th:onclick="|location.href='/PetTaming/MainPage/ViweAnimal/${Animals.list_id}'|">
<div style="position:absolute; z-index: 1;" id = "moving">
<table>
<tr th:each="action : ${Animals.animalAction.x_axis_line}">
<div th:each="actionLine : ${action.x_axis}" th:switch="${actionLine}">
<td th:case="color_1"
th:class="${Animals.color_1}+_box"
style="height: 10px; width: 10px; "></td>
<td th:case="color_2"
th:class="${Animals.color_2}+_box"
style="height: 10px; width: 10px; "></td>
<td th:case="color_border"
th:class="${Animals.color_border}+_box"
style="height: 10px; width: 10px; "></td>

```

시간에 따른 펫의 행동 변화 메커니즘



PET의 주요 코드

```
//시간 지남에 따라 모든 동물의 상태를 업데이트 하는 메서드  
public List<TamingAnimals> animal_update_findAll() {  
  
    List<TamingAnimals> allAnimals = animal_findAll();  
  
    for(TamingAnimals animal : allAnimals) {  
        animal_Status_Update(animal);  
    }  
  
    return allAnimals;  
}
```

```
//시간 지남에 따라 동물의 상태를 업데이트 하는 메서드  
private void animal_Status_Update(TamingAnimals animal) {  
  
    Random random = new Random();  
  
    Timestamp lastTime = animal.getLast_Access_time();  
    Timestamp nowTime = new Timestamp(System.currentTimeMillis());  
  
    int lastMinute = lastTime.getMinutes();  
    int nowMinute = nowTime.getMinutes();  
  
    int lastHour = lastTime.getHours();  
    int nowHour = nowTime.getHours();  
  
    int lastDay = lastTime.getDay();  
    int nowDay = nowTime.getDay();  
  
    int lastMonth = lastTime.getMonth();  
    int nowMonth = nowTime.getMonth();  
  
    int lastYear = lastTime.getYear();  
    int nowYear = nowTime.getYear();
```

PET의 주요 코드

```
//수면시간이 되면 잠자기 액션(6)으로 고정됩니다.
if( nowHour < 7 ){
    animal.setAnimal_status(6);
    animal.setLast_Access_time(nowTime);

}
//수면시간이 끝날경우 일반 액션(1)으로 돌아옵니다.
else if(animal.getAnimal_status() == 6
    && nowHour >= 7) {

    animal.setAnimal_status(1);
    animal.setLast_Access_time(nowTime);
}
```

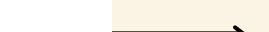
```
//식사시간이 되면 배고픔 액션(4)이 됩니다.
else if((lastHour < 8 && 8 < nowHour) ||
(lastHour == 8 && 8 == nowHour &&
lastDay != nowDay ||
lastMonth != nowMonth ||
lastYear != nowYear))

|| (lastHour < 12 && 12 < nowHour)
|| (lastHour == 12 && 12 == nowHour &&
lastDay != nowDay ||
lastMonth != nowMonth ||
lastYear != nowYear)

|| (lastHour < 19 && 19 == nowHour)
|| (lastHour == 19 && 19 == nowHour &&
lastDay != nowDay ||
lastMonth != nowMonth ||
lastYear != nowYear)) {

    animal.setAnimal_status(4);
    animal.setLast_Access_time(nowTime);

}
```



PET의 주요 코드

```
//식사시간이 되면 배고픔 액션(4)이 됩니다.
else if((lastHour < 8 && 8 < nowHour) ||
(lastHour == 8 && 8 == nowHour &&
(lastDay != nowDay ||
lastMonth != nowMonth ||
lastYear != nowYear ))

|| (lastHour < 12 && 12 < nowHour)
|| (lastHour == 12 && 12 == nowHour &&
(lastDay != nowDay ||
lastMonth != nowMonth ||
lastYear != nowYear ))

|| (lastHour < 19 && 19 == nowHour)
|| (lastHour == 19 && 19 == nowHour &&
(lastDay != nowDay ||
lastMonth != nowMonth ||
lastYear != nowYear)) {

animal.setAnimal_status(4);
animal.setLast_Access_time(nowTime);

}
```

```
//배고픔 액션이 시작되면 1시간동안 액션은 자동으로 해제되지 않습니다.
else if( animal.getAnimal_status() == 4
&& ((lastHour == nowHour) || ((60 - lastMinute) + nowMinute <= 60) &&
(lastHour < nowHour))) {}

//1시간동안 배고픔 액션(4)이 지속될 경우 슬픔 액션(3)으로 전환됩니다.
else if( animal.getAnimal_status() == 4
&& (((60 - lastMinute) + nowMinute > 60) &&
(lastHour < nowHour)) ||
lastDay != nowDay ||
lastMonth != nowMonth ||
lastYear != nowYear ) {

animal.setAnimal_status(3);
animal.setLast_Access_time(nowTime);

}

//주인이 밥을 줄경우 밥먹기(5)액션이 되며 밥먹기가 시작되고 1분이 지나면 기쁨(2)액션으로 전환됩니다.
else if( animal.getAnimal_status() == 5
&& (nowMinute - lastMinute > 1 ||
((60 - lastMinute) + nowMinute > 1) &&
(lastHour != nowHour) ||
lastDay != nowDay ||
lastMonth != nowMonth ||
lastYear != nowYear )) {

animal.setAnimal_status(2);
animal.setLast_Access_time(nowTime);

}
```

PET의 주요 코드

```
//각 기쁨(2) 슬픔(3)상태는 1분간 유지되며 이후 기본(1)액션으로 돌아옵니다.  
else if( (animal.getAnimal_status() == 2 ||  
    animal.getAnimal_status() == 3 )&& (nowMinute - lastMinute > 1 ||  
    ((60 - lastMinute) + nowMinute > 1) &&  
    (lastHour != nowHour ||  
     lastDay != nowDay ||  
     lastMonth != nowMonth ||  
     lastYear != nowYear ))) {  
  
    animal.setAnimal_status(1);  
    animal.setLast_Access_time(nowTime);  
}  
  
//기본(1)액션 혹은 기타(7,8,9)액션 상태에서 5분이 지나면 기타(7,8,9)액션중 하나로 랜덤하게 업데이트 됩니다.  
else if((animal.getAnimal_status() == 1 ||  
    animal.getAnimal_status() == 7 ||  
    animal.getAnimal_status() == 8 ||  
    animal.getAnimal_status() == 9 ) &&  
    (nowMinute - lastMinute >= 5) ||  
    ((60 - lastMinute) + nowMinute >= 5 &&  
    (lastHour != nowHour)) ||  
    (lastDay != nowDay ||  
     lastMonth != nowMonth ||  
     lastYear != nowYear )) {  
  
    int ranum = random.nextInt(2) + 7;  
  
    animal.setAnimal_status(ranum);  
    animal.setLast_Access_time(nowTime);  
}
```

PET TAMING

PET의 주요 코드

동물의 액션

출력 방법



PET TAMING

PET의 주요 코드

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** The left sidebar shows the 'SCHEMAS' section with 'animal_action_db' selected.
- Tables:** Under 'animal_action_db', 'cat_1' is selected, and its structure is shown in the bottom-left pane.
- Query Editor:** The main area displays the result of the query: `1 • SELECT * FROM animal_action_db.cat_1;`
- Result Grid:** The results are presented in a grid format with 20 columns labeled Y, x0, x1, ..., x20. The first column (Y) contains values from 1 to 20. The other columns contain various text values such as 'color_2', 'white', 'Pink', and 'color_1'.
- Toolbar:** The top bar includes standard database management icons like file operations, search, and refresh.
- Status Bar:** The bottom status bar indicates 'Limit to 1000 rows'.

PET의 주요 코드

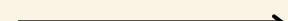
```
//DataBase를 연결을 저장하기 위한 커넥션변수
private Connection conn;

//DataBase를 연결하기위한 매서드
@SuppressWarnings("deprecation")
public boolean open() {

    String id = "root";
    String pw = "admin";

    try {

        Class.forName("com.mysql.cj.jdbc.Driver").newInstance();
        conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/animal_action_db?serverTimezone=Asia/Seoul", id, pw);
        return true;
    } catch (SQLException e) {
        System.out.println("Couldn't connect to database: " + e.getMessage());
        return false;
    } catch (Exception e) {
        System.out.println("Couldn't connect to database: " + e.getMessage());
        return false;
    }
}
```



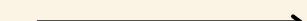
PET의 주요 코드

```
//지정된 동물 객체의 액션을 저장하는 메서드
public void getAnimalAction(TamingAnimals animal) throws Exception {

    Integer animal_id = animal.getAnimalid().getAnimalid();
    String animal_status = String.valueOf(animal.getAnimal_status());

    animal.setAnimalAction(action_Select(animal_id,animal_status));
    animal.setAnimalMoveAction(action_Select(animal_id, animal_status + "_action"));
}
```

```
//동물(animal_id)_행동(actionNum)을 기반으로 애니멀 액션 테이블을 반환하는 메서드
public action action_Select(Integer animal_id, String actionNum) throws Exception {
    String animal = anre.findById(animal_id).get().getAnimal();
    return AnimalDAO.get_animal(animal, actionNum);
}
```

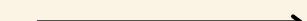


PET의 주요 코드

```
//애니멀 액션 테이블을 반환하기 위한 메서드
public action get_animal(String animal, String actionNum) throws Exception {

    //애니멀 액션 객체의 한 줄을 담을 개체
    action_line Animal_Object = null;
    //완성된 action_line 객체를 순차적으로 모아두는 객체
    List<action_line> Animal = null;
    Statement mySt = null;
    ResultSet myRs = null;

    //애니멀 액션 테이블을 불러오는 쿼리문
    String animal_sql = "SELECT * FROM " + animal + "_" + actionNum;
    open();
```



PET의 주요 코드

```

while(myRs.next()) {
    //y좌표값을 변수로 저장
    int y_Axis = myRs.getInt("Y");
    //작업에는 컬럼의 최대개수가 필요하기 때문에 컬럼의 개수를 저장
    int x_Axis = myRs.getMetaData().getColumnCount();
    //도트 한줄을 저장하기 위해 리스트 초기화
    List<String> x_Axis_list = new ArrayList<>();
    //컬럼의 개수만큼 반복하여 테이블의 x_ + i에 저장된 문자열을 리스트에 저장
    for(int i = 0; i < 21 ; i++) {
        String x = myRs.getString("x" + i);
        x_Axis_list.add(x);
    }

    //저장된 도트 한줄과 작업에 필요한 모든 정보를 객체화후 리스트에 저장
    Animal_Object = new action_line(y_Axis,x_Axis_list);
    Animal.add(Animal_Object);
}

```

```

//모든 도트를 저장한 애니멀 객체를 반환
return new action(Animal);
}
finally {
    SQLClose(conn, mySt, myRs);
}

```

PET의 주요 코드

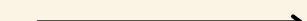
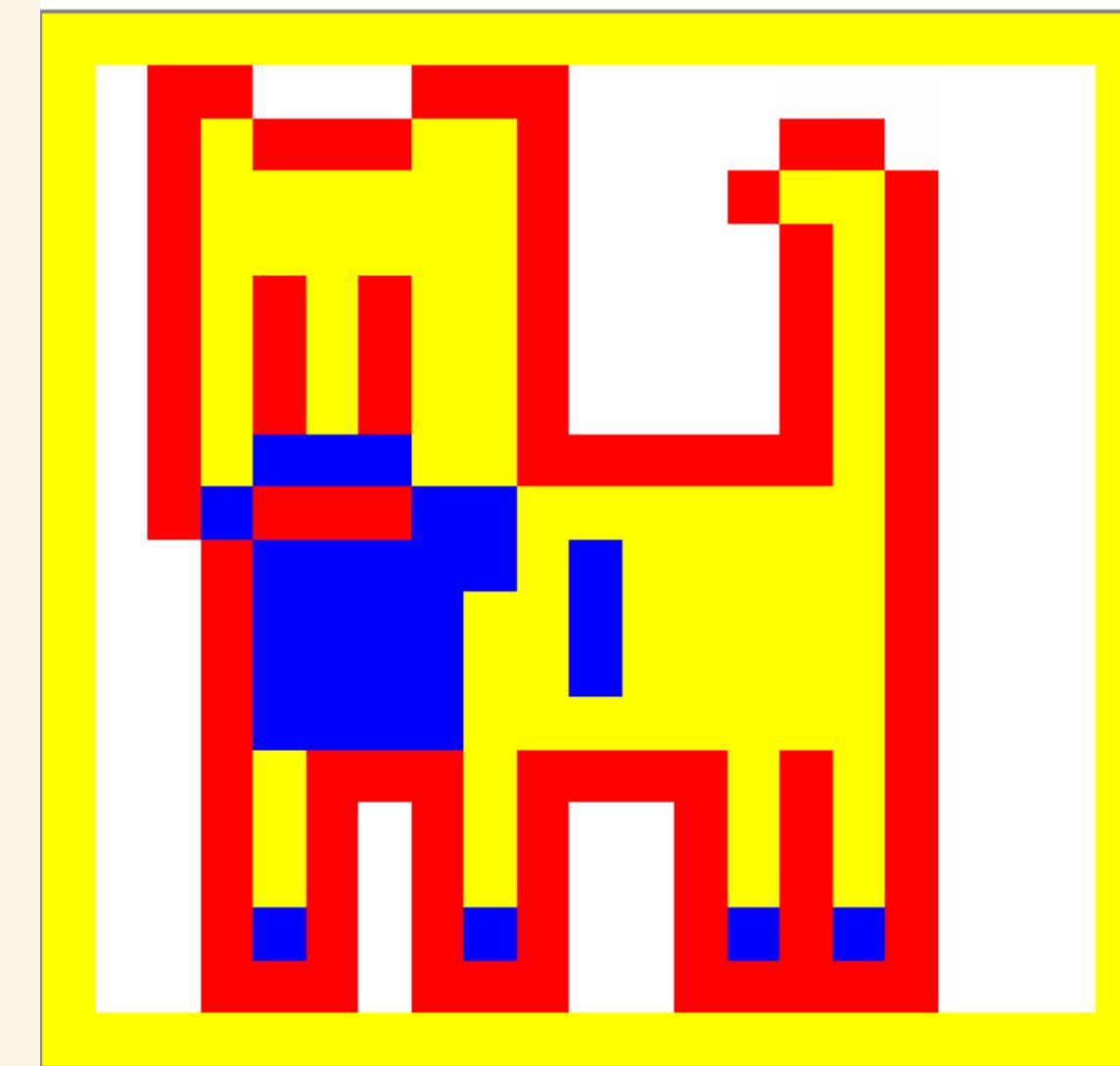
```
<table>
  <tr th:each="action : ${Animals.animalAction.x_axis_line}">
    <div th:each="actionLine : ${action.x_axis}"
      th:switch="${actionLine}">

      <td th:case="color_1"
        th:class="${Animals.color_1}_box"
        style="height: 20px; width: 20px;"></td>

      <td th:case="color_2"
        th:class="${Animals.color_2}_box"
        style="height: 20px; width: 20px;"></td>

      <td th:case="color_border"
        th:class="${Animals.color_border}_box"
        style="height: 20px; width: 20px;"></td>

      <td th:case="*"
        th:class="${actionLine}_box"
        style="height: 20px; width: 20px;"></td>
    </div>
  </tr>
</table>
```



개선점

1. 전반적인 디자인
 2. 펫 검색 기능 구현
 3. 게시글 사진 업로드 구현
 4. 다양한 동물 종류 구현
 5. 펫 그려보기 기능등 구현
 6. 펫 업데이트 기능 구현
-

PET TAMING

THANKS YOU!

PET TAMING PROJECT TEAM
