上海电力大学

本科毕业设计（论文）

题　　目：流浪动物救助平台的设计与实现

院　　系：数理学院数学系

专业年级：信息与计算科学专业 2016 级

学生姓名：脱继虎　学号：20161252

指导教师：冯莉

2020 年　6　月　3　日

摘 要

近年来，随着经济的发展，人民的生活水平日益提高，许多的家庭开始饲养宠物，城市中的宠物数量也是随之大幅度增长。然而，在人们享受着宠物为生活所带来的乐趣的同时，一系列的安全威胁和宠物丢弃等问题也是铺天盖地地袭来。在城市宠物数量在快速增长的同时，许多的宠物由于主人的疏忽而走失，或者因为主人的某些生活原因，一些宠物在主人短暂的宠爱之后被无情的丢弃……这一系列的原因，导致城市中的流浪动物数量快速增长，给居民的安居、交通、卫生、人身安全等造成了严重的威胁。

面对以上问题，虽然城市中存在着一些宠物救助机构和热心居民，时常为它们送去关怀和救助，但是庞大的流浪动物基数以及流浪位置的分散性和不确定性，导致难以管理和救助效果不佳。

流浪动物救助平台是基于 J2EE 设计的管理平台，可以分为普通游客端、注册用户端和管理员端。在该平台上，普通游客端可以浏览别人发布的救助、寻宠、领养等信息，及时的参与到救助中去。而注册用户端不仅可以浏览各类信息，同时也可以登记自己发现的流浪动物的信息，或者是寻找丢失的宠物和为爱宠寻找新的主人等操作。而管理员端，可以发布公告，对注册用户及被救助动物进行管理、登记、审核相关信息以及查询统计等功能。该平台是一个功能较完善、操作简单的平台。

关键词：流浪动物救助平台；JAVA EE；Java

Design and Implementation of a Rescue Platform for Stray Animals

Abstract

In recent years, with the development of economy and the improvement of people's living standard, many families begin to keep pets, and the number of pets in the city also increases greatly. However, while people enjoy the fun of pets for life, a series of security threats and pet discard problems are also overwhelming. At the same time of the rapid growth of the number of pets in the city, many pets are lost due to the negligence of their owners, or some pets are mercilessly discarded after their owners' short-term love for some living reasons This series of reasons lead to the rapid growth of the number of stray animals in the city, which poses a serious threat to the residents' living, traffic, health and life safety.

In the face of the above problems, although there are some pet rescue institutions and enthusiastic residents in the city, they are often sent to care and rescue, but the large number of stray animals and the dispersion and uncertainty of the location of stray animals make it difficult to manage and rescue.

Tramp animal rescue platform is a management platform designed based on J2EE, which can be divided into ordinary tourists, registered users and administrators. On this platform, ordinary tourists can browse the information of rescue, pet seeking, adoption and other information released by others, and participate in the rescue in time. The registered client can not only browse all kinds of information, but also register the information of stray animals found by itself, or search for lost pets and new owners for pets. The administrator side can issue announcements, manage, register, audit relevant information and query statistics for registered users and rescued animals. This platform is a platform with perfect function and simple operation.

Key Words： Rescue Platform for Stray Animals; JAVA EE; Java

目录

第一章 引言

流浪动物救助平台是一个解决城市中庞大的流浪动物群的管理难以及流浪动物救助信息分享难的平台。

近年来，随着计算机技术的发展，利用计算机创建的平台来对城市中的流浪动物的信息进行有效地登记和管理是非常有必要的。通过该平台，人们不仅可以随时随地的将自己发现的流浪动物信息进行登记，同时还可以浏览别人发布的救助信息，并进行评论和留言，与其他的爱心人士共同讨论救助方案等。这一功能极大的解决了流浪动物信息分享难的问题，同时为爱心人士搭建了交流平台，可以让更多的人参与到流浪动物的救助中来。而那些丢失爱宠的主人，也可以快速的将寻宠信息发布到该平台上，让更多的同城居民看到信息，及时提供线索。而那些因为工作或者其他原因不能继续喂养宠物的主人，也可以通过该平台发布为爱宠寻主的信息通知，及时为自己的爱宠找到新家。这些功能不仅可以解决宠物丢失寻找难的问题和降低宠物遗弃问题的发生，从其他方面来讲也可以减少城市流浪动物的增加数量

目前国内外对于流浪动物的解决也是存在不同的问题。流浪动物问题是一个世界性难题，无论是国内国外形式都很严峻。在国外通常采取“收容、认养”的方式来解决流浪动物问题，收容站既有地方政府办的，也有民办的。但是总的收容所数量有限，能够收容处理的流浪动物也是极少数能够被人们所发现的，大多数流浪动物依旧没有被收容甚至是发现。针对流浪动物问题，在国内，目前北京、上海等地也实行“收养、绝育、接种、认养”的方式，并且还建立了一些宠物医院。一些大学校园中也是开设了流浪动物救助的社团，用于解决学校周边的一些流浪动物问题。总的来说，无论是国内国外，对于流浪动物的处理最大问题依旧是流浪动物分散位置的信息获得以及如何有效的将爱心人士统一在一起，共同对更多的流浪动物进行专业科学的管理和救助。

第二章　流浪动物救助平台

2.1 流浪动物救助平台简介

流浪动物救助平台主要是为了解决城市中流浪动物的信息管理问题的平台。该平台主要分为三种用户操作，普通游客端：可以浏览页面发布的公告及流浪动物信息、也可注册成为一名爱心人士；注册用户端：用户的注册、登录、修改个人信息等功能，既可以登记自己发现的流浪动物信息，也可以查询、申请领养救助平台上的流浪动物，并发布其被领养救助动物的日常生活状态；管理员端：管理员端的用户注册、登录、修改密码等功能；管理员可以发布公告、对注册用户及被救助动物进行管理、登记、审核相关信息以及查询统计等功能。

2.2 系统组织结构

开发流浪动物救助平台，游客可以在页面浏览救助、寻宠、领养、咨询等各种信息，也可以注册成为平台用户。注册用户具备信息发布和评论留言的功能，可以发布寻宠、领养、救助等信息，通过管理员的后台审核之后，就可以展示在页面上。而管理员端则可以对所有的注册用户进行增、删、改、查、修改密码等基础操作。同时对于注册用户发布的信息、评论留言等可以进行审核、修改、删除等操作。平台的主要意义是为流浪动物的救助信息、丢失宠物的寻找信息等进行发布共享，让更多的爱心人士参与到救助中来。

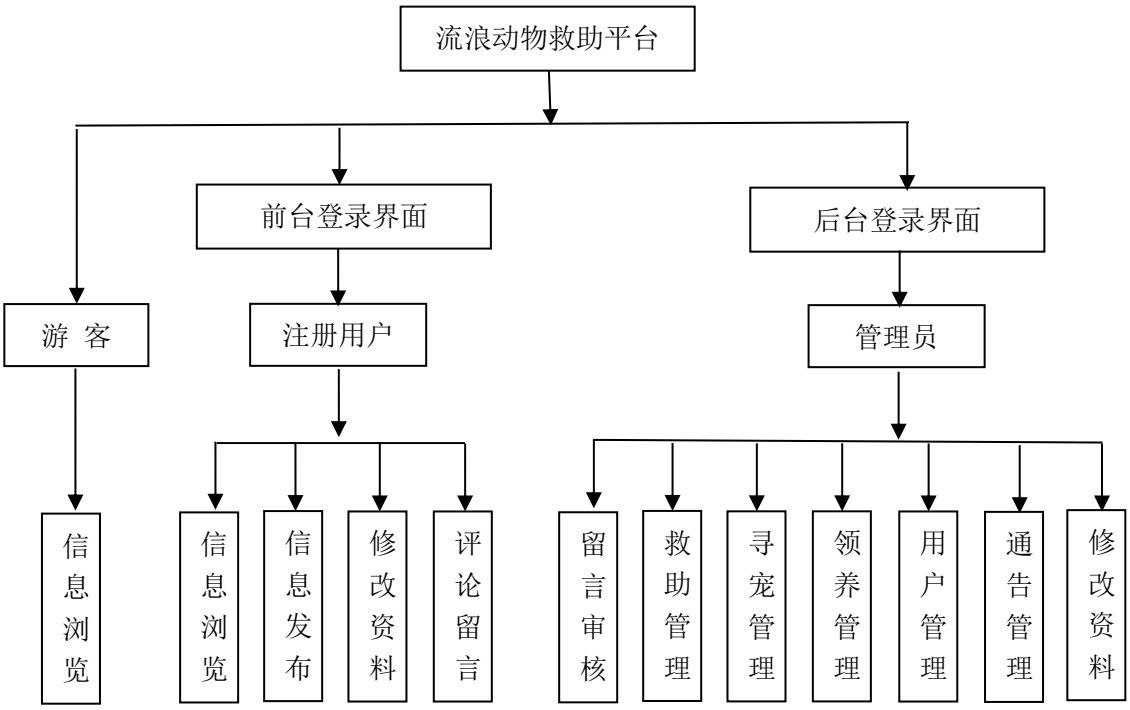

图 2-1　系统组织结构图

2.3 系统角色划分

流浪动物救助平台的用户包括三种，分别是游客端、注册用户端、管理员。然后按照用户的不同用例和需要进行的功能流程分别画出了各个用户的用例图，如下图所示。

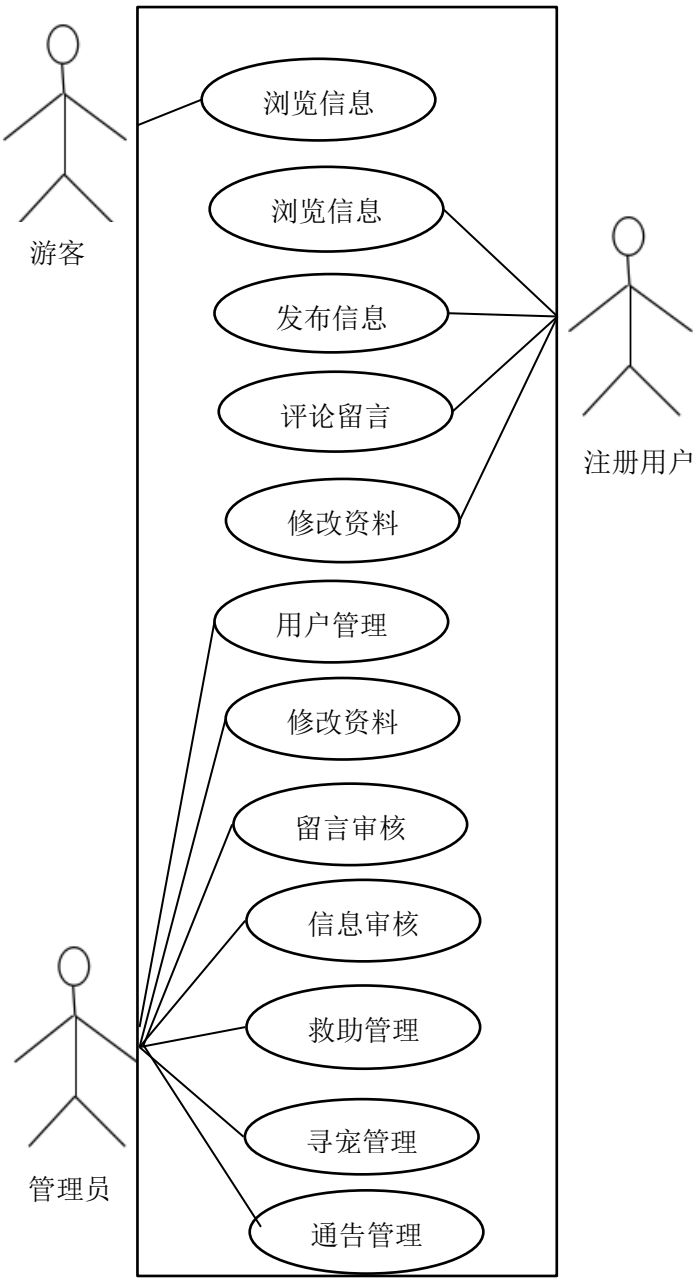

图 2-2　系统用例图

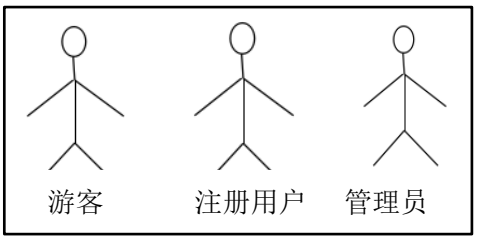

图 2-3　系统结构图

ID：1

用例名称：登录流浪动物救助平台前台 用户：注册用户或管理员

前置条件：无

后置条件：若用例成功，注册用户可以启动发布信息和评论留言以及资料修改的功能。反之，只能浏览页面内的发布信息。

基本事件流：

当注册用户打开网页点击需要登录的用例时，用例启动。

① 注册用户输入用户名和密码。

② 系统根据用户输入的用户名、密码进行验证用户信息，若正确，则用户登录成功，可以使用需要登录才能操作的功能。

ID：2

用例名称：发布信息 参与者：注册用户

前置条件：在启动用例之前，注册用户必须先登录到系统。否则，只能浏览已发布的信息。

后置条件：如果用例成功，注册用户可以发布救助、寻宠、收养等这种信息类。

基本事件流：当注册用户登录成功时，用例启动。注册用户可发布相应的信息。

ID：3

用例名称：登录流浪动物救助平台　用户：管理员

前置条件：角色必须为管理员

后置条件：若用例成功，管理员可以启动审核评论留言、用户管理、领养管理、救助管理、以及资料修改等功能。

基本事件流：

当管理员打开网页点击需要登录的用例时，用例启动。

① 管理员输入用户名和密码。

② 系统根据用户输入的用户名、密码进行验证用户信息，若正确，则用户登录成功，可以使用需要登录才能操作的功能。

第三章　系统设计要求

3.1 设计原则

3.1.1 技术平台选型

流浪动物救助平台是基于 B/S（浏览器/服务器）架构、MVVM 设计模式、J2EE 开发平台的 WEB 开发技术所构建的。考虑到流浪动物救助平台要为全国每个省份都提供服务，如果使用传统的 MVC 模式，这样一个应用被分成三个层，即模型层、视图层、控制层[1]。如果采用 JSP 的形式来展示数据，不仅会造成前后端的高耦合性，如果访问量日益增长的话还会严重影响效率，不利于今后的服务化处理。所以我们采用 MVVM 的设计模式，采用前后端分离的开发模式，为平台今后的服务化拓展提供了良好的基础。

MVVM（Model-View-ViewModel）本质上是 MVC 的改进版，将其中的 View 的状态和行为进行抽象化，让我们将视图 UI 和业务逻辑分开。传统的 MVC 模式，允许 View 和 Model 直接进行通信。而今后随着平台访问量和业务的不断庞大，View 和 Model 之间会出现难以处理的关系，这不符合开发中的“开放封闭原则”。 MVVM 具有低耦合性、可重用性、独立开发以及可测试的优点。

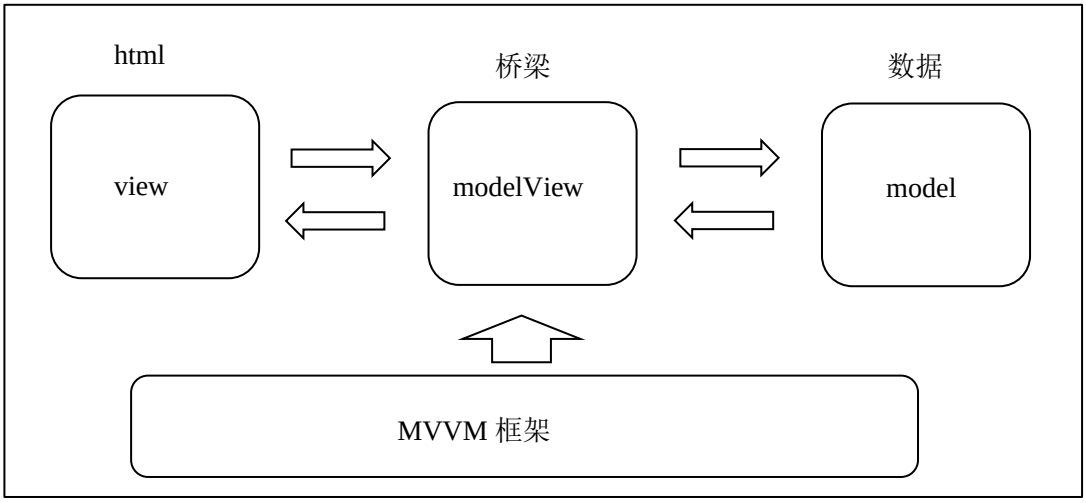

图 3-1 MVVM　技术架构图

J2EE 的全称是 Java 2 Platform Enterprise Edition，它是在 SUN 公司领导下，多家公司参与共同制定的企业级分布式应用程序开发规范。目前，J2EE 是市场上主流的企业级分布式应用平台的解决方案。

在应用服务器选择上，需要考虑性能稳定和后期的扩展等需求，选择适当的服务可以很好的满足整个系统的要求[2]。

流浪动物救助平台通过 J2EE 平台搭建，具有可伸缩性、灵活性、易维护性良好的机制。

3.1.2 系统扩展性

（1）采用前段后分离的模式进行开发，不仅降低了前后端代码的耦合度，同时降低了后台服务器的压力，如访问一些静态资源，如图片、HTML、CSS 等不在需要访问一次就需要使用一次 HTTP 请求获取一次。同时也为今后的服务化打下了基础。

（2）前后端分离的模式可以让代码的开发与维护变得更加方便，使得不同的开发人员可以更加专注于自己擅长的部分，让项目开发得更加有效率[3]。

（3）采用面向对象的分析和设计，采用参数化设计思想，以服务对象为中心，通过对基本业务逻辑元素的灵活组合，可形成新的服务类型或业务模式[4]。

（4）系统应能够适应多种主流主机平台、数据库平台、中间件平台，具有较强的跨系统平台的能力。

（5）Spring 框架的应用，降低了系统架构和代码之间的耦合性，让系统更灵活[5]。

（6）采用“前瞻设计，逐步扩充，分期实现”的设计原则，采用参数化、组件化的设计思想，从而保证系统的可靠性[6]。

3.1.3 系统性能

稳定性：流浪动物救助平台中用户的操作涉及到许多的数据关联处理，会给系统造成了巨大的压力，影响到系统的稳定性和反应力，所以确保系统的稳定性和高效性很重要[7]。

安全性：流浪动物救助平台使用了 JWT 请求鉴权来确保访问的安全性。对于用户密码使用 MD5 加盐的形式来确保安全性。

3.2 系统概念结构设计

3.2.1 系统 ER 图

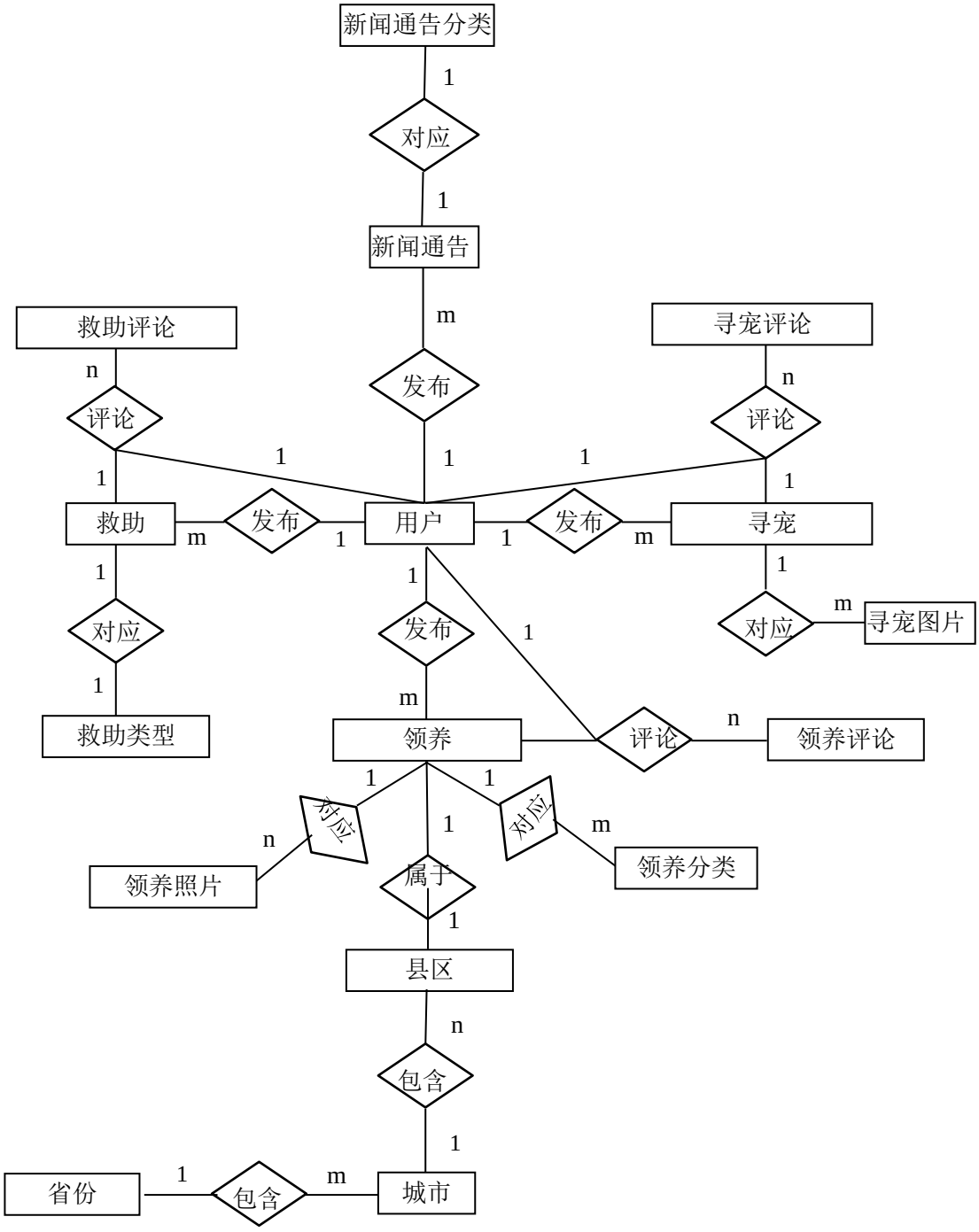

图 3-2　系统 ER 图

3.2.2 关系模型

用户表（ID、用户名、密码、邮箱、照片、生日、性别、手机、是否删除、创建时间、角色、盐值）

角色表（ID、名称）

领养表（ID、标题、宠物性别、县区、年龄、描述、联系人、微信、手机、创建

时间、宠物分类、宠物状态、用户 ID、是否删除、是否审核）

领养图片表（ID、名称、路径、领养 ID）

领养评论表（ID、评论者 ID、内容、是否删除、评论时间、领养 ID）

寻宠（ID、标题、宠物性别、县区、年龄、描述、联系人、微信、手机、创建时间、宠物分类、宠物状态、用户 ID、是否删除、是否审核）

寻宠图片表（ID、名称、路径、寻宠 ID）

寻宠评论表（ID、评论者 ID、内容、是否删除、评论时间、寻宠 ID）

救助报告表（ID、标题、内容、城市 ID、创建时间、更新时间、审核状态、删除状态、分类、用户 ID）

救助报告评论表（ID、评论者 ID、内容、是否删除、评论时间、救助报告 ID）

救助报告分类表（ID、类型）

新闻通告表（ID、标题、内容、照片、纯文本信息、创建时间、用户 ID、删除状态、审核状态、新闻通告分类 ID）

新闻通告分类表（ID、类型）

省份表（ID、名称）

城市表（ID、名称、省份）

县区表（ID、名称、城市 ID）

宠物分类表（ID、类型）

宠物分类详情表（ID、名称、宠物分类 ID）

3.2.3 数据库设计需求

数据库设计是信息系统设计与开发的关键性工作，一个信息系统的好坏在很大程度上取决于数据库设计的好坏[8]。数据库选用：流浪动物救助平台后台数据库选用为 MySQL，相较于现在主流的数据库，如 Oracle、SQL Server 等其他数据库，MySQL 具有开源、体积小、速度快、成本低的特点[9]。

根据需求分析，我们创建的主要实体有：用户、寻宠、领养、新闻通告、救助报告、省份城市、宠物分类、留言评论等实体，数据库的设计就是根据这些实体类的关系展开构建完成的。其中用户表中除了基本的用户名和密码信息之外，考虑到发布信息之后用户之间的沟通交流问题，我们还加入了手机号、微信等联系方式字段。对于密码安全性问题，我们额外设计了一个“salt”字段用来保存盐，再搭配 MD5 加密，来保证用户密码的安全性。此外，还设计了邮箱字段来进行用户注册，用户注册必须通过该邮箱获取验证码并输入正确之后才可以进行注册。而寻宠、领养表的结构比较相似，包含基本的宠物信息和联系人信息，如数量、性别、位置、照片、联系人名称、手机号、微信等字段。新闻通知表和救助报告表结构相似，除了创建时间和删除、审核状态之外，所有的文本、图片都存在一个“content”字段之中。因为前端我们引入了富文本框，因此所有的内容都保存在这一个字段之中即可。而且，在我

们设计删除操作的时候考虑到安全性问题，统一采用逻辑删除。即所有要删除的表中都会有一个“delete_status”字段，来控制该条内容是否被删除。考虑到数据表较多和部分表结构相似问题，下面只展示流浪动物救助平台使用到的部分主要的数据表：

表 3-1　用户表

字段名称	字段描述	数据类型	长度	是否允许空	缺省值	备注
id	用户 ID	bigint	20	N	NULL	主键
username	用户名	varchar	50	N	NULL	
password	密码	varchar	32	N	NULL	
email	邮箱	varchar	20	N	NULL	
photo	头像	varchar	255	Y	NULL	
birthday	生日	varchar	20	Y	NULL	
sex	性别	tinyint	1	Y	NULL	
phone	手机号	varchar	20	Y	NULL	
delete_status	删除状态	Tinyint	1	N	0	
create_time	创建时间	varchar	50	N	NULL	
sa_admin_category_id	角色分类 ID	tinyint	1	N	NULL	外键
salt	盐	Varchar	100	N	NULL	

表 3-2　领养表

字段名称	字段描述	数据类型	长度	是否允许空	缺省值	备注
id	ID	bigint	20	N	NULL	主键
title	标题	varchar	100	N	NULL	
sex	宠物性别	varchar	20	N	NULL	
sa_county_id	县区 ID	bigint	20	N	NULL	
amount	数量	tinyint	1	N	NULL	
age	宠物年龄	tinyint	1	N	NULL	
message	描述信息	varchar	255	N	NULL	
person_name	联系人姓名	varchar	50	N	NULL	
we_chat	微信号	varchar	20	Y	NULL	
phone	手机号	varchar	20	N	NULL	
create_time	创建时间	varchar	20	N	NULL	
sa_animal_status_id	宠物状态 ID	tinyint	1	N	2	外键
sa_animal_category_details_id	宠物分类 ID	tinyint	1	N	NULL	外键
sa_user_id	用户 ID	bigint	20	N	NULL	外键
delete_status	删除状态	tinyint	1	N	0	
verify_status	审核状态	tinyint	1	N	0	

表 3-3　救助报告表

字段名称	字段描述	数据类型	长度	是否允许空	缺省值	备注
id	课号	bigint	20	N	NULL	主键
title	标题	varchar	100	N	NULL	
content	内容	longtext	0	N	NULL	
sa_city_id	城市 ID	bingint	20	N	NULL	外键
create_time	创建时间	datetime	1	N	NULL	
update_time	更新时间	datetime	1	N		
verify_status	审核状态	tinyint	1	N		
delete_status	删除状态	tinyint	1	N	0	
sa_report_type_id	类型	tinyint	1	N		外键
sa_user_id	用户 ID	bigint	20	N		外键

表 3-4　宠物分类表

字段名称	字段描述	数据类型	长度	是否允许空	缺省值	备注
id	ID	tinyint	1	N	NULL	主键
name	分类名称	varchar	10	N	NULL	

表 3-5　宠物分类详情表

字段名称	字段描述	数据类型	长度	是否允许空	缺省值	备注
id	ID	tinyint	1	N	NULL	主键
name	分类名称	varchar	10	N	NULL	
sa_animal_category_id	分类 ID	tinyint	1	N		外键

表 3-6　省份表

字段名称	字段描述	数据类型	长度	是否允许空	缺省值	备注
id	ID	bigint	20	N	NULL	主键
name	名称	varchar	10	N	NULL	

表 3-7　城市表

字段名称	字段描述	数据类型	长度	是否允许空	缺省值	备注
id	ID	bigint	20	N	NULL	主键
name	名称	varchar	10	N	NULL	
sa_province_id	省份 ID	bigint	20	N	NULL	

第四章　系统实现

4.1 系统架构

流浪动物救助平台采用的是前后端分离的模式来实现，可以分为两个部分：后台管理系统、前台门户系统。

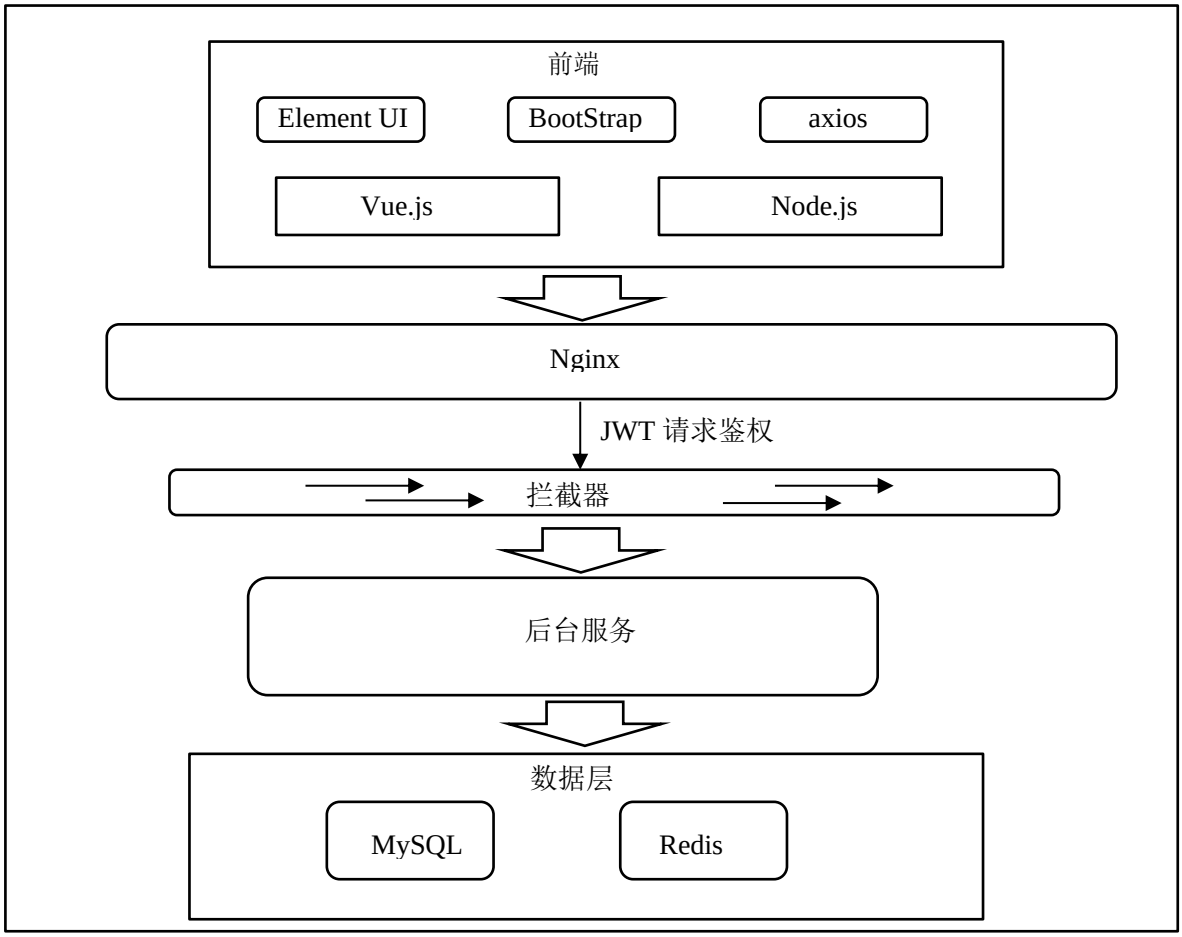

图 4-1　系统架构图

前端主要架构为：Vue.js、Node.js、axios、jQuery、Bootstrap、Element UI。
后端主要架构为：Spring、Spring Boot、MyBatis Plus、Nginx、JWT、Redis。
项目运行环境：JDK 1.8、Tomcat 9、Maven 3.6.1、MySQL 5.6.
开发工具：Idea 2018.3.3、Navicat for MySQL。
前台域名：www.sa.com（sa 是 stray animals 的缩写）。
后台域名；api.sa.com。
后台管理系统域名：manage.sa.com。

4.2 前台门户系统

前台门户系统是基于 Vue 的前端项目，开发的时候项目运行在 live-server 服务器中，后期需要部署到 Nginx 服务器中。服务端基于 Spring Boot、MyBatis Plus 构建，Spring 从一开始直到现在，而且包括以后，将持续获得更多开发者的青睐，因为它是优秀的、是为开发者设计的实用框架[10]。而 MyBatis 几乎能做到 JDBC 所能做到的所有事情[11]。

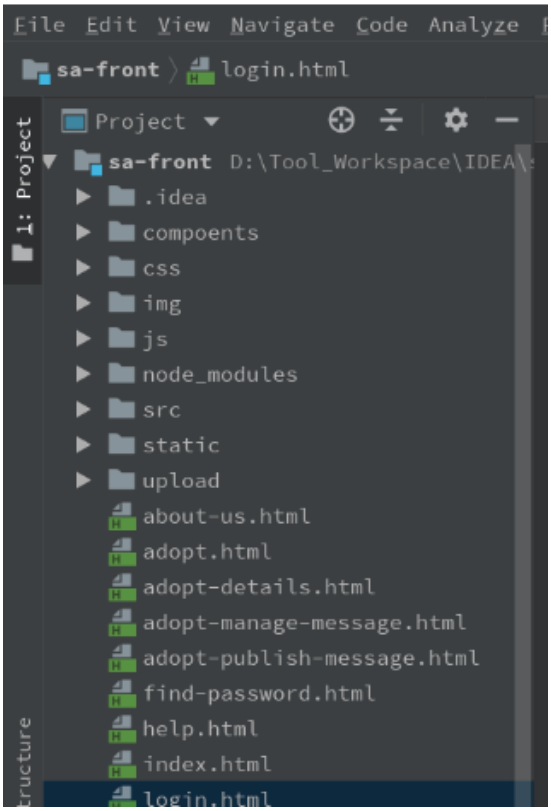

图 4-2　前端项目结构图

前台页面的构建较为简单，依然是采用 HTML5、CSS3、JavaScript 来完成基础页面的构建和样式的搭配，以及前端数据的验证。

在与后台交互和数据传输方面采用 Vue 和 axios，相对于比较传统的 jQuery 和 AJAX，访问 API 的函数更加的简单，这极大的提高了前端开发的效率。而在获取到后台数据后，只需要将数据保存在 Vue 的 data 范围中，页面需要的时候直接使用差值表达式将数据标记到页面需要的地方，等待页面渲染的时候会自动获取数据。这相比较之前的 DOM 操作，数据处理起来更加的直接和简便，我们不在需要操作复杂的 DOM 节点。

```
new Vue({
    el: "#Vue-app",
    components: {
        shotcut: () => import('./js/pages/shotcut.js'),
        ue: () => import('./js/pages/UEdit.js')
    },
    data: {
        report: {
            typeId: "",
            title: "",
            content: "",
            createTime: "",
            updateTime: "",
            userId: "",
            cityId: ""
        },
```

图 4-3　Vue 数据处理图

其次是使用了 Vue 的组件化技术，来减少页面重复内容的开发。如头部信息、脚部信息、导航栏等公共部分，将其抽取出来做成组件，在需要的页面引入即可。

在处理新闻通知、救助报告等部分，前端页面集成了百度的富文本框 UEditor，用户就可以像操作 Word 文档一样来书写自己想要的文本内容和插入需要的图片等操作。

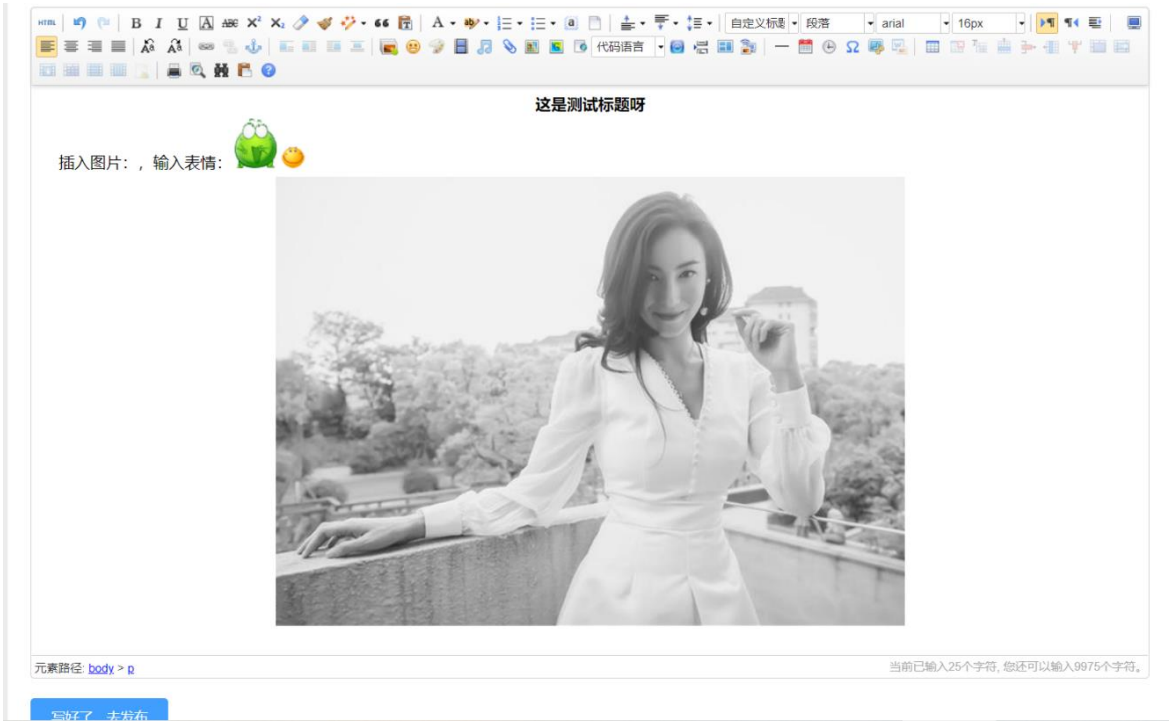

图 4-4　富文本框图

服务端代码我们采用严格的分层设计来实现。

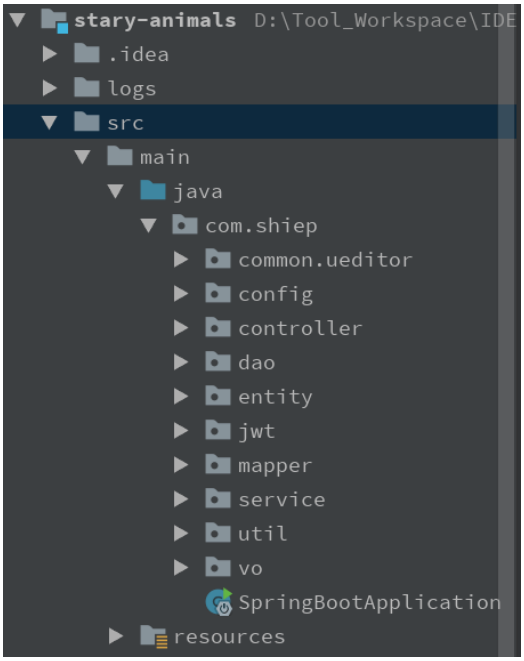

图 4-5　服务端代码架构图

其中，各个包的信息如下：

（1）common.ueditor：用来保存百度富文本框的一些依赖类和配置类。

（2）config：保存的是一些配置类，包括处理跨域请求的 CORS 配置类和 Redis 配置类等。

（3）entity：保存的是与数据表严格映射的实体类。

（4）vo：保存的是前端页面展示时需要的对象。

（5）controller：保存的是所有的 API 请求类。

（6）dao：用来保存实体类的访问数据层的一些方法。

（7）jwt：放置的是与 JWT 鉴权有关的配置，主要是关于 Token 的生成、过期验证等的一些类和方法。

（8）mapper：保存的是集成了 MyBatis Plus 的通用 Mapper 接口，也是提供一些访问数据库的基础方法。

（9）service：用来保存具体的业务实现类。

（10）util：用来保存一些工具类，如项目中使用到的文件上传工具类、Redis 操作工具类、验证码自动生成工具类等。

4.2.1 登录模块

登录模块的主要功能包括：用户登录、用户资料修改、退出登录、忘记密码、用户注册等功能。其中退出登录之后会回到登录页面，同时会清除掉浏览器中 LocalStore 中储存的 token 和用户基本信息。

(1) 登录页面：页面使用 HTML 和 CSS 来布局构建，前台数据验证使用 JavaScript 来验证。使用 axios 来实现服务端登录 API 的访问。该页面主要包括需要登录的用户名和密码，当输入正确时便可以登录到系统，未输入信息或者输入错误时会提醒用户名或密码错误[12]。

图 4-6　登录页面

（2）忘记密码界面：用户需要输入用户名和注册时的邮箱，验证通过后会将重置后的密码发送到用户邮箱。因为我们数据库中储存的密码是经过 MD5 加密的，MD5 是不可逆的，所以这里提供了重置密码的功能。邮件功能是整合了腾讯的 QQ 邮箱服务，可以利用邮箱向任何邮箱发送邮件。

图 4-7　找回密码页面

而重置后的密码为“shiep”加上随机的六位数字组成。

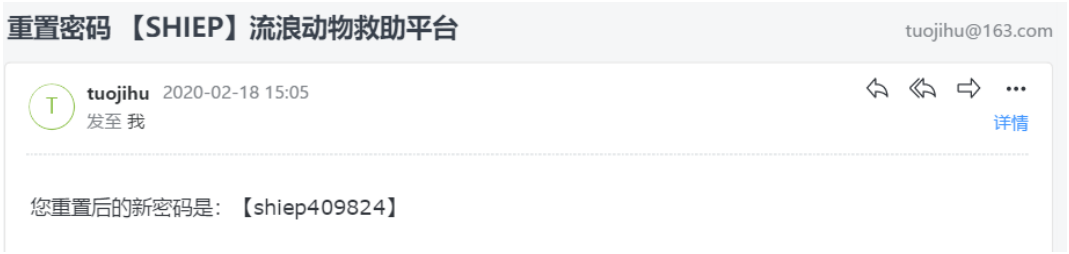

图 4-8　找回密码邮件

（3）用户资料修改界面：用户资料修改界面主要由三部分组成：基本资料修改、头像修改、密码修改。

首先是用户基本资料修改界面，该页面内用户可以修改生日、性别、用户名等基础信息。页面内的数据刷新采用 AJAX 异步刷新，即点击提交修改后会将数据更新到数据库中，同时获取到最新的信息显示到页面上。

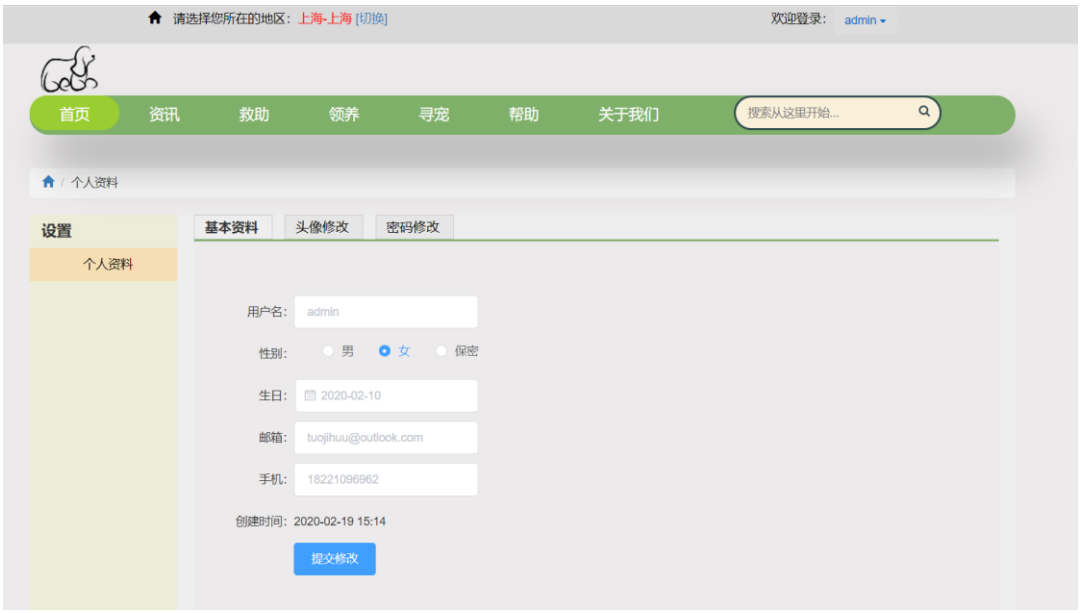

图 4-9　用户基本资料修改

然后是头像修改界面，进入到该界面的时候会提前将用户头像加载到页面左侧进行显示。而右侧的修改处引用了 Element UI 的上传组件，点击上传成功后会对图片进行回显，并提示用户修改成功。

图 4-10　用户头像修改

这里我们展示图片上传组件的部分代码：

```
<el-upload
        style="position: relative;left: 140px"
        :header="header"
        class="avatar-uploader"
        action="http://api.sa.com/stary-animals/user/updatePhoto/"
        :data= "data"
        :show-file-list="false"
        :on-success="handleAvatarSuccess"
        :on-error="errorPhoto"
        :before-upload="beforeAvatarUpload">
        <img v-if="updatePhoto" :src="updatePhoto" class="avatar">
        <i v-else class="el-icon-plus avatar-uploader-icon"></i>
</el-upload>
```

其中，“action”中的内容是服务端的头像上传的 API 的映射路径，“on-success”是图片上传成功后的回调函数，“on-error”是图片上传失败时的回调函数，“before-upload”是图片上传前的函数，可以用来对图片格式和大小等做出一些规范或者过滤操作。这里的“data”中对应的是 Vue 数据中的 data 数据，里面放置的是 token 信息。因为在访问后端的时候请求头 Header 中必须要带上后端生成的 Token 信息才可以正常访问服务端接口，所以我们将 Token 信息保存到 data 数据中一起发送到服务端。

最后是密码修改界面，考虑到用户密码安全性问题，再修改密码的时候必须要

提供注册时的邮箱才可以修改密码。

图 4-11　用户密码修改

这里关于数据是否输入以及两次新密码是否一致，我们都是采取了 JavaScript 来进行验证，提示信息使用 Element UI 的消息提示。如果两次密码输入不一致，会提醒用户。

图 4-12　用户密码修改提示信息

（4）用户注册界面：用户注册时需要输入用户名、密码、邮箱等基础信息。其中邮箱用来发送注册时的验证码，只有用户正确输入 6 位随机验证码之后才能注册成功。这里我们会将随机生成的六位验证码保存到 Redis 中，并设置了过期时间为 60 秒，过期后用户必须重新获取验证码。关于邮件服务，我们使用 Spring Boot 整合了腾讯 QQ 邮件服务，可以向邮箱发送内容。

图 4-13　用户注册页面

验证码获取邮件范例为：

图 4-14　用户注册邮箱页面

这里我们展示邮件发送的部分代码：

```
@Override
public Boolean sendMailAndGenerateCode(String mailAddress) {
    //  邮箱正则
    String check = "^([a-z0-9A-Z]+[-|\\.]?)+[a-z0-9A-Z]@([a-z0-9A-Z]+(-[a-z0-9A-Z]+)?\\.)+[a-zA-Z]{2,}$";
    Pattern regex = Pattern.compile(check);
    Matcher matcher = regex.matcher(mailAddress);
    boolean isMatched = matcher.matches();
    // null 或者不是邮箱
    if (StringUtils.isBlank(mailAddress) || !isMatched) {
        return false;
    }
```

```
            // 生成随机的六位验证码
            String code = NumberUtils.generateCode(6);
            // 将 code 存入到 Redis 中
            boolean flag = this.redisUtil.set(mailAddress, code, 60 / 1000);
            if (!flag) {
                return false;
            }
            try {
                SimpleMailMessage message = new SimpleMailMessage() ;
                message.setFrom("tuojihu@qq.com");   // 你自己的邮箱地址
                message.setTo(mailAddress); //接受者的邮箱
                message.setSubject("欢迎注册【SHIEP】流浪动物救助平台");
                message.setText("您的注册验证码是：" + "【" + code + "】, " + "
验证码的有效时间为 60 秒！ ");
                this.javaMailSender.send(message);
            } catch (Exception e) {
                e.printStackTrace();
                return false;
            }
            return true;
    }
```

首先我们会使用邮箱的正则来判断用户输入的邮箱是否合法，如果不合法，直接返回 false，前端会提示用户错误信息。如果正确，我们创建邮件发送对象，然后输入相关的邮箱以及主题内容，并将随机生成的验证码存到 Redis 中。

4.2.2 首页简介

首页是一个综合页面，包括了我们的寻宠、咨询、领养、救助等等方面的部分信息。整体的页面使用 HTML、CSS、JavaScript 来构建。导航栏使用的是 jQuery 导航组件实现。整个页面分为五个部分：头部、导航栏、咨询部分、领养部分、寻宠部分、脚部。头部、导航栏、脚部都是公共部分，我们实现了组件化，即页面内调用即可。其他的咨询、领养、寻宠等部分是选取部分内容来进行展示。

图 4-15　首页

考虑到地区问题，用户登录之后可以使用头部的切换按钮来点击更换自己所在的城市，查看该城市的相应信息。

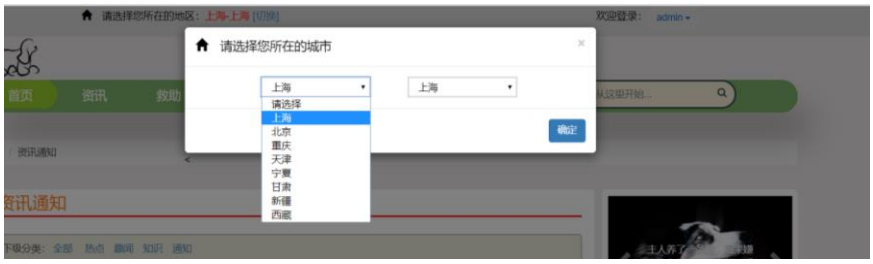

图 4-16　城市切换界面

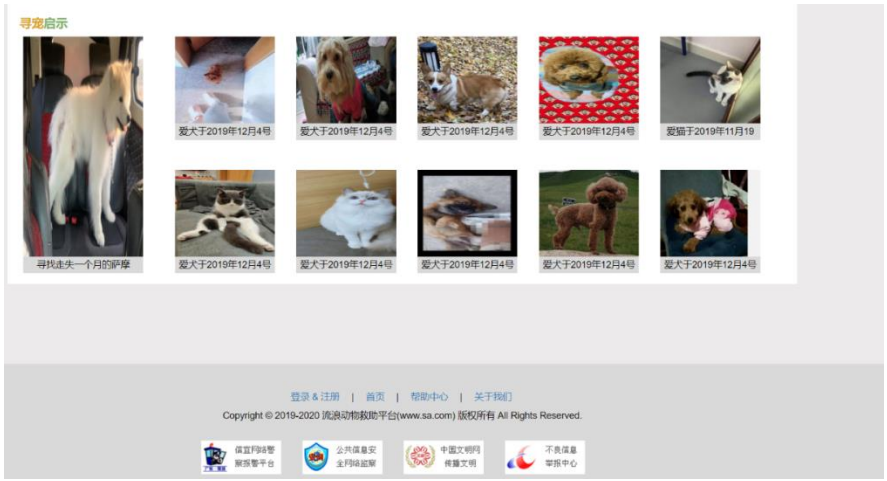

图 4-17　寻宠界面

页面内每条内容都是采用超链接来实现，当点击的时候会自动跳转到相应界面，进行详细内容的展示。具体页面内容我们到后面介绍。

4.2.3 咨询模块

咨询模块在前台部分主要涉及到的就是查看部分，而发表部分则是在后台管理系统中，必须由管理员发布。页面展示左侧是信息展示部分，上面有一个类别过滤器，用户可以选择不同的种类来具体查看。右侧是一个热点信息展示的部分，用户通过点击可以跳转到相应界面。主要的数据采用分页的形式来展示，当内容过多的时候会自动进行分页。

图 4-18　咨询界面

前台分页采用的是Element UI的分页组件，服务端采用了Page Helper分页组件，会对我们查询主表的内容进行截断，并根据页面显示内容条数来查询相应的内容。下面是服务端分页的部分代码：

```
    @GetMapping("getAllNewsVosInPage")
     public ResponseEntity<PageInfo<NewsVo>> queryAllCommentInPageBack(
                Integer size, Integer page, Integer typeId) {
        Page p = PageHelper.startPage(page, size);
        List<NewsVo> newsVos = this.newsService.queryAllNewsVos(typeId);
        PageInfo<NewsVo> pageInfo = new PageInfo(newsVos);
        pageInfo.setPages(p.getPages());//总页数
        pageInfo.setTotal(p.getTotal());//总条数
        return ResponseEntity.ok(pageInfo);
    }
```

其中 size 参数和 page 参数是与分页相关的参数，typeId 参数是咨询过滤器字段。而当我们点击具体的内容时，会进入到咨询详情页面。

图 4-19　咨询详情界面

该页面的内容时采用百度的富文本框 UEditor，由管理员在后台管理系统的发布界面编辑。数据库中储存了相应的内容以及 HTML 标签，然后将其解析到要显示的部分，就会将格式、图片等内容还原出来。

4.2.4 救助模块

该模块是专门为流浪动物的救助开设的，用户可以发布自己发现的流浪动物信息，也可以浏览别人发布的信息，并做出评论。同样，这里对内容设置了两个过滤器，一个是类型过滤器，另一个是时间过滤器。过滤操作都是在后台代码实现，通过对字段内容进行控制，查询出相应的内容。而点击发表帖子的时候，会自动跳转到页面底部的发布处，这里是使用 HTML 中的锚点功能完成。

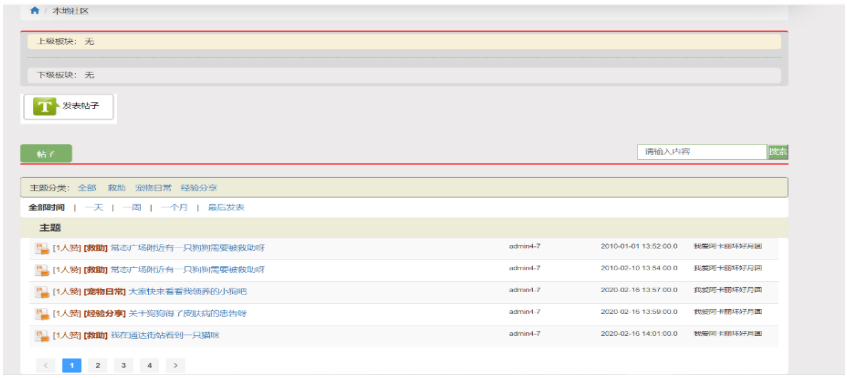

图 4-20　救助界面

底部的发布功能是集成了百度的富文本框 UEditor，用户可以随意地插入自己的文本类型和图片等。当用户输入了发帖类型和内容之后，就可以点击发布了。

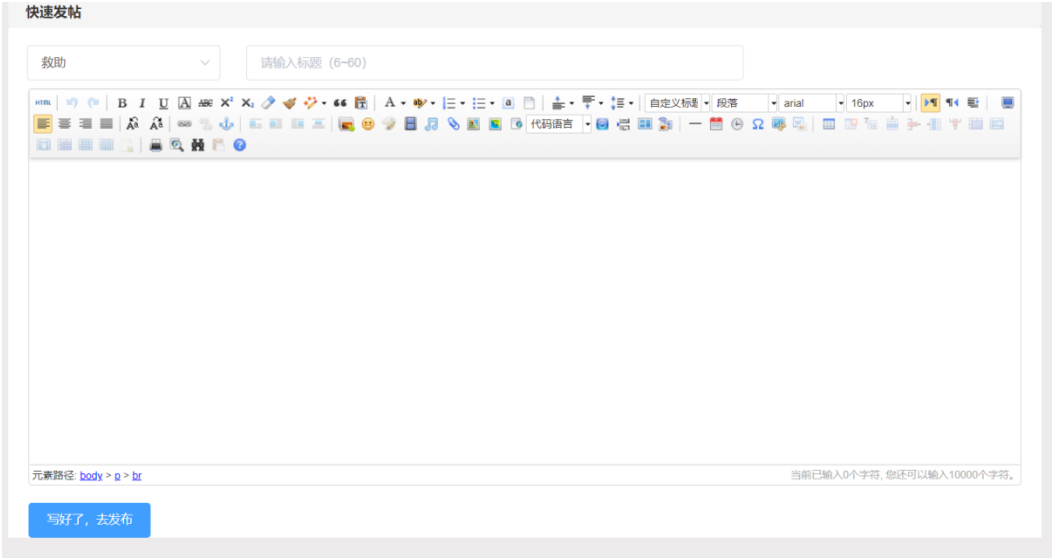

图 4-21　快速发帖界面

这里我们展示后台分页过滤器的部分实现代码：

```
        @Override
         public List<AnimalAdoptVo> queryAll(Long cityId, Integer categoryId, Long countyId) {
                List<AnimalAdopt> animalAdopts = null;
                // 1、）什么都不做　首先必须处理的是 cityId 如果为 null　查询所有 adopt
                if (cityId == null) {
                    QueryWrapper<AnimalAdopt> qw = new QueryWrapper<>();
                    qw.lambda().eq(AnimalAdopt::getDeleteStatus, 0);
                    animalAdopts = this.animalAdoptMapper.selectList(qw);
                    List<AnimalAdoptVo> animalAdoptVo = this.getAnimalAdoptVo(animalAdopts);
                    return animalAdoptVo;
                }
                // 2、）只能点击宠类过滤，即只做宠类过滤 县区为 null
            if (cityId == null && categoryId != null && categoryId != 0) {
            ……
```

这里我们要处理多对多的关系，所以前台展示的时候需要和救助相关的所有内

容，而我们的内容分开在多张表中，所以创建了 Vo 对象进行数据的组装，将各个表中的数据查询出来之后，组装到 Vo 中传到前端进行展示。考虑到代码的复用性，我们将这部分的内容单独抽离到“getAnimalAdoptVo”方法中来实现。

当用户点击具体的救助信息时，会跳转到相应的详情界面，界面左侧是发表的用户头像和名称，右侧是内容。

图 4-22　救助详情界面

而浏览用户还可以点击回复按钮来实现对该帖子内容或者用户的评论进行评论：

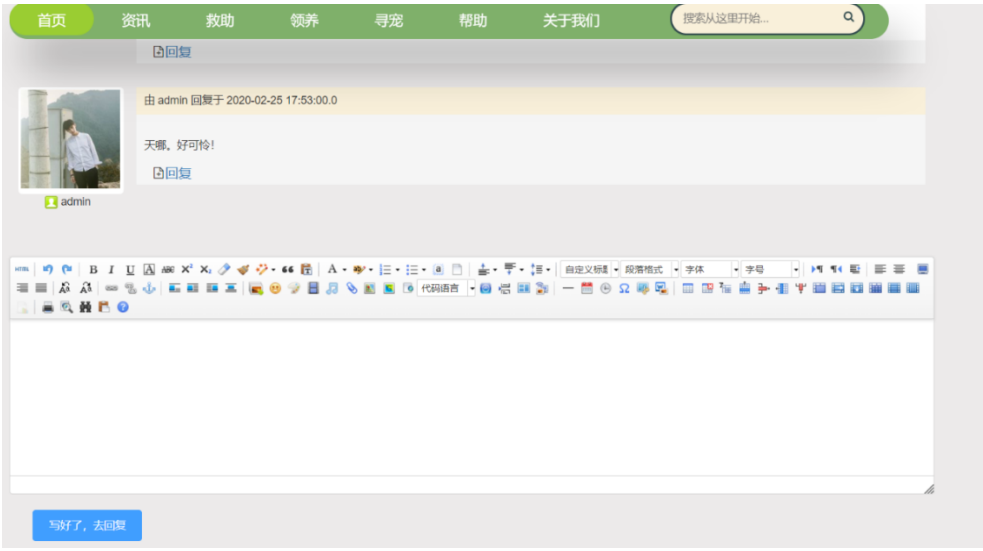

图 4-23　救助帖子回复界面

回复部分也是采用了富文本框的形式来实现。

图 4-24　救助评论界面

这里的评论单独存在于一张表中，记录了评论者 Id、评论时间、评论内容、帖子 Id 等信息。在渲染该 HTML 页面内容的时候，将内容详情和评论同时查询出来，渲染到节点上即可。下面是回复评论的 JavaScript 方法：

```
publishMsg() {
    this.adoptComment.adoptId = this.id;
    this.adoptComment.commentUserId = ly.store.getJson("user").id;
                    this.adoptComment.commentUserName                    =
ly.store.getJson("user").username;
                    this.adoptComment.commentTime = ly.formatDate(new Date(),
"yyyy-MM-dd hh:mm");
                    let _this = this;
                    ly.http.post("/adopt/saveComment",
ly.stringify(this.adoptComment))
                        .then(resp => {
                            if (resp.status === 200) {
                                _this.showSuccessMsg("留言成功！");
                                _this.adoptComment.content = "",
                        ……
```

这里当回复成功后，执行了“_this.queryCommentByAdoptId();”方法重新获取数据，也就做到了异步刷新的操作。

4. 2. 5 领养模块

该模块主要是为了解决宠物丢弃问题而设计的模块，如果主人因为某些工作原因等等，不能继续喂养宠物，就可以在该模块发布领养信息，为自己的爱宠寻找一个新家。而那些有意领养小动物的爱心人士，可以通过该页面寻找自己喜欢的小动物进行领养。

该页面顶部是过滤器内容，可以过滤到该城市的具体县区和自己喜欢的宠物类

别。过滤操作都是在后台代码实现，通过传入的过滤字段的内容查询相应的信息返回给前端即可。左侧是具体的信息展示，可以点击进行详情查看。右侧是发布信息和信息管理的按钮。

图 4-25　领养界面

当我们点击具体信息的时候，会跳转到详情页：

图 4-26　领养详情界面

该界面主要是对具体的信息进行显示，如宠物分类、性别、地址、微信号、手机号、图片、具体描述信息等。当鼠标滑过图片列表的时候，会将其放大到左上的“div”框中进行显示。页面右侧还有发布者的一些信息。浏览用户还可以在页面底部进行评论留言：

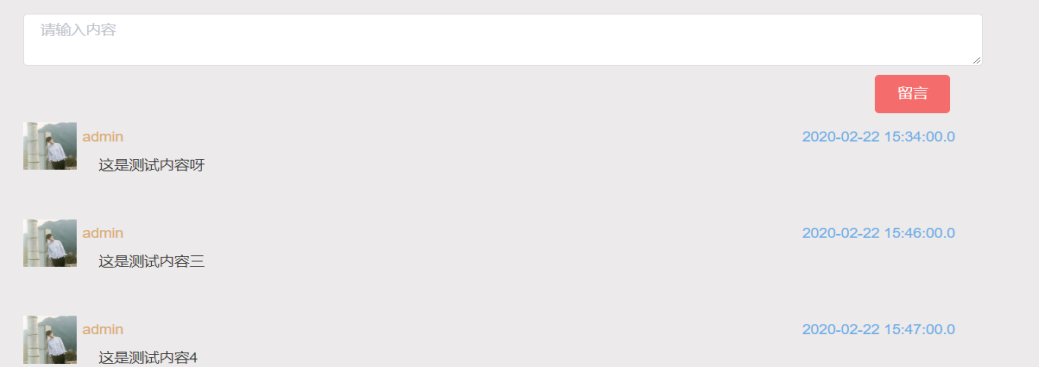

图 4-27　领养留言界面

用户留言成功后异步刷新数据，将评论内容显示到页面。这里我们展示留言部分的后台代码：

```
@Override
public boolean saveAdoptComment(AnimalAdoptComment adoptComment) {
        if (adoptComment == null) {
            return false;
        }
        return this.animalAdoptCommentMapper.insert(adoptComment) > 0;
}
```

这里我们调用了 MyBatis Plus 的通用 Mapper 的 insert 方法，只要将评论的对象传入，即可自动插入到数据库中。

当用户点击发布信息按钮的时候，就会进入到领养信息发布页面，该界面详细的列出了需要填写的各种信息，宠物的种类和县区选择都是在页面渲染的时候异步的将数据库中存储的信息查询后渲染到相应的选择器上。图片上传使用的是 Element UI 的上传组件，限制了上传的数量为 3 张，而且上传成功后会进行回显操作。

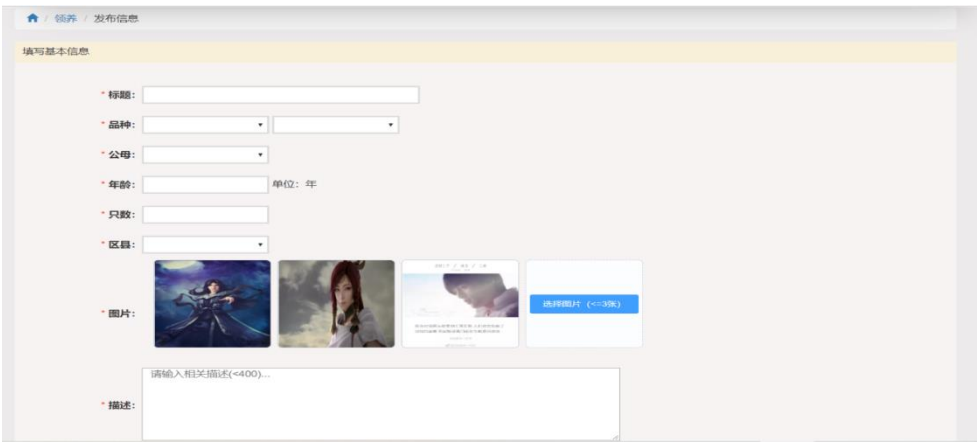

图 4-28　领养信息发布界面

这里我们展示信息发布的部分前端代码：

```
publish() {
        let _this = this;
        //设置创建时间
        _this.animalAdopt.createTime = ly.formatDate(new Date());
        ly.http.post("/adopt/save", leyou.stringify(this.animalAdopt)
        ).then(resp => {
            if (resp.status === 200) {
                // 将最新的值赋给 id，这个 saveAnimalAdoptId 不为 null，才可以上传图片
                _this.saveAnimalAdoptId = resp.data;
            ……
```

当用户点击信息管理的时候会跳转到信息管理界面，该界面主要是展示用户发布的所有领养信息的基础信息，包括标题、审核状态、发布时间等等。用户可以删除自己发布的信息。考虑到富文本框内容修改的复杂性，暂时不提供内容修改功能。

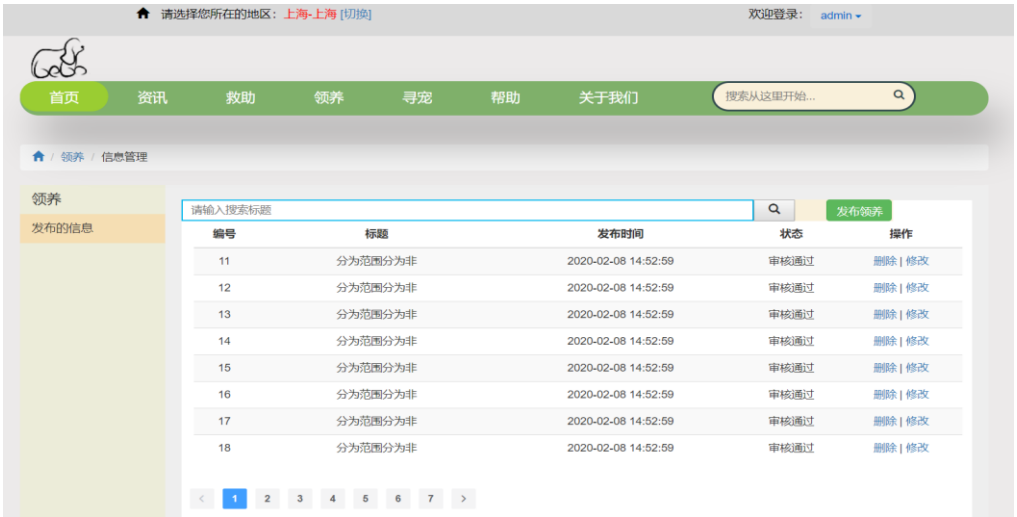

图 4-29　领养信息管理界面

4.2.6 寻宠模块

寻宠模块的设计主要是为了解决宠物丢失问题。丢失宠物的主人可以及时发布丢失信息包括酬金，让更多的同城用户看到，提供线索。而那些捡到宠物的爱心人士，也可以将信息发布到平台上，为宠物寻找主人。

图 4-30　寻宠界面

用户可以在头部选择自己所在的地区，同时页面提供了三种过滤器：地区、宠类、寻找类型。用户可以通过点击过滤条件来获取到不同的信息。具体的信息中显示了标题、部分内容、酬金、以及发布时间等信息。页面右侧是发布信息和信息管理的按钮，可以跳转到相应的界面。当用户点击具体的信息时，会跳转到详情页面。详情页面使用表格的形式列出了基本信息。右侧是发布作者的头像和用户名显示，以及发布信息按钮和信息管理按钮。

图 4-31　寻宠详情界面

浏览用户还可以在页面底部进行留言。留言成功后会提示用户留言成功，并异步刷新数据。

图 4-32　寻宠留言界面

下面展示的是获取基础信息的部分代码：

```
getanimalSearchDetailsById() {
                this.userId = ly.store.getJson("user").id; //  获取页面中的参数
                this.id = ly.getUrlParam("id");
                let _this = this;
                ly.http.get("/search/getAnimalSearchDetailsById?id="          +
this.id)
                    .then(resp => {
                        if (resp.status === 200) {
                            _this.animalSearchVo = resp.data;
                        _this.srcList                                          =
_this.animalSearchVo.photoAnimalSearch;
                        ……
```

当页面数据渲染之前会执行该方法，从数据库中获取数据，存储到 Vue 的 data 属性范围中，然后将数据渲染到相应节点处。

用户还可以通过点击发布信息来跳转到信息发布界面，该界面包含基础的信息填写和图片上传等操作。上传成功或者失败之后，都会有相应的信息提示。

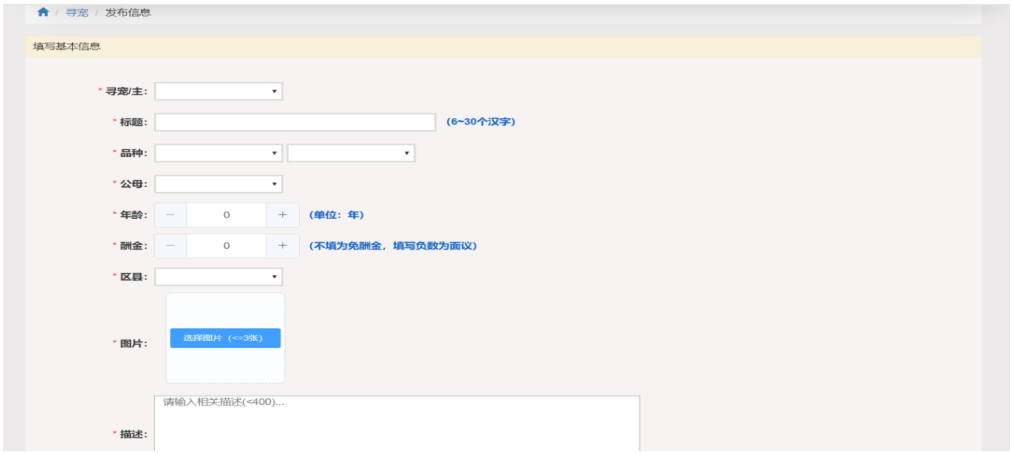

图 4-33　寻宠信息发布界面

用户还可以点击信息管理按钮来跳转到信息管理页面，该页面是对该用户发布的寻宠信息进行了分页展示，用户可以查看审核状态。

图 4-34　寻宠信息管理界面

该界面的信息展示部分是采用 HTML 中的“table”标签结合 Vue 中的“v-for”循环标签来对数据库中查询出来的集合信息进行循环展示。

4.2.7 帮助模块

该模块主要是针对用户如何发布信息、以及网站的一些规则等内容的展示。整体使用 Element UI 的手风琴组件来实现。

图 4-35　帮助信息界面

点击不同的标题就可以打开相应的内容进行显示。目前，此部分内容没有采用从数据库中读取的操作，而是直接写在 HTML 页面中。因为考虑到这些信息不需要经常修改或者更新，所以不需要从数据库中读取。

4.3 后台管理系统

考虑到系统拓展性和今后的服务化问题，我们将管理端抽取出来，重新构建一个后台管理系统。

后台管理系统与前台门户系统一样，依然采用 Vue 来进行开发。页面构建依然使用 HTML5、CSS3、JavaScript 构建，使用 axios 来与后台 API 来进行数据交互。

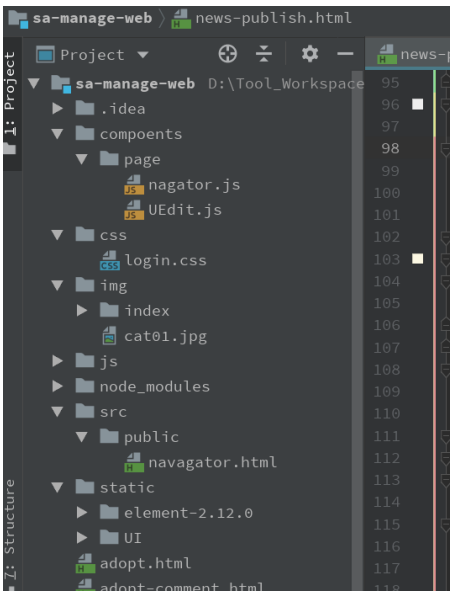

图 4-36　项目结构图

后台管理系统主要分为主页、用户、领养、寻宠、新闻通知、设置等板块，对项目的基础服务等都可以进行有效的管理和审核。

4.3.1 登录模块

登录界面设计的较为简单，只有管理员身份的用户才可以正常登录。角色信息存储在表中的“sa_admin_category_id”字段中。

图 4-37　首页图

4.3.2 首页简介

首页主要是展示一些快捷功能，如用户、领养、寻宠等，以及一些未审核的信息。页面是引用 ElementUI 来制作，搭配 HTML 的“a”标签就可以实现点击跳转到相应的功能部分。

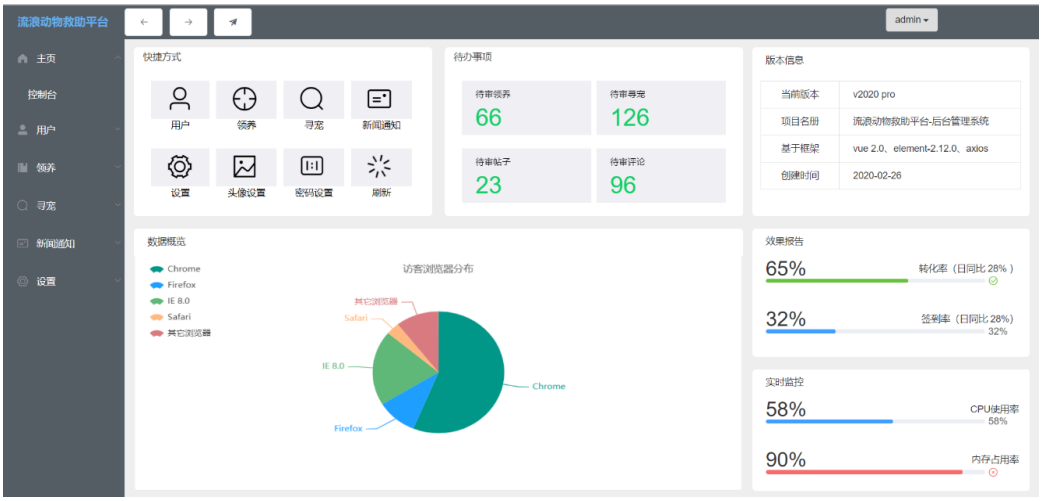

图 4-38　首页图

页面顶部的按钮可以选择收起和关闭导航栏，并且可以一键到达前端门户系统。这里我们采用的对布局的“div”的宽度来进行动态设置来达到效果，下面为该功能

实现函数：

```
open() {
    this.isCollapse = false;
  $("#left-div").css("width","13%");
  $("#right-div").css("width","87%");
},
close() {
  this.isCollapse = true;
  $("#left-div").css("width","3.2%");
  $("#right-div").css("width","96.8%");
},
```

4.3.3 用户模块

在用户管理方面，我们提供了非常充足的功能，如用户信息分页展示、修改用户信息、删除用户、恢复删除、角色设置等操作。这里，考虑到误删除操作，所以我们提供了恢复删除功能。考虑到日后的数据量问题，所以支持有多选操作。

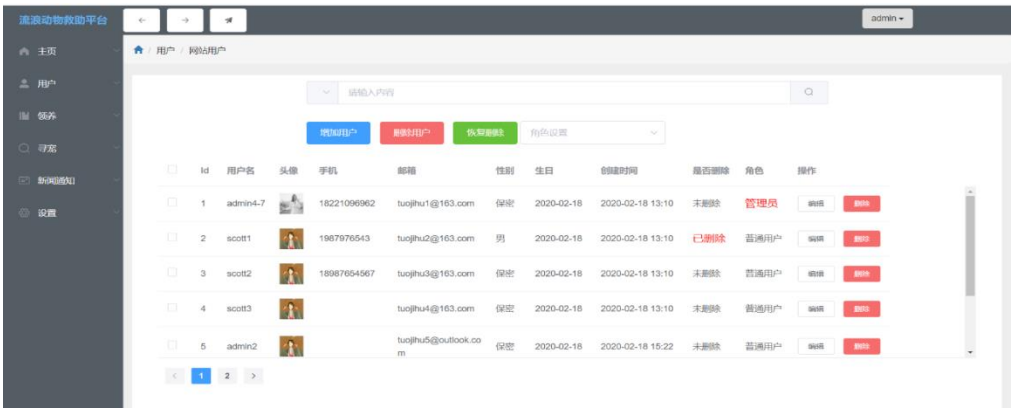

图 4-39　用户图

编辑功能展示：

图 4-40　编辑用户图

这里编辑窗口是采用 Bootstrap 的模态窗口来进行实现，点击编辑后根据点击的 Id 获取用户信息，将数据渲染到模态装中，管理员可以对其进行修改操作。

角色设置功能展示：

图 4-41　角色设置图

我们选中自己需要设置的用户，点击角色设置，就可以设置用户身份了。而这里的角色种类是从数据数据库中读取出来，将数据渲染到选择框中的。下面是角色获取部分的函数：

```
getAllRole() {
    let _this = this;
    ly.http.get("/user/queryAllRoles").then(resp => {
    if (resp.status === 200) {
    _this.roles = resp.data;
            }
       })
```

```
},
```

其中“ly”是我们定义的对象，“http”相当于调用 axios，“get”表示该次请求是一个 get 请求。而后台 API 中对应的该访问路径为“/user/queryAllRoles”。如果方法头部信息的状态码为 200，我们则将获取到的数据保存到 roles 数据中，然后在页面渲染的时候将数据渲染上去即可。

4.3.4 领养模块

领养部分的页面布局和用户的类似，主要包含信息管理和留言管理两部分。打开页面的时候会获取所有的领养信息，查询出来后进行分页显示。而我们也是提供了删除、恢复删除、新增、编辑、多选等操作。

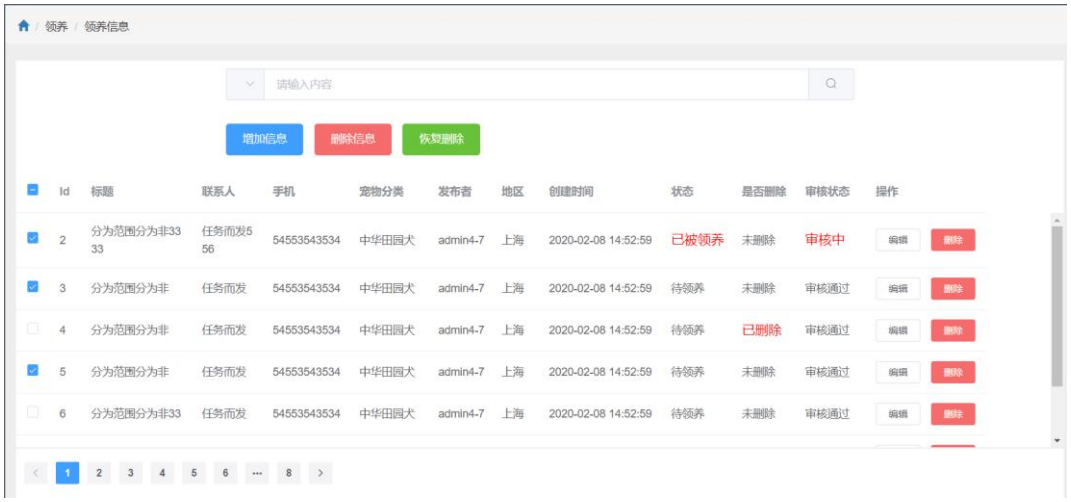

图 4-42　领养信息图

这里我们介绍编辑功能。点击编辑之后，可以修改基础信息以及对该信息的状态进行审核。因为默认用户发送的信息是审核中，只有审核通过的信息才会在前台进行展示。

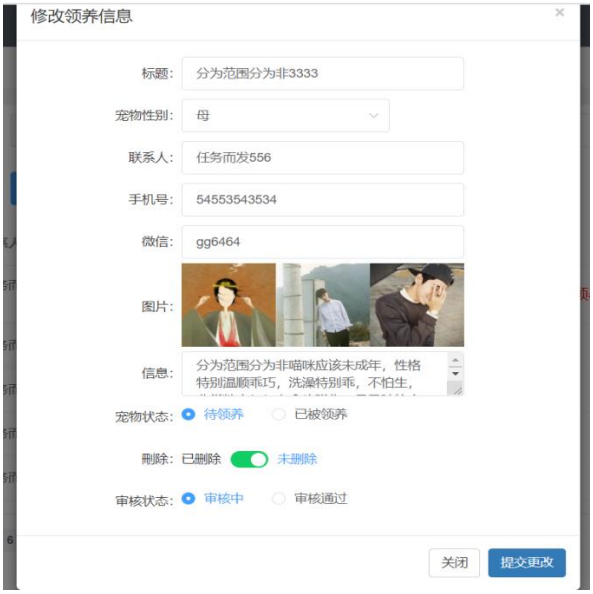

图 4-43　编辑领养信息图

这里的 UI 样式是引用的 Element UI 中的基础组件来构建，修改完成后点击提交修改就会触发点击事件绑定的函数，向后台的 API 发起请求。　留言部分的功能较为简单，可以分页列出所有留言，可以多选后进行删除和恢复删除操作。考虑到留言是用户的权利，这里不设置修改功能。如果留言内容涉及到问题，可以直接进行删除操作，这样前台就不会显示出对应的留言。

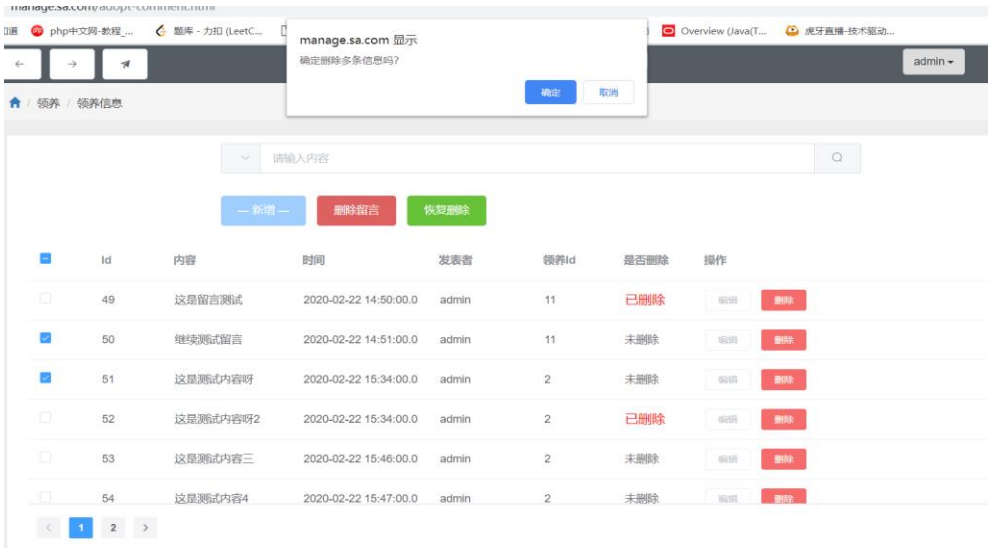

图 4-44　领养留言图

考虑到寻宠部分和领养部分内容较为相似，这里不再进行展示。

4.3.5 新闻通知模块

由于新闻通知是采用富文本框来输出内容到数据库中存储的（存储的内容是文本加 HTML 标签），所以要对其内容进行修改的话，较为复杂，需要进行正则过滤或者是调用富文本框的特定 API 来进行操作，所以我们这里也就没有设置对内容的修改。只提供了基础信息的修改和删除、恢复删除等操作。

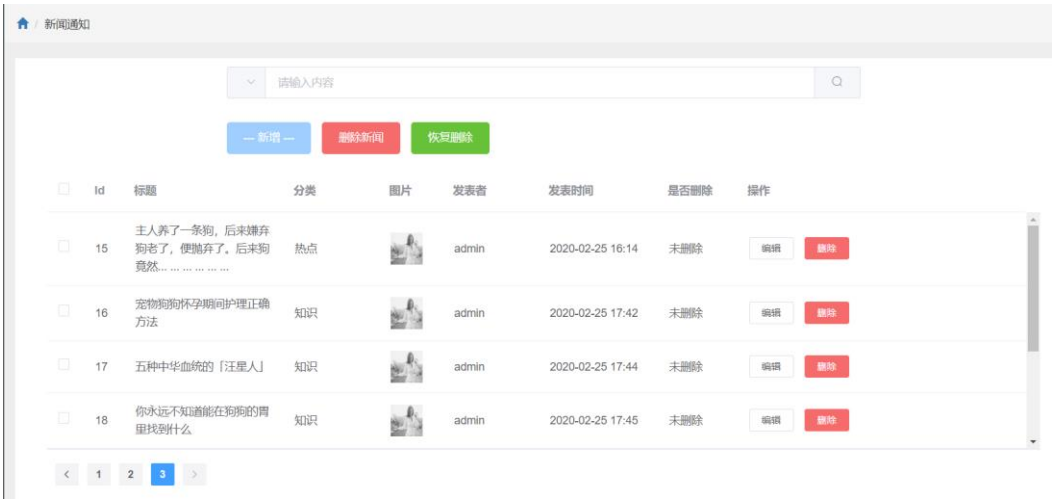

图 4-45　新闻通知图

新闻通知的另一部分包括发布新闻通知，这里我们也是在页面中集成了富文本框。发布的时候需要选择发布的主题和输入标题，然后在富文本框中输入内容。

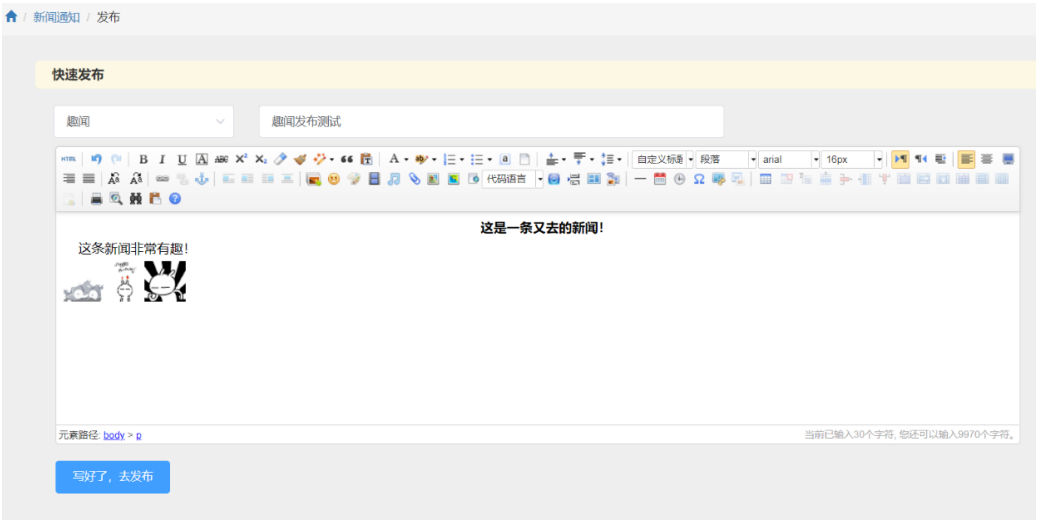

图 4-46　新闻通知发布图

新闻通知主题依然是通过查询数据库中的数据，将其渲染到选择框中，点击发布后会发布函数，与后台 API 交互。

4.3.6 设置模块

设置部分主要包括三部分，就是对管理员的基础资料修改、头像修改、密码修改。分为三部分是考虑到各部分的安全性和组件使用的简单性。

基础资料修改较为简单，使用 Element UI 的输入框组件展示信息即可。这里的生日选择部分，使用了 Element UI 的日期选择器，可以方便的选择日期，将其存储到数据中。

图 4-47　修改基本资料图

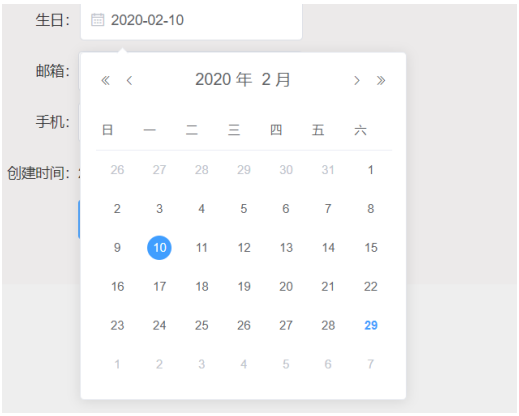

图 4-48　日期选择组件图

头像修改部分的左侧是初始头像，右侧是引用了 Element UI 的图片组件。

图 4-49　头像修改图

这里的头像上传，其实是调用后台的方法，将图片进行上传后将路径保存在数据中的“photo”字段中。下面展示的时候后台修改的部分方法：

```
@PostMapping("updatePhoto")
    public ResponseEntity<Void> updatePhotoById(@RequestParam("userId")Long userId, @RequestParam("file") MultipartFile file) {
        String contentType = file.getContentType(); //  图片文件类型
```

```
        String fileName =    file.getOriginalFilename();       // 图片名字
        System.out.println( "adopt-----"+ contentType + fileName + userId) ;
        UUID uuid=UUID.randomUUID();
        String                                newfileName                                =
uuid.toString()+fileName.substring(fileName.indexOf('.')); //文件重命名
        String    filePathInlocal    =    "D:\\Tool_Workspace\\IDEA\\sa-front\\"    +
"upload\\user\\";
        ……
 }
```

其中密码修改部分考虑到了安全性问题，所以在修改的时候需要正确的提供注册时使用的邮箱账号，通过之后才可以正常的修改密码。

第五章 结论

5.1 总结

该流浪动物救助平台的构建不仅解决了城市中数量庞大、分散性严重的流浪动物信息登记和管理问题，对于宠物寻找、宠物领养、流浪动物救助等各个方面都做出了解决。

通过该平台，全国各地的用户都可以将自己发现的流浪动物信息快速发布到平台，而爱心人士在查看到信息后也可以快速展开救援工作。针对于城市中经常发生的宠物丢失和弃养问题，我们也做了相应的模块功能进行解决。丢失宠物的主人可以将自己的宠物信息及时发布到平台上，同时也可以设置酬金，号召更多的人来提供线索，早日找到自己的爱宠。而那些因为工作或者是生活原因无法再继续照顾自己宠物的主人，也可以发布领养信息，为自己的爱宠找一个新家。而当越来越多的人通过该平台参与到流浪动物的信息发布和管理，以及宠物丢失、遗弃等问题的解决中来的时候，我相信城市中的流浪动物数量不仅会得到一个很好的控制，各种交通隐患、疾病传播的问题都会有显著的下降。

在设计实现该平台的过程中，遇到了一些问题，总结如下：

（1）关于使用 Page Helper 分页助手时遇到的问题：该平台前端分页是使用 Element UI 的分页组件，后端使用 Page Hepler 分页助手实现。因为我们在处理数据表多对多的时候是采用代码逻辑来实现控制，所以需要将数据库中多张表的信息查询出来后组装到 Vo 对象之后，再传输到前端进行显示。因为组装 Vo 的原因，导致 Page Hepler 在对主表进行截断的时候出现了数据总条数“total”等于分页显示条数“size”的问题。由于分页总数错误的原因，会导致前端无法正常分页。通过查看分页助手源码之后，发现发生问题的原因。由于我们返回的 List 集合是 Collection 接口的子类而不是 Page 的子类，所以会导致“pageNum=list.size”，这样查询出来的总数就是每页显示的集合的数量，即“size”的值

```
    if (list instanceof Page) {
        Page page = (Page)list;
        this.pageNum = page.getPageNum();
        this.pageSize = page.getPageSize();
        this.pages = page.getPages();
        this.list = page;
        this.size = page.size();
        this.total = page.getTotal();
        if (this.size == 0) {
            this.startRow = 0;
            this.endRow = 0;
        } else {
            this.startRow = page.getStartRow() + 1;
            this.endRow = this.startRow - 1 + this.size;
        }
    } else if (list instanceof Collection) {
        this.pageNum = 1;
        this.pageSize = list.size();
        this.pages = this.pageSize > 0 ? 1 : 0;
        this.list = list;
        this.size = list.size();
```

图 5-1 Page Helper 部分源码图

解决办法就是在构造方法之后重新手动设置。代码如下：

```
pageInfo.setPages(p.getPages());
pageInfo.setTotal(p.getTotal());
```

（2）在整合百度的富文本框 UEditor 的时候遇到问题：因为这款富文本框组件年代较早，所以是基于 JSP 页面来进行开发的。但是由于我们采用的是前后端分离的模式来进行，所以在整合上就需要进行改动。为此将其源码就行拆分，将设置部分内容保存在前端项目中，而配置资源以及处理类等保存在服务端代码中。这样就产生了一个问题，因为我们的服务端和前端运行在不同的容器中，前端开发的时候运行在“live-server”服务器中，而服务端则是运行在 Tomcat 服务器中，所以在读取配置文件的时候会产生跨域问题，导致集成失败。

开始的时候想通过处理普通跨域问题一样使用“CORS”的方法来处理。但是最终发现行不通，因为该次产生跨域问题的原因是“CORB”。最终通过查看 UEditor 源码发现，它是通过判断访问域名跨域之后通过 JSONP 的形式来进行请求访问的，最终将其修改为 AJAX 的形式后顺利解决。

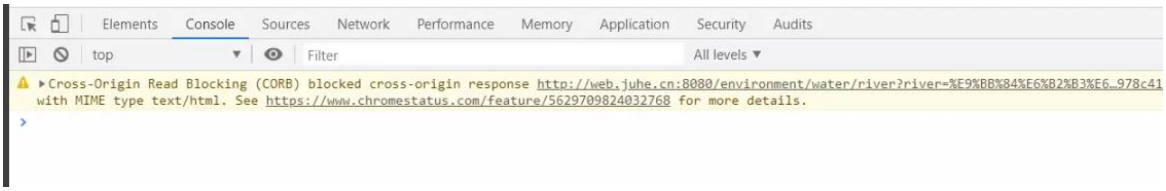

图 5-2　跨域问题错误图

（3）关于跨域问题的统一处理：因为我们的前台门户、后台管理、服务端都是运行不同的服务器或者端口下，所以每一次的请求访问都会产生跨域问题。我们统一采用 CORS 的方法来进行解决。因为 Spring MVC 提供了一个“corsFilter”过滤器，我们对其配置即可。下面是部分代码：

```
    @Bean
    public CorsFilter corsFilter() {// 初始化 cors 配置对象
            CorsConfiguration corsConfiguration = new CorsConfiguration();
            // 设置允许跨域的域名（也可以使用 setAllowedOrigins()方法，但是参数是 List 集合，比较麻烦）
        corsConfiguration.addAllowedOrigin("http://www.sa.com");
        corsConfiguration.addAllowedOrigin("http://manage.sa.com");
            // 允许携带 Cookie，此时上面允许跨域访问的域名就不能写成“*”，即表示所有域名都可以跨域
        ……
```

（4）关于使用 Spring“@Autowired”注解的问题：如果我们注入的时候有多个实现类的话，就需要按照名称注入，如果只使用了“@Autowired”注解，默认是按照类型注入的，所以此时就需要搭配“@Qualifier”来写上 Bean 的名称，或者是使用“@Resource”注解来实现，因为在不指定名称和类型的时候，Spring 会自动通过反射来获取名称，从而按照名称注入。

5.2 系统不足

从系统的整体架构方面来讲，流浪动物救助平台总体来说不算完美，因为考虑到开发的时间和工作量问题，并没有采用微服务的形式来完成。但是从整体的需求分析以及今后的运行来讲，最终是需要服务化的。因为该平台面对的用户对象是全国的每一个省份的居民，今后随着使用人数的增加，服务器的压力依然会越来越大。这时候如果不采取服务化，一旦系统的某一个模块出现问题，那么最终需要面临的问题是整个服务的崩溃。但是考虑到了这一点问题的存在，所以我们使用了前后端分离的模式，这为今后的微服务、多端化服务（多种客户端，例如：浏览器，车载终端，安卓，IOS 等）打下了坚实的基础。

从页面来讲，我们在样式布局、色彩搭配等方面对一些知名网站进行了借鉴，较为大方美观。但是从页面渲染的效率来讲，因为多处采用了 Vue 的差值表达式，也是产生了“差值表达式闪烁”的通病。所以，对于页面还需要做一些足够的优化工作才行。

从代码书写的角度上来讲，由于自己的开发经验不足，在一些逻辑判断上不是很明确，大量的使用“if”和“else”来进行判断，从而导致代码的简洁度不高。

从系统效率上来讲，虽然该平台引入了 Redis 非关系型数据库，但是对于一些经常需要查询的数据并没有将其放入到缓存中来，这样就浪费了 Redis 的高性能，有待优化。

5.3 系统展望

现在对于使用者较多的高并发项目，都是采用微服务的形式来进行开发的。如果将我们的平台服务化的话，首先会解决多个模块集中于一个服务的复杂性问题，这样每个模块都是单独分开的，如平台中的寻宠、领养等模块就可以拆分成不同的服务，更利于治理和维护。其次每个单独的服务都是可以独立部署的，这样今后为我们的平台持续化集成提供了条件。我们还可以按照需求的多少来部署不同数量的微服务，做到效益最高的负载均衡处理。今后如果有其他的需求，我们可以独立拓展一些服务而不影响整体的服务。

致　　谢

时光如白驹过隙，转瞬即逝间，大学四年的时光便是从指间悄然滑过。当初的局外人，不觉间已是局中人。感叹间拾起过往的回忆，脑海中浮现最多的便是教室中上课的情景和食堂间穿梭的人群，还有那图书馆外昏暗路灯下行色匆匆的路人。

思绪回到现在，经过了这么多天努力，我终于完成了自己的毕业设计工作。从开题报告开始到整个平台的最终实现，我通过运用学过的知识和学习新的技术最终完成了平台的构建。通过这一次项目的完成，我不仅巩固了自己之前学习过的基础知识和学习了新的技术，更是提高了自己动手能力和解决实际问题的能力。

在这里，我非常感谢我的毕业设计指导老师冯莉老师。从开始的开题报告、译文翻译到现在的论文，冯莉老师给了我许多的指导和修改意见，让我对整个过程有了一个非常明确的方向，最终能成功的完成平台开发和论文。再次，我再次向她表示我最真挚的感谢。其次，我还要感谢大学四年的其他任课老师，在大家的辛苦教授和悉心指导下，我不仅学到了扎实的数学知识和计算机知识，同时也提高了自己的实际动手能力和学习能力。正是因为这些培养了我的思维和解决问题的能力，我才能快速的学习一些技术，并投入到实际的使用中去，顺利的完成我的毕设。

同时我还要感谢所有在我毕设期间对我进行指导的同学和一些好友。在一些不明白的前端代码和后台逻辑上，对我提供了优秀的解决思路和参考文档，帮助我解决了遇到的问题。正是因为有了你们的帮助和指导，我才能成功的完成我的毕设。在此，我再次向他们表示最真挚的感谢！

参考文献

[1] 韦佳佳，任海鹏，孙宇. Java EE 在轻量级智慧校园架构设计中的应用[J]. 太原学院学报(自然科学版)，2018，36（01）：50-53.

[2] 杨静. 基于 JAVA WEB 中 MVC 模式的研究与应用[J]. 电脑知识与技术，2014，9（28）：68-71.

[3] 李兴华，马云涛. 第一行代码 Java 视频讲解版[M]. 北京：人民邮电出版社，2017：590-593.

[4] Bruce Eckel. Thinking in Java [M]. USA：Prentice Hall，2015：367-869.

[5] 钱新杰，胡桂香. 基于 Java EE 的 Web 系统中数据库设计技术研究[J]. 信息与电脑(理论版)，2015（04）：75-76.

[6] Licai. The English Teaching System Design and Implementation Based on J2EE Campus Network[C]//International Conference on Education Technology，Management and Humanities Science(ETMHS 2015)：Atlantis Press，2015：1323-1326.

[7] Sourabh Sharma. Mastering Microservices with Java[M]. USA:Packt Publishing，2016：2-155.

[8] 吴霁轩. MySQL 数据库后台优化方案[J]. 科技创新与应用，2016（22）：107.

[9] 苟文博，于强. 基于 MySQL 的数据管理系统设计与实现[J]. 电子设计工程，2017（06）：62-65.

[10] 唐权. SSM 框架在 Java EE 教学中的应用与实践[J]. 福建电脑，2017，33（12）：93-94+61.

[11] 乔岚. 基于 MyBatis 和 Spring 的 Java EE 数据持久层的研究与应用[J]. 信息与电脑(理论版)，2017（08）：73-76.

[12] 匡成宝. HTML 语言的网页制作方法与技巧探讨[J]. 电脑迷，2017（03）：190-191.