



綿陽師範學院
MIANYANG NORMAL UNIVERSITY

(2020) 届本科生毕业论文

题 目 流浪猫犬救助领养平台建设

专 业 计算机科学与技术

院 部 信息工程学院

学 号 1618130234

姓 名 卿三思

指 导 教 师 范敏 副教授

答 辩 时 间 二 0 二 0 年 五 月

工作时间：2019 年 11 月 至 2020 年 5 月

流浪猫犬救助领养平台建设

学 生：卿三思

指导教师：范 敏

摘 要：随着越来越多的人开始饲养宠物，宠物数量不断增加，因各种原因遭到遗弃的宠物数量也在逐年增长。被遗弃的宠物多以猫犬为主，它们散布、游走在城市街道社区、小区花园等区域，其自身福利得不到保障的同时，对城市环境和公共卫生也都构成了一定威胁。但随着人民综合素质的不断提高，流浪猫犬的生存问题提高到了道德伦理的层面上，由此催生出许多以救助流浪猫犬为目的的组织。但是在这信息化的时代，救助组织的工作和宣传都缺少信息化手段的辅助。因此，设计该救助领养平台的初衷即是为救助流浪猫犬的组织机构提供一个功能完善的网站平台，既提高了救助的工作效率，也为救助组织的发展提供了动力。

本系统基于时下流行的 SSM(Spring+SpringMVC+MyBatis)框架设计而成，采用了 Maven 项目管理工具，使用了 Shiro 安全框架，运用了 JSP、Bootstrap、Ajax、JQuery 等前端技术进行开发。系统主要由用户端和管理端两部分组成，用户端面向普通用户和爱心人士，提供了流浪宠物领养、流浪宠物救助、同城领养等功能，用户可以领养到心仪的宠物，爱心人士可以为救助流浪猫犬出一份力。管理端面向流浪猫犬救助组织，具有完善的用户认证和授权功能，保证了系统的安全性，提供了用户管理、物资管理、宠物管理等管理功能，提高了救助组织的工作效率，为更好地开展救助活动奠定了基础。

关键词：流浪宠物；救助领养；SSM；Bootstrap；Maven；Shiro

Construction of Rescue and Adoption Platform for Stray Cats and Dogs

Undergraduate: Qing Sansi

Supervisor: Fan Min

Abstract: As more and more people begin to keep pets, the number of pets is increasing, and the number of pets abandoned for various reasons is also increasing year by year. Most of the abandoned pets are mainly cats and dogs. They are scattered and wandering in urban street communities, community gardens and other areas, their own welfare is not guaranteed, but also pose a certain threat to the urban environment and public health. But with the continuous improvement of the people's comprehensive quality, the survival of stray cats and dogs has been raised to the level of moral ethics, which has given birth to many organizations for the purpose of rescuing stray cats and dogs. But in this information age, the work and propaganda of relief organizations lack the assistance of information means. Therefore, the original purpose of this platform is to provide a well-functioning website platform for the organization of helping stray cats and dogs, which not only improves the efficiency of rescue work, but also provides the impetus for the development of rescue organizations.

The system is based on the popular SSM(Spring+SpringMVC+MyBatis) framework, using Maven project management tools, using Shiro security framework, using front-end technologies such as JSP、Bootstrap、Ajax、JQuery. The system is mainly composed of two parts: the client and the management, the client is aimed at ordinary users and caring people, stray pet adoption, stray pet rescue, city adoption and other functions, users can adopt favorite pets, caring people can contribute to the rescue of stray cats and dogs. The management faces the stray cat and dog rescue organization, has a perfect user authentication and authorization function, guarantees the system's security, provides the user management, the material management, the pet management and other management functions, enhances the rescue organization's work efficiency, has laid the foundation for the better development rescue activity.

Key words: Stray Pets; Rescue and Adoption; SSM; Bootstrap; Maven; Shiro

目 录

1	系统需求	1
1.1	需求分析	1
1.2	功能分析	1
1.2.1	用户端功能	1
1.2.2	管理端功能	2
1.3	可行性分析	2
1.3.1	技术可行性	2
1.3.2	经济可行性	2
1.3.3	操作可行性	3
1.4	运行环境	3
2	系统设计	4
2.1	功能模块	4
2.1.1	用户端功能设计	4
2.1.2	管理端功能设计	5
2.2	数据库设计	6
2.2.1	逻辑结构设计	6
2.2.2	物理结构设计	6
2.3	人机界面设计	14
3	系统功能实现	15
3.1	通用功能模块	15
3.1.1	全局异常处理	15
3.1.2	自定义返回值类型	15
3.2	用户端	15
3.2.1	注册登录	15
3.2.2	领养宠物	18
3.2.3	同城领养	19
3.2.4	拯救毛孩子	23
3.2.5	养宠小知识	25
3.2.6	我的文章	26
3.3	管理端	27
3.3.1	登录	27
3.3.2	系统管理	30
3.3.3	站点管理	34
3.3.4	用户管理	38
3.3.5	宠物管理	40

3.3.6	文章管理	43
4	系统测试	47
4.1	测试的目的及意义	47
4.2	测试环境	47
4.3	测试用例设计	47
5	总结与展望	52
5.1	总结	52
5.2	展望	52
	参考文献	53
	致 谢	54

1 系统需求

1.1 需求分析

自改革开放以来,人民的经济条件、生活水平不断地提升,饲养宠物成为了不少人青睐的生活方式。随着宠物市场诞生后,人们饲养宠物观念的逐渐普及,据统计,2018 年全国宠物数量高达 2.5 亿只。有些人饲养宠物仅仅是一时兴起,没有对宠物负责一生的决心,或者因为其他各种各样的原因将宠物遗弃^[1]。在如此大的基数下,流浪宠物的数量也在日渐增加。由于流浪宠物居无定所,生活环境恶劣,自身会携带大量的细菌、病毒和寄生虫,它们游荡在大街小巷,对人们的生活带来了极大的潜在危害^[2]。

我国仅出台有《野生动物保护法》,对于宠物从出生饲养到其生命结束过程中产生的一系列问题以及流浪动物的一系列问题均无法可依^[3]。有些人为了利益对宠物施行了暴力行为,由此引起了极其恶劣的影响^[4];有些人则采取合乎道德情理的方式,成立流浪宠物救助组织救助流浪宠物,然后寻找新的主人领养,这在一定程度上解决了流浪宠物的问题^[5]。但这也带来了许多新的问题,这类组织的运转是需要广大爱心人士的支持和帮助的^[6]。如果没有有力的宣传途径就无法受到关注,也就得不到人们的支持,救助活动就只能宣告停止^[7]。但若受到人们高度关注后,爱心人士就可能捐款捐物或者志愿参与救助活动等,流浪宠物救助组织将面临庞大的业务量,这时如果没有高效并且准确可靠的信息处理方式,将可能造成不可挽回的过失。事实上,流浪宠物主要是流浪猫和流浪狗,在互联网如此发达的时代,以我们熟知的传统流浪猫狗救助站的形式显然是行不通的,因此,设计一款建立在互联网基础上的流浪猫犬救助领养平台是合乎现实需要的。该平台以网站的形式呈现,为方便救助组织在全国多地设立线下站点以提供救助领养服务给出了高效的信息解决方案,不仅能提高工作效率,便于宣传,而且能有效吸引广大爱心人士的参与,也可以为他们提供较高的用户参与度。

1.2 功能分析

本网站系统提供的功能主要分为用户端和管理端,分别为广大普通用户和救助组织提供了相应的功能,用户端和管理端相互配合以完成流浪宠物的救助与领养工作。

1.2.1 用户端功能

用户注册并登录平台,可以浏览选择心怡的宠物并按要求进行领养流程,亦可成为志愿者参与到流浪宠物的救助活动中来。

(1) 宠物领养:查看各站点发布的待领养宠物信息,根据意愿联系站点进行领养手续的办理。

(2) 同城领养:定位用户当前所在城市,并能查看该城市内其他用户发布的同城领养的宠物信息,用户也可以发布自己的同城领养信息,用户间自行进行宠物领养流程。

(3) 拯救毛孩子：若用户当前所在城市没有设立平台的线下站点，该功能将不开放。用户需注册为志愿者方可使用该功能，注册后志愿者可以发布当前所在城市内流浪宠物需要救助的信息，并配合该城市内的线下站点的员工进行救助，救助后将宠物暂养于站点内，等待领养，志愿者还可监督该城市内其他救助信息的进展情况。

(4) 养宠小知识：查看平台发布的与养宠相关的文章。

(5) 我的文章：管理自己发布的同城领养或拯救毛孩子的文章，编辑文章、更改文章状态或删除。

1.2.2 管理端功能

管理端根据管理员的角色不同分配不同的权限，拥有不同权限的管理员各司其职，任何管理员无法越权操作，保证信息的安全性。

- (1) 系统管理：管理所有管理员账号，分配角色；管理角色，分配系统权限；管理菜单、权限。
- (2) 站点管理：管理各个站点信息，各站点管理其站点物资信息，物资申请、入库、补仓、物资记录等。
- (3) 用户管理：管理普通用户和志愿者信息。
- (4) 宠物管理：各站点管理其站点收养的宠物信息，记录领养人信息以及领养协议等，对领养人进行回访时记录回访信息。
- (5) 文章管理：对向客户端展示的待领养信息文章、同城领养文章、养宠小知识文章、救助文章进行管理。

1.3 可行性分析

经对市场上同类产品的调查，以及对已有流浪猫狗救助领养组织工作流程的调查、研究、分析，本系统的可行性已具备，以下从技术、经济、操作三方面进一步分析。

1.3.1 技术可行性

本系统属于 Java Web 开发，通过 Maven 构建并管理项目^[8]，前端采用基于 Bootstrap3 开发的完全响应式扁平化主题框架^[9]，使用了 Html5、CSS3、JavaScript、jQuery(v3.1.1)等技术。后端采用 Spring 框架对底层类的实例化和生命周期进行管理^[10]，通过其提供的 AOP 编程^[11]方便的实现对程序进行权限拦截，通过其声明式事务^[12]功能方便的完成事务的管理，以及采用其 SpringMVC 模块^[13]设计开发整个项目。项目还使用 Mybatis 进行数据库操作^[14]，使用 Shiro 安全框架进行权限控制^[15]，使用 MySQL 数据库储存数据。各项技术已具备成熟的开发条件，因此本系统在技术上是可行的。

1.3.2 经济可行性

本系统的开发工具采用 Eclipse，数据库采用 MySQL，均免费，因此在开发过程中没有开发工具成本，只有人力成本。系统部署运行至少需要租赁一台配置为 1 核 CPU、2GB 内存、50GB 硬盘、2Mbps 带宽的服务器。以一般服务器为例，租金约为 900 元/年，注册 .com 域名成本约为 50 元/年，域名解析服务成本约为

1288 元/年，因此系统部署运行时除人力成本外，成本约为 2238 元/年。系统在维护阶段也只需要少量人力成本。如果有其他需求，基于该系统进行二次开发或者功能增强较为容易，可以节省很多时间和成本，故本系统在经济上是可行的。

1.3.3 操作可行性

本系统用户端页面美观、简洁、操作简单，且页面适应多种设备，用户仅需有基本的手机、PC 等操作能力即可。对于管理端，使用人员除了需要具备手机、PC 等基本操作常识以外还需要熟悉系统业务的工作流程，这仅需要短暂的培训即可。故本系统在操作上是可行的。

1.4 运行环境

操作系统：Windows7 及以上。

数据库：MySQL5.7.26 及以上。

Web 服务器：Tomcat 8.5

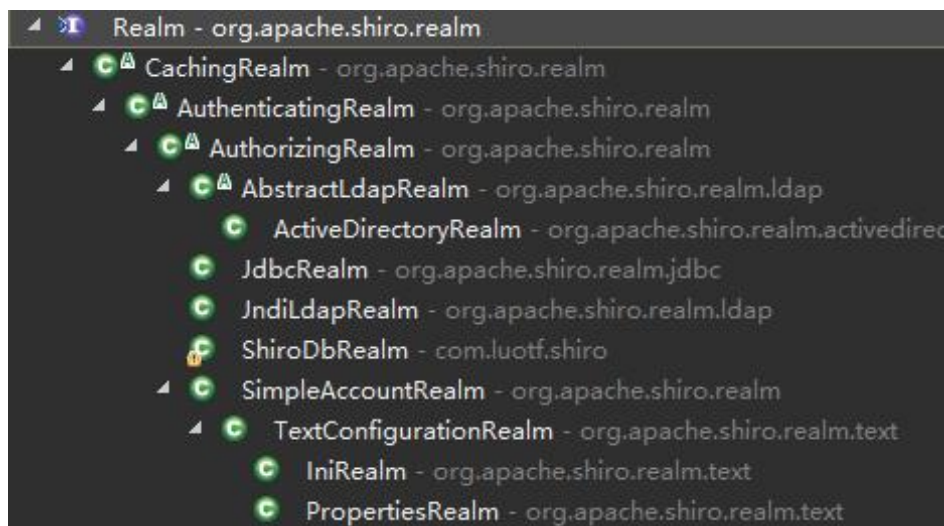
浏览器：IE10 以上。

2 系统设计

2.1 功能模块

本系统基于 Shiro 安全框架的身份认证流程，创建自定义的 Realm 类，继承 AuthorizingRealm 类，实现 doGetAuthenticationInfo 方法，由此来实现登录功能，未通过身份认证的用户只能访问允许匿名访问的页面。

Realm 的继承关系如图 2-1 所示。



系统采用多 Realm 认证流程，用户端和管理端各使用一个自定义 Realm，对应不同的认证规则，根据 Shiro 多 Realm 认证规则，用户登录需通过授权处理类 ModularRealmAuthorizer 的验证策略才能认证成功，本系统采用的验证策略是 AtLeastOneSuccessfulStrategy，即通过任意 Realm 认证就算认证成功。

认证成功后进行 Realm 的授权流程，授权功能是通过实现 AuthorizingRealm 类的 doGetAuthorizationInfo 方法来完成的。管理员登录时通过该方法查询数据库中该管理员所拥有的角色、权限来赋予管理员应有权限，也阻止了管理员越权操作。用户端的 Realm 没有进行授权流程，因此用户无法访问管理端。

本系统的大部分功能方法均基于上述授权规则而写，是本系统架构的根本之一，也是系统最基本的功能设计之一，使其具备了企业级的安全等级。接下来分别阐述用户端和管理端的主要功能设计。

2.1.1 用户端功能设计

用户端分为注册登录、宠物领养、同城领养、拯救毛孩子、养宠小知识、我的文章六大功能模块，具体如图 2-2 所示。

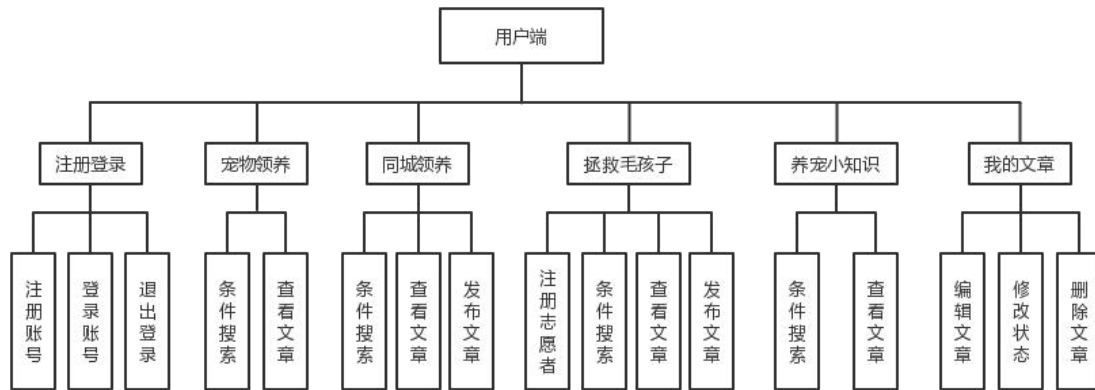


图 2-2 用户端功能结构图

对于用户端来说，用户主要承担着平台宣传和参与到领养救助环节的作用，这也是整个平台运营不可或缺的一环。用户通过简单的注册即可使用除救助功能即拯救毛孩子以外的所有功能，如果用户有意愿参与到流浪猫犬的救助活动中，只要其所在城市有平台的线下站点时就能注册为志愿者，使用平台的救助功能帮助该城市的站点救助流浪猫犬。

由于同城领养和救助功能都是针对用户所在城市来显示相关信息的，因此系统采用高德地图 API 进行精确城市定位。

2.1.2 管理端功能设计

管理端分为登录、系统管理、站点管理、用户管理、宠物管理、文章管理六大功能模块，具体如图 2-3 所示。

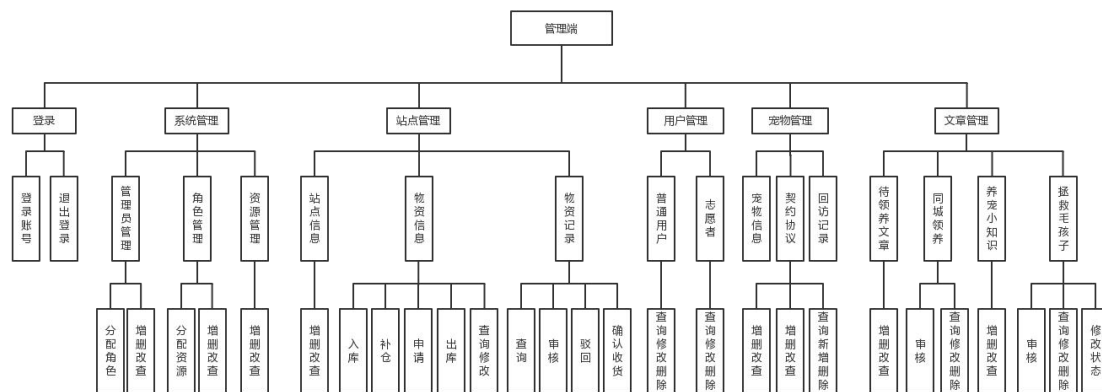


图 2-3 管理端功能结构图

管理端暂有超级管理员、管理员、站点管理员三种角色，同一管理员可兼具多种角色，每种角色拥有不同的资源权限。仅超级管理员拥有系统管理权限，对所有管理员、角色、资源进行分配；普通管理员拥有除系统管理外的所有权限；站点管理员拥有管理其所属站点的志愿者、物资信息、物资记录、宠物信息、契约协议、回访记录、待领养文章、拯救毛孩子文章的权限，且仅能对其所属站点的这些信息进行各种操作，无法越权操作其它站点的信息。

通过分配系统的各项权限，使不同的角色承担系统的各项业务功能，例如超级管理员只需对整个系统进行监督和分配，普通管理员负责大部分审核工作，站点管理员则主要进行救助、领养等具体业务工作。这使得系统的业务流程更加清

表 2-1 管理员表 (续)

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
mobile	手机号	varchar	100	否	是
dept_id	所属部门 id	int	11	否	是
status	状态	int	11	否	否
create_time	创建时间	datetime	0	否	否
update_time	修改时间	datetime	0	否	否

(2) 角色表

该表存放角色信息, 包含 id、名称、备注、创建时间、修改时间字段, 如表 2-2 所示。

表 2-2 角色表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
role_id	id	int	11	是	否
name	名称	varchar	100	否	否
remark	备注	varchar	100	否	是
create_time	创建时间	datetime	0	否	否
update_time	修改时间	datetime	0	否	否

(3) 资源表

该表存放角色信息, 包含 id、父资源 id、名称、路径、url、权限、类型、图标、排序、创建时间、修改时间字段, 如表 2-3 所示。

表 2-3 资源表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
resource_id	id	int	11	是	否
parent_id	父资源 id	int	11	否	否
name	名称	varchar	50	否	否
path	路径	varchar	255	否	是
url	url	varchar	200	否	是
permission	权限	varchar	500	否	是
type	类型	int	11	否	否
icon	图标	varchar	50	否	是
order_num	排序	int	11	否	是
create_time	创建时间	datetime	0	否	否
update_time	修改时间	datetime	0	否	否

(4) 管理员角色对应表

该表存放管理员表与角色表多对多对应关系信息, 包含 id、管理员 id、角色 id 字段, 如表 2-4 所示。

表 2-4 管理员角色对应表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
id	id	int	20	是	否
user_id	管理员 id	int	11	否	否
role_id	角色 id	int	11	否	否

(5) 角色资源对应表

该表存放角色表与资源表多对多对应关系信息, 包含 id、角色 id、资源 id 字段, 如表 2-5 所示。

表 2-5 角色资源对应表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
id	id	int	11	是	否
role_id	角色 id	int	11	否	否
resource_id	资源 id	int	11	否	否

(6) 用户表

该表存放用户信息, 包含 id、用户名、密码、私盐(盐值加密^[16])、邮箱、手机号、是否注册志愿者字段, 如表 2-6 所示。

表 2-6 用户表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
user_id	id	int	11	是	否
username	用户名	varchar	50	否	否
password	密码	varchar	100	否	否
salt	私盐	varchar	32	否	否
email	邮箱	varchar	100	否	是
tel	手机号	varchar	100	否	是
isvol	是否是志愿者	int	1	否	否

(7) 志愿者表

该表存放志愿者信息, 包含 id、用户 id、注册站点 id、真实姓名、性别、手机号、身份证号、省、市、区字段, 如表 2-7 所示。

表 2-7 志愿者表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
vol_id	id	int	11	是	否
user_id	用户 id	int	11	否	否
dept_id	注册站点 id	int	11	否	否
realname	真实姓名	varchar	30	否	否
gender	性别	int	1	否	否
tel	手机号	varchar	50	否	否

表 2-7 志愿者表 (续)

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
idcard	身份证号	varchar	50	否	否
province	志愿地区(省)	int	20	否	否
city	志愿地区(市)	int	20	否	否
district	志愿地区(区)	int	20	否	否

(8) 线下站点表

该表存放线下站点信息, 包含 id、站点名称、负责人、负责人电话、类型、省、市、区、详细地址、备注字段, 如表 2-8 所示。

表 2-8 线下站点表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
dept_id	id	int	11	是	否
dept_name	站点名称	varchar	50	否	否
dept_manager	负责人	varchar	50	否	否
manager_tel	负责人电话	varchar	20	否	否
type	类型	int	1	否	否
parea_id	所在地(省)	int	20	否	否
carea_id	所在地(市)	int	20	否	否
darea_id	所在地(区)	int	20	否	否
dept_add	详细地址	varchar	100	否	否
remark	备注	varchar	100	否	是

(9) 物资表

该表存放物资信息, 包含 id、父 id、物资名称、数量、所属站点 id、单价、状态、备注、创建时间、修改时间字段, 如表 2-9 所示。

表 2-9 物资表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
goods_id	id	int	20	是	否
parent_id	父 id	int	20	否	否
name	物资名称	varchar	100	否	否
count	数量	varchar	100	否	否
dept_id	所属站点 id	int	11	否	否
create_time	创建时间	datetime	0	否	否
update_time	修改时间	datetime	0	否	否
price	单价	decimal	10	否	否
status	状态	int	1	否	否
remark	备注	varchar	100	否	是

(10)物资记录表

该表存放物资记录信息，包含 id、物资 id、所属站点 id、物资变化数量、原物资数量、状态、申请物资时间、物资变化时间、确认收货时间字段，如表 2-10 所示。

表 2-10 物资记录表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
id	id	int	11	是	否
goods_id	物资 id	int	11	否	否
dept_id	所属站点 id	int	11	否	否
count	物资变化数量	int	20	否	否
restcount	原物资数量	int	20	否	否
status	状态	int	1	否	否
request_time	申请物资时间	datetime	0	否	是
update_time	物资变化时间	datetime	0	否	是
confirm_time	确认收货时间	datetime	0	否	是

(11)宠物表

该表存放宠物信息，包含 id、宠物类型、品种、宠物名字、性别、年龄、生日、照片、收养站点、状态、备注、创建时间字段，如表 2-11 所示。

表 2-11 宠物表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
pet_id	id	int	11	是	否
type	宠物类型	int	1	否	否
variety	品种	varchar	50	否	否
petname	宠物名字	varchar	30	否	否
gender	性别	int	1	否	是
age	年龄	varchar	30	否	是
birthday	生日	varchar	30	否	是
dept_id	收养站点	int	11	否	否
status	状态	int	1	否	否
remark	备注	varchar	50	否	是
create_time	创建时间	datetime	0	否	否
photo	照片	varchar	255	否	是

(12)契约协议表

该表存放领养宠物的契约和协议信息，包含 id、领养站点 id、领养人名、领养人年龄、领养人职业、领养人电话、领养人住址、领养人身份证号、领养宠物 id、协议图片、契约图片、创建时间字段，如表 2-12 所示。

表 2-12 契约协议表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
contract_id	id	int	11	是	否
dept_id	领养站点 id	int	11	否	否
master	领养人名	varchar	30	否	否
age	领养人年龄	int	11	否	否
career	领养人职业	varchar	50	否	否
tel	领养人电话	varchar	50	否	否
addr	领养人住址	varchar	50	否	否
idcard	领养人身份证	varchar	50	否	否
pet_id	领养宠物 id	int	11	否	否
agreement	协议图片	varchar	255	否	是
contract	契约图片	varchar	255	否	是
create_time	创建时间	datetime	0	否	否

(13) 回访记录表

该表存放回访领养人记录信息, 包含 id、契约协议 id、总计回访次数、信息是否改变、回访是否合格、所属站点 id、回访时间字段, 如表 2-13 所示。

表 2-13 回访记录表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
revisit_id	id	int	11	是	否
contract_id	契约协议 id	int	11	否	否
count	总计回访次数	int	11	否	否
ischange	信息是否改变	int	1	否	否
pass	回访是否合格	int	1	否	否
dept_id	所属站点 id	int	11	否	否
create_time	回访时间	datetime	0	否	否

(14) 待领养宠物文章表

该表存放发布的待领养宠物文章信息, 包含 id、宠物 id、发布站点 id、标题、简介、标签、内容、发布时间、修改时间字段, 如表 2-14 所示。

表 2-14 待领养宠物文章表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
adopt_id	id	int	11	是	否
pet_id	宠物 id	int	11	否	否
dept_id	发布站点 id	int	11	否	否
title	标题	varchar	255	否	否
introduction	简介	varchar	255	否	是

表 2-14 待领养宠物文章表 (续)

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
tag	标签	varchar	255	否	是
content	内容	longtext	0	否	是
create_time	发布时间	datetime	0	否	否
update_time	修改时间	datetime	0	否	否

(15)同城领养文章表

该表存放发布的同城领养文章信息, 包含 id、发布用户 id、省、市、宠物名字、品种、性别、年龄、生日、照片、标题、简介、标签、内容、状态、审核、发布时间、修改时间字段, 如表 2-15 所示。

表 2-15 同城领养文章表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
cityadopt_id	id	int	11	是	否
user_id	发布用户 id	int	11	否	否
province	所在地(省)	int	20	否	否
city	所在地(市)	int	20	否	否
petname	宠物名字	varchar	30	否	否
variety	品种	varchar	50	否	否
gendar	性别	int	1	否	否
age	年龄	varchar	30	否	是
birthday	生日	varchar	30	否	是
photo	照片	varchar	255	否	是
title	标题	varchar	255	否	否
introduction	简介	varchar	255	否	是
tag	标签	varchar	255	否	是
content	内容	longtext	0	否	是
status	状态	int	1	否	否
verify	审核	int	1	否	否
create_time	发布时间	datetime	0	否	否
update_time	修改时间	datetime	0	否	否

(16)养宠小知识文章表

该表存放养宠小知识文章信息, 包含 id、标题、简介、标签、内容、发布时间、修改时间字段, 如表 2-16 所示。

表 2-16 养宠小知识文章表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
tips_id	id	int	11	是	否
title	标题	varchar	255	否	否
introduction	简介	varchar	255	否	是
tag	标签	varchar	255	否	是
content	内容	longtext	0	否	否
create_time	发布时间	datetime	0	否	否
update_time	修改时间	datetime	0	否	否

(17)救助文章表

该表存放救助文章信息，包含 id、发布用户 id、标题、简介、标签、内容、省、市、状态、审核、发布时间、修改时间字段，如表 2-17 所示。

表 2-17 救助文章表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
rescue_id	id	int	11	是	否
user_id	发布用户 id	int	11	否	否
title	标题	varchar	255	否	否
introduction	简介	varchar	255	否	是
tag	标签	varchar	255	否	是
content	内容	longtext	0	否	否
province	所在地（省）	int	20	否	否
city	所在地（市）	int	20	否	否
status	状态	int	1	否	否
verify	审核	int	1	否	否
create_time	发布时间	datetime	0	否	否
update_time	修改时间	datetime	0	否	否

(18)全国省市区县表

该表存放全国所有省、市、区/县信息，共计 30435 条记录，包含 id、名称、父 id、类型字段，如表 2-18 所示。

表 2-18 全国省市区县表

字段名称	字段意义	字段类型	字段长度	是否主键	能否为空
id	id	int	255	是	否
name	名称	varchar	255	否	否
parent_id	父 id	int	11	否	否
type	类型	int	1	否	否

2.3 人机界面设计

本系统用户端与管理端均在左侧列表显示各个功能菜单，大部分菜单为两级菜单，可通过点击下拉查看二级菜单。

用户端界面功能菜单有：主页、领养宠物、同城领养、养宠小知识、拯救毛孩子、我的文章（同城领养、拯救毛孩子），用户端人机界面视图如图 2-5 所示。

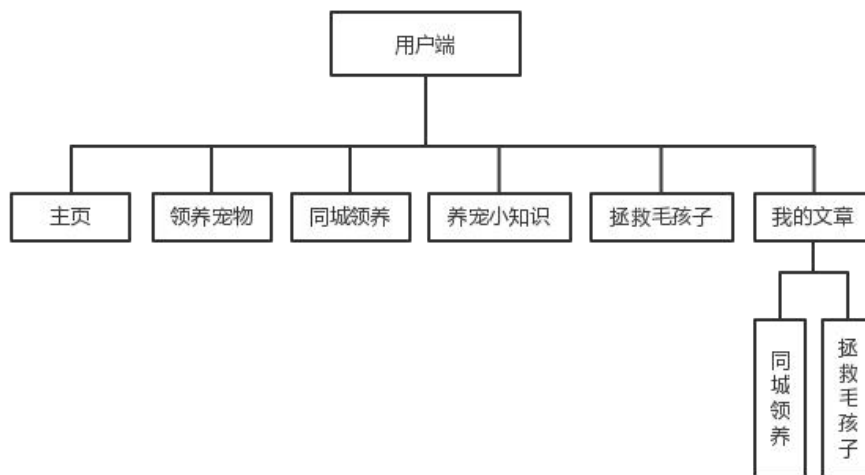


图 2-5 用户端人机界面视图

管理端界面功能菜单有：主页、系统管理（管理员、角色管理、资源管理）、站点管理（站点信息、物资信息、物资记录）、用户管理（普通用户、志愿者）、宠物管理（宠物信息、契约协议、回访记录）、文章管理（待领养文章、同城领养、养宠小知识、拯救毛孩子），管理端人机界面视图如图 2-6 所示。

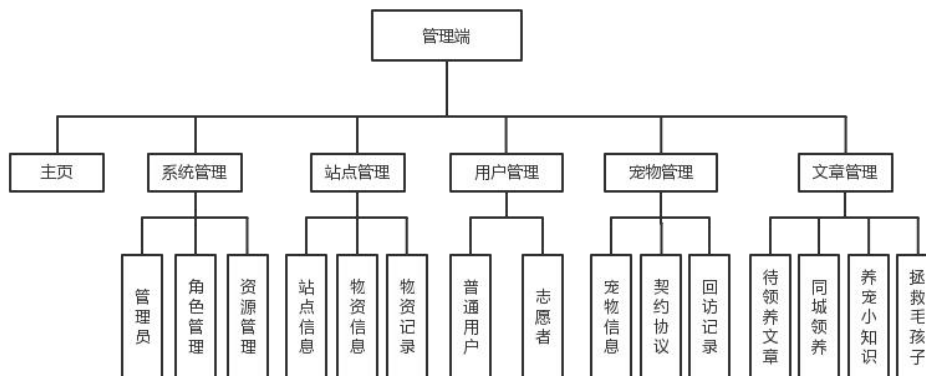


图 2-6 管理端人机界面视图

3 系统功能实现

前端页面以 JSP 页面为主, 以 Bootstrap3 为基础做页面的布局, 同时使用了许多前端插件, 如使用 Bootstrap-table、Treeview、Validate、Summernote 等来简化前端与后端的交互, 又使用 Chosen、Datepicker、Fontawesome、Prettyfile、Layer 等来美化前端页面。大部分前端与后端的交互通过 Ajax 实现^[17], 数据以 JSON 格式传输, 页面特效通过 JavaScript、JQuery 完成。

后端以 SpringMVC 结构设计, 分为 Controller (控制器)、Entity (实体类)、Mapper (数据库映射文件)、Service (业务逻辑) 四层。由于系统采用 Mybatis 技术, 并使用了 Mybatis-plus 插件, 因此通过拓展的 Mybatis-generator 逆向工程技术^[18]能快速搭建 SpringMVC 框架。在此基础上使用 Mybatis-plus 提供的基础增删改查方法或者重写新的方法来完成复杂的业务逻辑功能。

3.1 通用功能模块

3.1.1 全局异常处理

项目的开发中, 都不可避免会遇到各种异常需要处理。每个过程都单独处理异常, 系统的代码耦合度高, 工作量大且不好统一, 维护的工作量也很大。将所有类型的异常处理从各处理过程解耦出来, 这样既保证了相关处理过程的功能较单一, 也实现了异常信息的统一处理和维, 将异常信息统一捕捉并组装成固定格式的数据返回。

使用@ControllerAdvice 注解增强 Controller 可以实现全局异常处理。该模块可以处理认证异常、授权异常、自定义业务异常、其它未知异常以及返回模型和视图对象。

3.1.2 自定义返回值类型

该自定义返回值类型, 本质是 HashMap 类型, 可用于大部分控制器方法返回 JSON 格式的数据, 也方便返回请求的响应信息, 如请求成功或失败信息、登录成功或失败信息等, 甚至是返回全局异常处理模块的响应信息, 前端页面通过 Layer 弹窗显示响应信息。后文代码中的 R 返回值类型均指此类型。

3.2 用户端

用户端主要面向广大用户和志愿者, 提供了简洁大方的页面外观, 操作简单, 主要有注册登录、领养宠物、同城领养、拯救毛孩子等功能。

3.2.1 注册登录

注册登录模块是用户端最重要的功能之一, 用户可以匿名访问用户端首页, 如需使用用户端的各项功能, 必须通过登录认证。新用户可在登录页面点击“立即注册”链接在弹出的注册窗口上进行注册, 注册需通过前端数据校验方可成功。

用户登录流程: 用户输入用户名和密码后点击登录按钮, 将输入的用户名同数据库中用户数据进行对比, 若存在该用户名, 则将输入的密码与数据库中该用

户的密码进行对比，若相同则登录认证成功，跳转至用户端首页，失败则返回密码错误信息。

登录页面如图 3-1 所示。

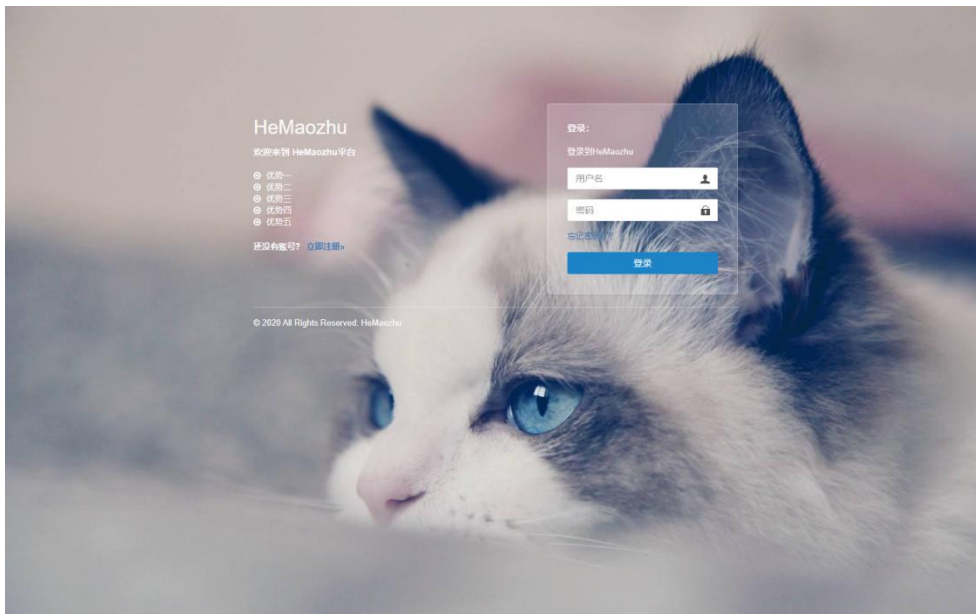


图 3-1 登录页面

用户登录认证实现方法如下：

系统调用控制器的登录请求，将用户名和密码封装为 Shiro 提供的 UsernamePasswordToken 对象，

```
UsernamePasswordToken token = new UsernamePasswordToken(username, password);
```

再实例化 Shiro 的 Subject 类，

```
Subject subject = SecurityUtils.getSubject();
```

调用 Subject 的 login 方法，

```
subject.login(token);
```

login 方法将调用用户端自定义 Realm 类的 doGetAuthenticationInfo 方法，通过 Mybatis 条件构造器（后文简称条件构造器），

```
QueryWrapper<Users> param = new QueryWrapper<>();
param.eq("username", username);
Users user = userMapper.selectOne(param);
```

查询数据库中用户表是否存在相同用户名的数据，若不存在则返回 UnknownAccountException 异常并处理，告知用户用户名不存在。若用户名存在则将输入的密码通过自定义的 MD5 盐值加密工具类中的方法加密，

```
public static String md5_private_salt(String source,String salt) {
//1.先用共盐对原始密码加密
md5_public_salt(source);
//2.再对加密后的密文用私盐加密一次
return new Md5Hash(md5_public_salt(source), salt, hashIterations).toString();}
```

同数据库中该用户按同样规则加密的密码进行对比，若相同则登录认证成功，否则返回 CredentialsException 异常并处理，告知用户密码错误。

用户注册流程：用户按提示输入相应信息，用户名需未被使用，两次输入密

码需一致，邮箱和手机号需按照正确格式输入并且未被其他用户使用，数据校验通过后调用 `regist` 请求存入数据库。

注册页面弹窗如图 3-2 所示。

图 3-2 注册页面

注册功能实现方法如下：

前端采用 `Validate` 前端校验数据非空和合法，通过 `Ajax` 实现后端校验，验证输入的用户名、邮箱、手机号是否被使用。例如用户名 `Ajax` 后端校验代码如下：

```
var used = false;
$.ajax({
  dataType : "JSON",
  type : "post",
  async:false,
  url : "${ctx}/fore/users/check",
  data:{
    username : value,
  },
  success:function(data){
    used = data;
  }
});
return used;
```

同理进行邮箱和手机号的后端校验。前后端校验通过后，调用 `regist` 请求将用户名和密码封装为 `Users` 对象并通过 `Service` 层通过上述盐值加密方法加密密码后存入数据库中。

```
//随机生成私盐
String salt = UUID.randomUUID().toString().toUpperCase().replace("-", "");
password = MD5Util.md5_private_salt(password, salt);
user.setPassword(password);
user.setSalt(salt);
return super.save(user);
```


领养宠物功能实现如下：

文章列表通过 Ajax 请求得到数据库数据后转为 JSON 格式，再以 JS 代码动态生成而成，数据在后台已进行分页。

文章列表 Ajax 代码如下：

```
function load(page){
    $.ajax({
        url: '${ctx}/fore/adopt/data',
        type: 'get',
        data: {
            current: page, // 当前页 1
            size: 5, // 一页显示多少 10
            deptId: $('#deptId').val(),
            title: $('#Name').val()
        },
        dataType: 'JSON',
        success : function(data){
            //动态生成列表
        }
    })
}
```

后台请求及分页代码如下：

```
@RequestMapping("/data")
@ResponseBody
public R data(Integer deptId, String title, Page<Adoption> page) {
    QueryWrapper<Adoption> param = new QueryWrapper<>();
    if(deptId != -1 && deptId != null) {
        param.eq("dept_id", deptId);
    }
    if(!StringUtils.isEmpty(title)) {
        param.like("title", title);
    }
    param.orderByDesc("create_time");
    page = adoptService.page(page, param);
    return R.ok(page)
        .put("current", page.getCurrent())
        .put("maxpage", page.getPages());
}
```

文章详情页面通过文章 id 查询数据库后得到一条数据，并封装为对象，JSP 页面通过 EL 表达式直接读取数据即可。图片内容在数据库中只存放其路径，文件存放在服务器上，页面中以相对路径形式读取即可。文章内容部分是通过富文本编辑器编辑，格式标签已设置好，因此仅需通过 EL 表达式读取即可。

后文其他文章详情页面显示均以此方法，则不再重复赘述。

3.2.3 同城领养

用户可通过此功能查所在城市其他用户发布的宠物领养信息，并与之取得联系自行完成同城领养流程。用户也可自己发布同城领养文章，发布后需通过管理端审核后才能在同城领养文章列表中显示。

该功能结构和实现大致与上述领养宠物模块相同，不同点在于该功能采用了高德地图 API 进行精确城市定位以及加入了文章发布功能。

同城领养文章列表页面如图 3-5 所示。



图 3-5 同城领养文章列表页面

高德地图 API 城市定位代码如下：

```
<!-- 高德地图 -->
<script type="text/javascript" src="https://webapi.amap.com/maps?v=1.4.15&key=(高德地图开
发者申请的key)&plugin=AMap.CitySearch"></script>
$(function(){
    //实例化城市查询类
    var citysearch = new AMap.CitySearch();
    var provinceinfo = "";
    var cityinfo = "";
    //自动获取用户IP，返回当前城市
    citysearch.getLocalCity(function(status, result) {
        provinceinfo = result.province; //获得省信息
        cityinfo = result.city; //获得城市信息
    });
});
```

同城领养文章发布流程：用户填写表单内容，通过前端数据校验后提交，后台将表单数据封装为实体对象后保存到数据库。若表单中上传了图片，则由服务器将文件上传到服务器，再在数据库中保存该图片的路径。

同城领养文章发布页面如图 3-6 所示。

图 3-6 同城领养文章发布页面

功能实现如下：

填写如图 3-6 表单中的内容后，若前端校验通过，则除照片外的所有数据封装为对应数据表的实体类对象，照片先封装为 `MultipartFile` 对象上传到服务器后再向对象中保存照片的路径，最后调用 `Service` 向数据库写入该对象数据。

图片文件上传代码如下：

```
String fileName = uploadFile.getOriginalFilename();
String suffix = fileName.substring(fileName.lastIndexOf("."));
String uuid = UUID.randomUUID().toString().toUpperCase().replace("-", "");
fileName = uuid + suffix;
String foldername = commonService.getToday();
String path = fileLocation + "/" + (自定义文件夹名) + "/" + foldername;
String parent = session.getServletContext().getRealPath(path);
File file = new File(parent, fileName);
if (!file.exists()) {
    file.mkdirs();
}
uploadFile.transferTo(file); // 上传文件
return path + "/" + fileName; // 返回文件保存路径
```

控制器保存实体类对象代码如下：

```
@RequestMapping(value = "/add", method = RequestMethod.POST)
@ResponseBody
public R add(Cityadopt cd, @RequestParam(value = "file", required = false) MultipartFile file) {
    return cdService.add(cd, file);
}
```

文章内容部分使用富文本编辑器 `Summernote` 编写，该编辑器上也可以增加

图片，但该编辑器将图片以 Base64 编码^[19]格式进行保存，极大的增加了数据库的负担，故需要重写该编辑器的图片上传方法。

根据官方 API，需重写其 callbacks 函数中的 onImageUpload 方法，

```
callbacks: {
  onImageUpload: function(files, editor, $editable) {
    for(i=0;i<files.length;i++){
      uploadImg(files[i], editor, $editable);
    }
  }
}
```

其中 uploadImg 方法为自定义方法，将编辑器中的图片文件通过 Ajax 调用上述图片文件上传方法上传到服务器并返回其路径再通过相对路径回显到页面，代码如下：

```
function uploadImg(file, editor, $editable){
  var filename = false;
  try{
    filename = file['name'];
  } catch(e){
    filename = false;
  }
  var url = "${ctx}";
  //防止在图片在编辑器内拖拽引发第二次上传导致的提示错误
  if(!filename){$("#note-alarm").remove();}
  data = new FormData();
  data.append("file", file);
  $.ajax({
    data: data,
    type: "POST",
    //图片上传出来的url，返回的是图片上传后的路径，http格式
    url: "${ctx}/fore/cityadopt/upload",
    contentType: false,
    cache: false,
    processData: false,
    success: function (response) {
      if (response.code == 0) {
        url += response.url;
        $('#summernote').summernote('insertImage', url, filename);
        $(".img").addClass("img-responsive");
      } else {
        window.parent.layer.alert(response.msg, {icon : 5, offset : '0px'});
      }
    }
  });
}
```

同时应提供删除服务器图片功能防止过多垃圾文件占用大量服务器资源，但 Summernote 没有提供删除接口，因此选择使用 Mutationbobserver 接口，它提供了监视对 DOM 树所做更改的能力，并通过 DOM 树提取出网页改变部分的内容^[20]。它被设计为旧的 Mutation Events 功能的替代品，该功能是 DOM3 Events 规范的一部分。利用 Mutationbobserver 监控富文本编辑器中标签的移除，来删除其对应的服务器上的图片文件，代码如下：

```
// Firefox和Chrome早期版本中带有前缀
var MutationObserver = window.MutationObserver || window.WebKitMutationObserver ||
window.MozMutationObserver;
// 选择目标节点
var target = document.querySelector('.note-editable');
// 创建观察者对象
var observer = new MutationObserver(function(mutations) { //观察对象的回调函数
    mutations.forEach(function(mutation) { //forEach: 遍历所有MutationRecord
        if(mutation.removedNodes[0]!=null) {
            if(mutation.removedNodes[0].tagName == "IMG") {
                var src = mutation.removedNodes[0].src;
                deleteImg(src);
            }
        }
    });
});
// 配置观察选项:
var config = { attributes: true, childList: true, characterData: true ,subtree:true};
// 传入目标节点和观察选项
observer.observe(target, config);
```

使用修改后的 Summernote 富文本编辑器，既为用户带来了更好的体验，也减轻了服务器的压力，后文使用同样的富文本编辑器，则不再赘述。

3.2.4 拯救毛孩子

用户使用该功能模块需要先注册为志愿者，系统将自动提示用户进行注册，志愿者注册将包含用户隐私信息。成为为志愿者后该功能将显示用户所在城市所有志愿者发布的救助文章，用户也可以发布新的救助文章，但如果该城市没有平台设立的线下站点，则用户无法在该城市使用救助功能，但仍可以查看其他城市的相关内容。

该模块的城市定位功能、文章列表、文章发布功能同上述同城领养功能模块的各项功能原理相同，此处不再重复。

拯救毛孩子文章列表页面如图 3-7 所示。



图 3-7 拯救毛孩子文章列表页面

拯救毛孩子文章详情页面如图 3-8 所示。

网红桥下的小可爱！

在成都网红桥下的桥墩上，用无人机看到是一只很小的蓝白长毛猫，大概2个月大，不知道怎么到那里的，救助可能需要绳梯从桥边爬下去，救助难度极大，危险系数很高，需要站点工作人员和广大志愿者的帮助。



救助联系电话：18280473733
QQ：964193177

图 3-8 拯救毛孩子文章详情页面

用户注册志愿者流程：用户需正确填写真实姓名、性别、联系方式、身份证信息等信息等隐私信息，并选择注册的线下站点和志愿服务的区域，表单数据通过数据校验后提交并保存到数据库。注册成功后将自动跳转到救助文章列表。

志愿者注册页面如图 3-9 所示。

您还不是志愿者，请注册

真实姓名：

性别：

请选择性别 ▾

联系方式：

身份证信息：

注册站点：

请选择站点 ▾

志愿地区：

-- 省 -- ▾ -- 市 -- ▾ -- 区/县 -- ▾

提交

重置

图 3-9 志愿者注册页面

志愿者注册功能实现如下：

页面通过 Shiro 标签获取当前登录用户的 id 信息,

```
<shiro:principal property="userId" />
```

用户填写表单数据后, 连同用户 id 一起封装为实体对象, 调用 Service 存入数据库, 同时修改用户表中该用户的 isvol 字段, 表示其已成为志愿者。控制器代码如下:

```
@RequestMapping("/add/{userId}")
public String add(@PathVariable Integer userId, Model model) {
    model.addAttribute("userId", userId);
    QueryWrapper<Dept> param = new QueryWrapper<>();
    param.ne("type", 1);
    model.addAttribute("deptlist", deptService.list(param));
    return "fore/voluteer/vol_add";
}
```

3.2.5 养宠小知识

该功能模块较小, 设计目的是为了留住用户, 提升用户体验。用户可以查看管理端发布的养宠相关的知识科普文章, 特别是对于在本平台领养了宠物的新手饲养人有更好的体验。该功能模块同领养宠物功能模块实现原理相同, 因此不再重复其实现方法。

养宠小知识文章列表页面如图 3-10 所示。



图 3-10 养宠小知识文章列表页面

养宠小知识文章详情页面如图 3-11 所示。



图 3-11 养宠小知识文章详情页面

3.2.6 我的文章

用户在我的文章模块可以管理自己发布的同城领养文章和拯救毛孩子文章，文章以上述相同功能模块的文章列表页面的形式显示，不同的是用户可以修改文章的状态或者删除文章，以及查看文章是否通过审核。用户点击文章标题后跳转到文章编辑页面，用户可以如发布文章时一样修改文章的内容。

3.2.6.1 我的同城领养文章

我的同城领养文章页面如图 3-12 所示。

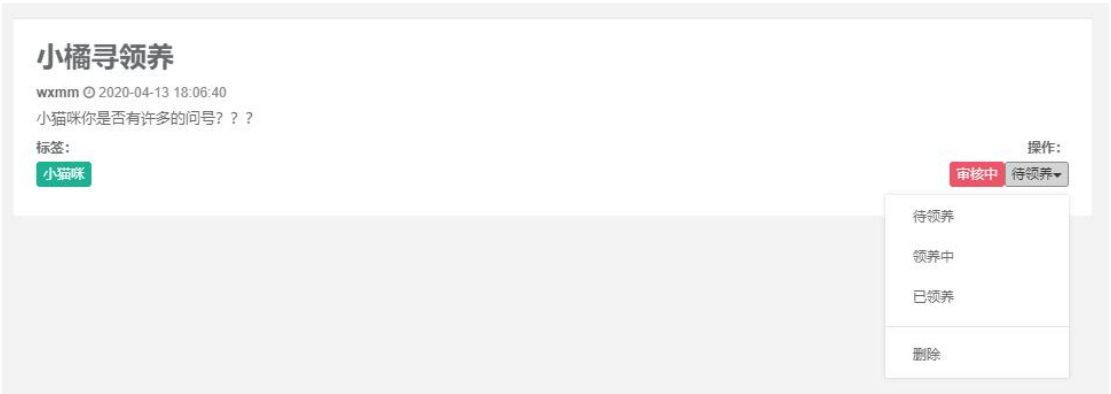


图 3-12 我的同城领养文章页面

用户可以点击下拉按钮修改发布的同城领养文章状态，如有某用户看到该文

章后联系文章作者，谈妥准备领养时，作者可以修改文章状态为领养中，其他用户便能得知并不再联系作者，文章也会排序到靠后的位置。

状态修改控制器代码如下：

```
@RequestMapping("/status")
@ResponseBody
public R status(Integer cityadoptId, Integer status) {
    Cityadopt cd = cdService.getById(cityadoptId);
    //未通过审核无法修改状态
    if(cd.getVerify() == 0) {
        return R.error("审核中...");
    }
    else {
        cd.setStatus(status);
        cdService.updateById(cd);
        return R.ok();
    }
}
```

3.2.6.2 我的拯救毛孩子文章

我的拯救毛孩子文章页面如图 3-13 所示。



图 3-13 我的拯救毛孩子文章页面

拯救毛孩子文章状态由管理端根据实际救助情况进行修改，该功能属于管理端，不在此说明。用户仅可以查看发布的文章是否通过审核以及删除文章。

3.3 管理端

3.3.1 登录

管理员登录功能是管理端最重要的功能模块，只有身份认证成功后才能进入管理端首页并使用授权的功能。

管理员登录流程：管理员输入用户名和密码后，系统将输入的用户名同数据库中用户数据进行对比，若存在该用户名，则将输入的密码与数据库中该用户的密码进行对比，若相同则查询该管理员数据的 `status`（状态）字段，若状态为禁用或锁定，则拒绝登录并返回相应状态信息，若状态为正常登录认证成功，跳转至管理端端首页。若密码不相同则返回密码错误信息。

管理员登录页面如图 3-14 所示。

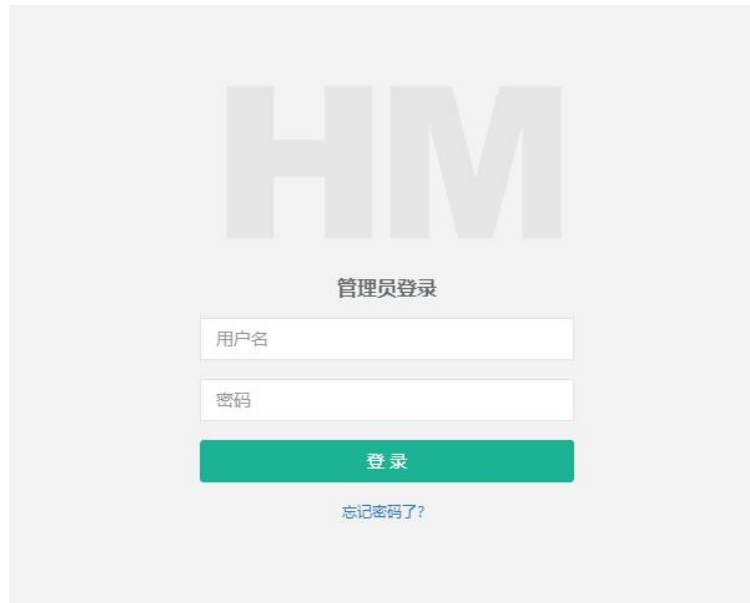


图 3-14 管理员登录页面

登录功能的身份认证的实现同用户端登录身份认证原理相同，条件构造器变为：

```
// 根据用户名去 DB 查询对应的用户信息
QueryWrapper<AdminUser> param = new QueryWrapper<>();
param.eq("username", username);
AdminUser user = userMapper.selectOne(param);
```

用户密码验证正确后判断管理员状态，分别以 DisabledAccountException 处理账号禁用异常，LockedAccountException 处理账号锁定异常，AccountException 处理未知异常。

管理员身份认证成功后将进行授权操作，通过查询数据库管理员所拥有的角色的权限分配管理员相应的功能菜单以及操作权限。

授权功能实现如下：

授权操作由管理端的自定义 Realm 重写 doGetAuthorizationInfo 方法实现，代码如下：

```
@Override
protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principals) {
    AdminUser user = (AdminUser) principals.getPrimaryPrincipal();
    // 简单授权信息对象，对象中包含用户的角色和权限信息
    SimpleAuthorizationInfo info = new SimpleAuthorizationInfo();
    // 从数据库获取当前用户的角色 通过用户名查询该用户拥有的角色名称
    Set<String> roleNameSet = userMapper.selectUserRoleNameSet(user.getUserId());
    info.addRoles(roleNameSet);
    // 从数据库获取当前用户的权限 通过用户名查询该用户拥有的权限名称
    Set<String> permissionNameSet =
    userMapper.selectUserPermissionNameSet(user.getUserId());
    Set<String> permissions = new HashSet<>();
    for (String name : permissionNameSet) {
        for (String permission : name.split(",")) {
            permissions.add(permission);
        }
    }
}
```

```

info.addStringPermissions(permissions);
System.out.println("授权完成....");
return info;
}

```

该方法返回一个简单授权信息对象，对象中包含用户的角色和权限信息，管理员进行后续操作时是否拥有相应权限需要从该信息对象中查询。

管理员因其角色和权限的不同于首页显示不同的功能菜单，超级管理员拥有所有权限，超级管理员菜单页面如图 3-15 所示。



图 3-15 超级管理员菜单页面

获取用户菜单树代码如下：

```

@Override
public List<TreeNode> getMenuTreeByUserId(Integer id) {
    // 查询用户拥有的菜单资源
    List<Menu> menuList = userMapper.selectMenuList(id);
    if (menuList.isEmpty()) {
        return new ArrayList<>();
    }
    // 存储父 id 是 0 的节点的 id
    List<Integer> nodeIds = new ArrayList<>();
    List<TreeNode> treeNodeList = new ArrayList<>();
    for (Menu menu : menuList) {
        TreeNode treeNode = new TreeNode();
        treeNode.setId(menu.getResourceId());
        treeNode.setName(menu.getName());
        treeNode.setParentId(menu.getParentId());
        treeNode.setUrl(menu.getUrl());
        treeNode.setIcon(menu.getIcon());
        treeNodeList.add(treeNode);
        if (treeNode.getParentId() == 0) {
            nodeIds.add(treeNode.getId());
        }
    }
    TreeUtil treeUtil = new TreeUtil(treeNodeList);
    List<TreeNode> treeNodeData = new ArrayList<>();
}

```

```

for (Integer nodeId : nodeIds) {
    treeNodeData.add(treeUtil.generateTree(nodeId));
}
return treeNodeData;
}

```

首页菜单使用 metisMenu 插件动态生成后台获取的菜单树，JS 代码如下：

```

<script id="tpl-menu" type="text/html">
  {{each menuList menu}}
  <li>
    <a href="#">
      <i class="{{menu.icon}}"></i>
      <span class="nav-label">{{menu.text}}</span>
      <span class="fa arrow"></span>
    </a>
    <ul class="nav nav-second-level">
      {{each menu.nodes sub_menu}}
      <li>
        <a class="J_menuItem" href="{{ctx}}/{{sub_menu.url}}">
          {{sub_menu.text}}</a>
        </li>
      {{/each}}
    </ul>
  </li>
  {{/each}}
</script>
<script>
  $.ajax({
    //获取用户菜单树请求
    url : '{{ctx}}/page/menu',
    type : 'get',
    async: false,
    dataType : 'JSON',
    success : function(response) {
      var html = template('tpl-menu', {
        menuList : response.data
      });
      $('#sidebar-collapse .side-menu').html(html);
      $('#side-menu').metisMenu();
    }
  });
</script>

```

3.3.2 系统管理

仅超级管理员拥有系统管理的权限，可以对所有管理员进行增删改查操作，并给他们分配角色，也可以对所有角色进行增删改查操作，并给角色分配资源权限，还可以对所有资源权限进行增删改查操作。

3.3.2.1 管理员管理

管理员管理页面如图 3-16 所示。

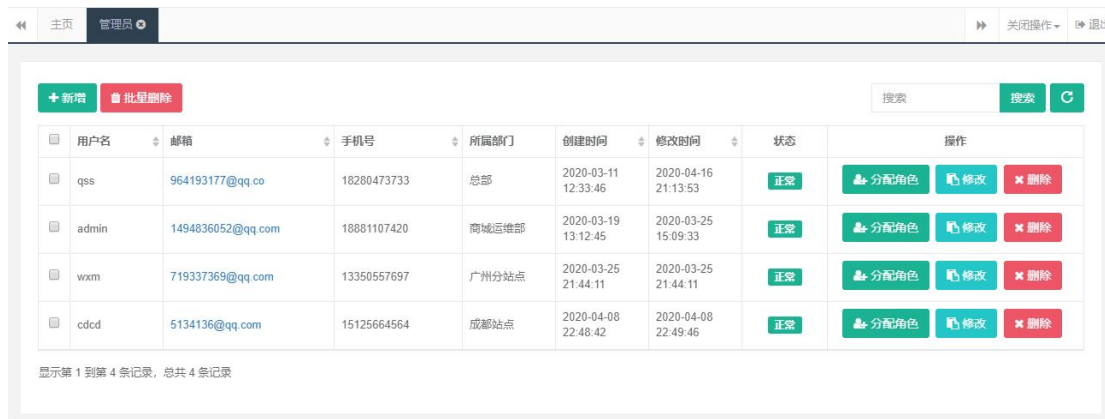


图 3-16 管理员管理页面

管理端所有表格页面均采用 Bootstrap-table 插件，简化了表格生成和分页查询功能。此处分页查询采用客户端分页方式，后文均采用服务端分页方式。

获取列表数据代码如下：

```
@RequestMapping("/data")
@ResponseBody
@RequiresPermissions("sys:user:list")
public R data(String username) {
    QueryWrapper<AdminUser> param = new QueryWrapper<>();
    if (!StringUtils.isEmpty(username)) {
        param.like("username", username);
    }
    // 查询满足条件的数据集
    List<AdminUser> rows = userService.getAdminUserList(param);
    // 查询满足条件的总记录数
    Integer total = userService.count(param);
    return R.ok()
        .put("count", total)
        .put("data", rows);
}
```

分配角色页面如图 3-17 所示。

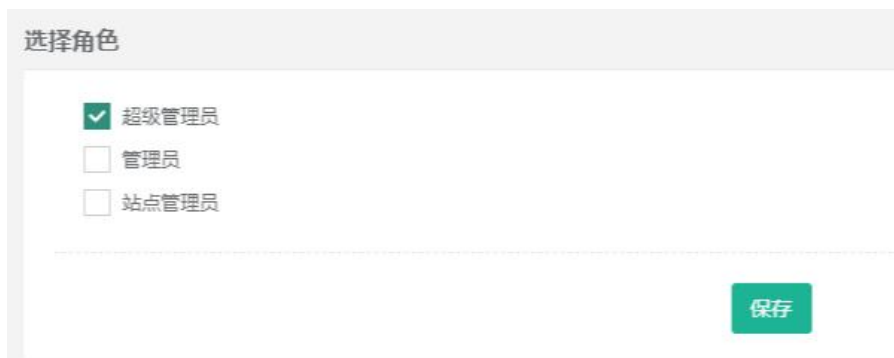


图 3-17 分配角色页面

3.3.2.2 角色管理

分配角色代码如下：

```
@RequestMapping(value = "/assign/role", method = RequestMethod.POST)
@ResponseBody
@RequiresPermissions("sys:user:assign:role")
```

```

public R assignRole(Integer userId, @RequestParam(value = "roleId",required = false)
List<Integer> roleIdList) {
    userRoleService.save(userId,roleIdList);
    return R.ok();
}

```

角色管理页面如图 3-18 所示。



图 3-18 角色管理页面

采用服务端分页获取列表数据代码如下：

```

@RequestMapping("/data")
@ResponseBody
@RequiresPermissions("sys:role:list")
public R data(String roleName, Page<AdminRole> page, OrderItem order) {
    //0.处理排序列名驼峰规则
    page = commonService.handleOrder(page, order);
    //1.构造查询条件构造器
    QueryWrapper<AdminRole> queryWrapper = new QueryWrapper<>();
    if (!StringUtils.isEmpty(roleName)) {
        queryWrapper.like("name", roleName);
    }
    //2.分页查询
    roleService.page(page, queryWrapper);
    //3.返回分页数据
    return R.ok(page);
}

```

资源分配页面如图 3-19 所示。



图 3-19 分配资源页面

分配资源代码如下：

```
@RequestMapping(value = "/assign/resource/{roleId}", method = RequestMethod.POST)
@ResponseBody
@RequiresPermissions("sys:role:assign:resource")
public R assignResource(@PathVariable Integer roleId, @RequestBody
List<AdminRoleResource> roleResourceList) {
    roleResourceService.save(roleId, roleResourceList);
    return R.ok();
}
```

3.3.2.3 资源管理

资源管理页面如图 3-20 所示。

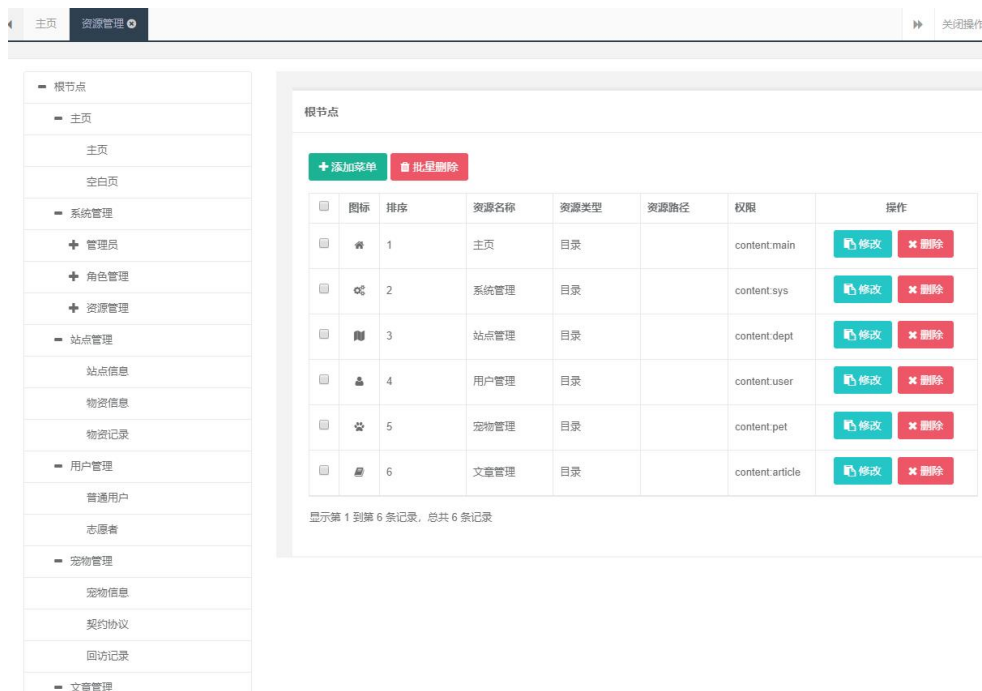


图 3-20 资源管理页面

资源管理页面由左侧菜单树和右侧列表组成，点击左侧任意节点右侧则会显示对应节点的子节点的信息。

获取左侧菜单树代码如下：

```
@RequestMapping("/tree")
@ResponseBody
@RequiresPermissions("sys:resource:list")
public R tree() {
    TreeNode treeNode = resourceService.getTreeById(0);
    return R.ok("请求成功", treeNode);
}
```

获取列表数据代码如下：

```
@RequestMapping("/data")
@ResponseBody
@RequiresPermissions("sys:resource:list")
public R data(@RequestParam(defaultValue = "0") Integer parentId, Page<AdminResource>
page){
    QueryWrapper<AdminResource> queryWrapper = new QueryWrapper<>();
    queryWrapper.eq("parent_id", parentId);
    queryWrapper.orderByAsc("order_num");
}
```

```
resourceService.page(page,queryWrapper);
return R.ok(page);
}
```

3.3.3 站点管理

超级管理员和管理员可使用该模块对所有站点以及部门信息进行增删改查操作，并对总部的物资进行调度。站点管理员能调度所属站点的物资信息以及当物资短缺时向总部申请物资。所有物资的申请、调度、消耗记录均可在物资记录中查看。

3.3.3.1 站点信息管理

站点信息管理页面如图 3-21 所示。

<input type="checkbox"/>	站点名称	站点负责人	负责人电话	站点类型	站点所在地	服务范围	操作
<input type="checkbox"/>	总部	卿二思		总部	四川省成都市青白江区	全国	i 更多信息 修改 x 删除
<input type="checkbox"/>	南城运维部			总部	广东省广州市番禺区	全国	i 更多信息 修改 x 删除
<input type="checkbox"/>	广州分站点			分站点	广东省广州市番禺区	广东省广州市	i 更多信息 修改 x 删除
<input type="checkbox"/>	成都站点	test23		分站点	四川省成都市锦江区	四川省成都市	i 更多信息 修改 x 删除

显示第 1 到第 4 条记录, 总共 4 条记录

图 3-21 站点信息管理页面

获取列表数据代码如下：

```
@RequestMapping("/data")
@ResponseBody
@RequiresPermissions("dept:info:list")
public R data(String deptName, Page<Dept> page){
    QueryWrapper<Dept> param = new QueryWrapper<>();
    if(!StringUtils.isEmpty(deptName)) {
        param.like("dept_name", deptName);
    }
    param.orderByAsc("dept_id");
    // 查询满足条件的数据集
    page = deptService.page(page, param);
    return R.ok(page);
}
```

3.3.3.2 物资信息管理

总部物资信息管理页面如图 3-22 所示。

<div>+ 入库</div> <div>+ 补仓</div>		<div>搜索</div> <div>搜索</div> <div>刷新</div>						
物资名称	剩余数量	物资所属站点	单价	入库时间	最近补仓时间	状态	备注	操作
猫粮3号	43	总部	40.99	2020-03-31 11:53:26	2020-04-15 03:52:12	<div>补仓</div>		<div>历史记录</div> <div>修改</div>
猫粮4号	100	总部	40.99	2020-03-31 11:53:26	2020-04-01 19:12:46	<div>补仓</div>	-	<div>历史记录</div> <div>修改</div>
猫粮5号	199	总部	20	2020-03-31 16:49:28	2020-04-01 21:23:45	<div>补仓</div>	-	<div>历史记录</div> <div>修改</div>
猫砂1号	299	总部	56.43	2020-03-31 16:59:59	2020-03-31 16:59:59	<div>补仓</div>		<div>历史记录</div> <div>修改</div>
猫砂2	35	总部	45.32	2020-03-31 17:18:07	2020-03-31 17:18:07	<div>补仓</div>		<div>历史记录</div> <div>修改</div>
化毛膏1号	300	总部	56.34	2020-03-31 23:47:33	2020-03-31 23:47:33	<div>补仓</div>		<div>历史记录</div> <div>修改</div>
化毛膏2号	100	总部	40	2020-03-31 23:47:51	2020-03-31 23:47:51	<div>补仓</div>		<div>历史记录</div> <div>修改</div>
猫砂盆1号	100	总部	123	2020-03-31 23:48:58	2020-03-31 23:48:58	<div>补仓</div>		<div>历史记录</div> <div>修改</div>
猫粮2号	99	总部	40.99	2020-03-31 11:53:26	2020-03-31 14:17:00	<div>补仓</div>	-	<div>历史记录</div> <div>修改</div>
猫粮1号	199	总部	40.99	2020-03-31 11:53:26	2020-03-31 14:17:55	<div>补仓</div>	-	<div>历史记录</div> <div>修改</div>

显示第 1 到第 10 条记录, 总共 11 条记录 每页显示 10 条记录

«

«

1

2

»

图 3-22 总部物资信息管理页面

点击入库可以录入新的物资信息；点击补仓按钮可以增加物资数量，需要先选择列表数据项；点击历史记录则可以查看该物资的申请、调度、消耗记录，同下文物资记录功能相同。

站点物资信息管理页面如图 3-23 所示。

<div>+ 入库</div> <div>+ 申请</div> <div>输入数量</div> <div>日 出库</div>		<div>搜索</div> <div>搜索</div> <div>刷新</div>						
物资名称	剩余数量	物资所属站点	单价	入库时间	最近补仓时间	状态	备注	操作
猫粮3号	42	广州分站点	40.99	2020-03-31 11:53:26	2020-04-15 03:52:40	<div>补仓</div>	-	<div>历史记录</div>
猫砂1号	54	广州分站点	56.43	2020-03-31 17:44:19	2020-03-31 17:44:19	<div>补仓</div>		<div>历史记录</div>
猫粮5号	99	广州分站点	20	2020-04-01 21:20:53	2020-04-01 21:32:32	<div>补仓</div>		<div>历史记录</div>
猫粮1号	199	广州分站点	40.99	2020-03-31 11:53:26	2020-03-31 14:17:55	<div>补仓</div>	-	<div>历史记录</div>
猫粮2号	144	广州分站点	40.99	2020-03-31 11:53:26	2020-04-01 21:04:55	<div>补仓</div>	-	<div>历史记录</div>
猫粮4号	200	广州分站点	40.99	2020-03-31 11:53:26	2020-04-01 21:03:55	<div>补仓</div>	-	<div>历史记录</div>

显示第 1 到第 6 条记录, 总共 6 条记录

图 3-23 站点物资信息管理页面

站点物资均通过总部物资调度而来，无法自行加入其他物资信息。点击入库录入总部调度来的新物资信息；点击申请可以向总部申请物资，如果总部该物资数量不足申请数量则返回提示信息，在一次申请流程未完成前无法进行二次申请；点击出库需要先选择数据项并输入数量，表示消耗的物资数量，若输入数量大于库存数量则返回提示信息。

获取数据列表代码如下：

```
@RequestMapping("/data")
@ResponseBody
public R data(Integer deptId, String name, Page<DeptGoods> page, OrderItem order) {
    //0.处理排序列名驼峰规则
```



```

page = commonService.handleOrder(page, order);
//1.构造查询条件构造器
QueryWrapper<DeptGoods> param = new QueryWrapper<>();
if(!StringUtils.isEmpty(name)) {
    param.like("name", name);
}
param.eq("dept_id", deptId);
if(StringUtils.isEmpty(order.getColumn())) {
    param.orderByDesc("status");
}
//2.分页查询
deptgoodsService.page(page, param);
//3.返回分页数据
return R.ok(page);
}

```

入库功能代码如下：

```

@RequestMapping(value = "/add", method = RequestMethod.POST)
@ResponseBody
public R add(DeptGoods goods) {
    deptgoodsService.save(goods);
    return R.ok();
}

```

补仓功能代码如下：

```

@RequestMapping(value = "/more", method = RequestMethod.POST)
@ResponseBody
public R more(DeptGoods goods) {
    DeptGoods newgoods = deptgoodsService.getById(goods.getGoodsId());
    deptgoodsService.more(newgoods, goods.getCount());
    logService.log(newgoods, goods.getCount());
    return R.ok();
}

```

申请功能代码如下：

```

@RequestMapping(value = "/request", method = RequestMethod.POST)
@ResponseBody
public R request(DeptGoods goods) {
    DeptGoods newgoods = deptgoodsService.getById(goods.getGoodsId());
    DeptGoods parentgoods = deptgoodsService.getById(newgoods.getParentId());
    QueryWrapper<DeptgoodsLog> param = new QueryWrapper<>();
    param.eq("goods_id", goods.getGoodsId());
    param.ne("status", 1);
    param.ne("status", 0);
    if(logService.count(param) != 0){
        return R.error("上次申请为完成，请勿重复申请");
    } else if(goods.getCount() > parentgoods.getCount()) {
        return R.error("总部库存不足");
    } else {
        logService.request(newgoods, goods.getCount());
        return R.ok();
    }
}

```

出库功能代码如下：

```

@RequestMapping("/expand/{goodsId}/{count}")
@ResponseBody
public R expand(@PathVariable Integer goodsId, @PathVariable Integer count) {
    DeptGoods newgoods = deptgoodsService.getById(goodsId);
}

```

```

    if(deptgoodsService.expand(newgoods, count)) {
        logService.log(newgoods, -count);
        return R.ok();
    }
    else
        return R.error("库存不足");
}

```

3.3.3.3 物资记录管理

物资记录页面如图 3-24 所示。

审核

驳回

2020-03-29

到

2020-04-30

物资名称

所属部门

请求数量/变化数量

原库存

请求时间

变化时间

收货时间

状态

操作

猫粮3号

广州分站点

1

41

2020-04-15 03:51:58

2020-04-15 03:52:40

2020-04-15 03:52:40

已完成

-

猫粮3号

总部

-1

43

-

2020-04-15 03:52:12

-

已完成

-

猫粮3号

广州分站点

2

36

-

2020-04-01 21:46:56

-

已完成

-

猫粮3号

广州分站点

2

39

2020-04-01 21:35:54

2020-04-01 21:46:56

2020-04-01 21:46:56

已完成

-

猫粮3号

总部

-2

50

-

2020-04-01 21:46:09

-

已完成

-

猫粮3号

广州分站点

-3

39

-

2020-04-01 21:45:15

-

已完成

-

猫粮5号

广州分站点

-178

277

-

2020-04-01 21:32:32

-

已完成

-

猫粮5号

广州分站点

100

200

2020-04-01 21:21:54

2020-04-01 21:24:50

2020-04-01 21:24:50

已完成

-

猫粮2号

广州分站点

45

34

2020-04-01 17:54:27

2020-04-01 21:04:55

2020-04-01 21:04:55

已完成

-

猫粮4号

广州分站点

200

0

2020-04-01 19:03:09

2020-04-01 21:03:55

2020-04-01 21:03:55

已完成

-

显示第 1 到第 10 条记录, 总共 14 条记录 每页显示

10

条记录

«

<

1

2

>

»

图 3-24 物资记录管理页面

记录信息默认显示当前日前 7 天的数据, 也可以使用日期选择器查看对应日期范围内的数据。

管理员和超级管理员可以查看所有站点的物资记录, 并对物资申请记录进行审核或驳回, 审核后记录状态变为发货状态并减少总部相应物资的数量, 驳回则状态变为驳回状态。站点管理员仅能查看其所属站点的物资记录, 无审核功能, 在申请物资记录审核通过后, 物资调度到货时需要点击确认收货操作, 记录状态变为已完成状态并增加站点相应物资的数量。

获取数据列表代码如下:

```

@RequestMapping("/data")
@ResponseBody
public R data(Integer deptId, Integer goodsId, String pd, String d, Page<DeptgoodsLog> page, OrderItem order) {
    //0.处理排序列名驼峰规则
    page = commonService.handleOrder(page, order);
    //1.构造查询条件构造器
    QueryWrapper<DeptgoodsLog> param = new QueryWrapper<>();
    if(goodsId != null) {
        param.eq("goods_id", goodsId);
    }
    if(deptId != 1) {
        param.eq("dept_id", deptId);
    }
}

```

```

        Date today = commonService.parse(d);
        Date pastday = commonService.parse(pd);
        param.and(QueryWrapper -> QueryWrapper.between("update_time", pastday,
commonService.addeday(today, 1)).or()
                .between("request_time", pastday, commonService.addeday(today, 1)).or()
                .between("confirm_time", pastday, commonService.addeday(today, 1)));
        if(StringUtils.isEmpty(order.getColumn())) {
            param.orderByDesc("update_time");
        }
        //2.分页查询
        logService.page(page, param);
        //3.返回分页数据
        return R.ok(page);
    }

```

审核功能代码如下：

```

@RequestMapping("/accept/{id}")
@ResponseBody
public R accept(@PathVariable Integer id) {
    if(logService.accept(id))
        return R.ok();
    else
        return R.error("库存不足");
}

```

驳回功能代码如下：

```

@RequestMapping("/ban/{id}")
@ResponseBody
public R ban(@PathVariable Integer id) {
    if(logService.ban(id))
        return R.ok();
    else
        return R.error("已审核，驳回失败");
}

```

确认收货功能代码如下：

```

@RequestMapping("/check/{id}")
@ResponseBody
public R check(@PathVariable Integer id) {
    logService.check(id);
    return R.ok();
}

```

3.3.4 用户管理

超级管理员和管理员可以对普通用户信息和志愿者信息进行修改和删除操作，站点管理员仅能操作在其站点注册的志愿者信息。由于志愿者信息包含如身份证信息等隐私数据，故读取的身份证信息是不完整的，除非数据库被攻破，否则任何人无法取得完整的身份证数据。

3.3.4.1 普通用户管理

普通用户管理页面如图 3-25 所示。

	删除	搜索	搜索	
<input type="checkbox"/>	用户名	邮箱	手机号码	操作
<input type="checkbox"/>	qss3	740551032@qq.com	13425632743	修改
<input type="checkbox"/>	wxmm	9641931777@qq.com	15161773475	修改
<input type="checkbox"/>	1618130234	555555@qq.com	13558891990	修改
<input type="checkbox"/>	卿三思猪猪	719337369@qq.com	13350557697	修改
<input type="checkbox"/>	zyyky	123456789@qq.com	15182386111	修改
<input type="checkbox"/>	bobxiao	736385398@qq.com	13111111111	修改
<input type="checkbox"/>	admin	admin@admin.cn	13011111223	修改
<input type="checkbox"/>	柠檬小鱼鱼	1099460048@qq.com	18224418301	修改
<input type="checkbox"/>	aaa啊	123@qq.com	15520675348	修改
<input type="checkbox"/>	aaaaa	1234@qq.com	15520675349	修改

显示第 1 到第 10 条记录, 总共 10 条记录

图 3-25 普通用户管理页面

获取数据列表代码如下:

```

@RequestMapping("/data")
@ResponseBody
public R data(String username, Page<Users> page) {
    //1.构造查询条件构造器
    QueryWrapper<Users> param = new QueryWrapper<>();
    if (!StringUtils.isEmpty(username)) {
        param.like("username", username);
    }
    //2.分页查询
    userService.page(page, param);
    //3.返回分页数据
    return R.ok(page);
}

```

3.3.4.2 志愿者管理

志愿者管理页面如图 3-26 所示。

删除

搜索

搜索

<input type="checkbox"/>	用户名	注册站点	真实姓名	性别	联系方式	身份证号	志愿地区	操作
<input type="checkbox"/>	wxmm	成都站点	枚枚	女	15181231413	510*****912	四川省成都市青白江区	<div><div>详细信息</div><div>修改</div></div>
<input type="checkbox"/>	qss3	成都站点	卿三思	男	18280473733	510*****912	四川省成都市金牛区	<div><div>详细信息</div><div>修改</div></div>

显示第 1 到第 2 条记录, 总共 2 条记录

图 3-26 志愿者管理页面

获取数据列表代码如下:

```

@RequestMapping("/data")
@ResponseBody
public R data(String realname, Page<Volunteer> page) {
    QueryWrapper<Volunteer> param = new QueryWrapper<>();
}

```

```

if(!StringUtils.isEmpty(realname)) {
    param.like("realname", realname);
}
param.orderByAsc("vol_id");
page = volService.page(page, param);
return R.ok(page);
}

```

3.3.5 宠物管理

超级管理员和管理员可以对所有宠物信息进行增删改查操作，查看、修改和删除所有契约协议信息，查看和删除所有回访记录信息。

站点管理员则仅能操作所属站点的宠物信息。当有领养人领养宠物后，站点管理员需要录入领养人签的协议照片和与领养宠物合照的契约照片，并录入领养人隐私信息，以确保回访时能准确联系领养人。站点工作人员需要定期回访领养人，检查被领养宠物的健康状况，检查领养人信息是否变动等，并记录到回访记录中。

3.3.5.1 宠物信息管理

宠物信息管理页面如图 3-27 所示。

+ 新增		删除						搜索	搜索	C
宠物名	救助站点	类型	品种	性别	是否被领养	记录时间	操作			
仔仔	广州分站点	猫咪	奶牛	公	否	2020-04-07 16:11:01	详细信息	修改		
22	广州分站点	猫咪	暹罗	母	否	2020-04-08 22:16:45	详细信息	修改		
33	成都站点	猫咪	梨花	公	否	2020-04-08 22:47:26	详细信息	修改		
5555	广州分站点	狗狗	柯基	公	否	2020-04-11 11:48:18	详细信息	修改		
摩洛哥	成都站点	狗狗	哈士奇	公	是	2020-04-11 21:36:24	详细信息	修改		
丑丑	成都站点	猫咪	梨花	公	否	2020-04-30 10:46:21	详细信息	修改		

显示第 1 到第 6 条记录，总共 6 条记录

图 3-27 宠物信息管理页面

宠物信息详情页面如图 3-28 所示。

类型:

品种:

救助站点:

宠物名:

性别:

年龄:

生日:

照片:

成都官方领养

姓名: 丑丑 年纪: 5个月

性别: MM 绝育: 否

疫苗: 否 来源: 流浪

备注:

长按识别二维码加微信带它回家
握爪盖章·余生请多指教



是否被领养:

备注: 未绝育, 未打疫苗, 来源: 流浪

图 3-28 宠物信息详情页面

获取数据列表代码如下:

```
@RequestMapping("/data")
@ResponseBody
public R data(Integer deptId, String petname, Page<Pet> page) {
    QueryWrapper<Pet> param = new QueryWrapper<>();
    if(!StringUtils.isEmpty(petname)) {
        param.like("petname", petname);
    }
    if(deptId != 1) {
        param.eq("dept_id", deptId);
    }
    param.orderByAsc("pet_id");
    page = petService.page(page, param);
    return R.ok(page);
}
```

3.3.5.2 契约协议管理

契约协议管理页面如图 3-29 所示。

+ 新增		删除					搜索	搜索	刷新
领养站点	领养人	联系方式	居住地址	领养宠物	记录时间	操作			
<input type="checkbox"/> 成都站点	卿三思	18280473733	测试地址	丑丑	2020-04-30 20:55:23	详细信息	修改		
<input type="checkbox"/> 成都站点	test	18280473733	test	摩洛哥	2020-04-29 14:38:28	详细信息	修改		

显示第 1 到第 2 条记录, 总共 2 条记录

图 3-29 契约协议管理页面

契约协议详情页面如图 3-30 所示。

领养人:

年龄:

职业:

联系方式:

居住地址:

身份证信息:

领养站点:

领养宠物:

签约图片:

廖三思

22

程序员


16280473733

测试地址

510681199709305912

成都站点

丑丑



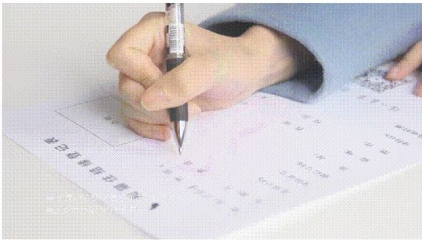


图 3-30 契约协议详情页面

获取数据列表代码如下：

```
@RequestMapping("/data")
@ResponseBody
public R data(Integer deptId, String master, Page<Contract> page) {
    QueryWrapper<Contract> param = new QueryWrapper<>();
    if(!StringUtils.isEmpty(master)) {
        param.like("master", master);
    }
    if(deptId != 1) {
        param.eq("dept_id", deptId);
    }
    param.orderByDesc("create_time");
    page = conService.page(page, param);
    return R.ok(page);
}
```

3.3.5.3 回访记录管理

回访记录管理页面如图 3-31 所示。

+ 新增

删除

搜索

搜索

C

<input type="checkbox"/>	回访站点	回访人	总计回访次数	信息是否更改	是否合格
<input type="checkbox"/>	成都站点	廖三思	1	否	是
<input type="checkbox"/>	成都站点	test	1	否	是

显示第 1 到第 2 条记录，总共 2 条记录

图 3-31 回访记录管理页面

获取数据列表代码如下：

```
@RequestMapping("/data")
@ResponseBody
public R data(Integer deptId, String master, Page<Revisit> page) {
    QueryWrapper<Revisit> param = new QueryWrapper<>();
    QueryWrapper<Contract> p = new QueryWrapper<>();
    if(!StringUtils.isEmpty(master)) {
        p.like("master", master);
        List<Contract> conlist = conService.list(p);
        if(!conlist.isEmpty()) {
            for (Contract con : conlist) {
                param.eq("contract_id", con.getContractId());
            }
        }
    } else {
        param.eq("contract_id", 0);
    }
    if(deptId != 1) {
        param.eq("dept_id", deptId);
    }
    param.orderByDesc("create_time");
    page = rvService.page(page, param);
    return R.ok(page);
}
```

3.3.6 文章管理

超级管理员和管理员可以查看、修改和删除所有代理领养宠物文章；可以查看、修改、删除和审核同城领养文章；可以增删改查养宠小知识文章；可以查看、修改、删除和审核拯救毛孩子文章。

站点管理员只有待领养宠物文章和拯救毛孩子文章的管理权限，不同于超级管理员和管理员，站点管理员负责发布待领养宠物文章，且只能选择所属部门中未被领养的宠物进行发布。文章的发布、修改、查看详情功能的实现同用户端文章发布、修改、查看详情功能实现原理相同，此处不再重复。

3.3.6.1 待领养宠物文章管理

待领养宠物文章管理页面如图 3-32 所示。

+ 新增		删除							搜索	搜索	C
发布站点	待领养宠物	标题	介绍信息	标签	发布时间	修改时间	操作				
成都站点	丑丑	是丑非丑	丑丑其实不丑，一半梨花一半白，5个月大的小家伙很亲人，快来带走它吧，狸爪盖章，余生多多指教！	猫咪	2020-04-30 11:03:38	2020-04-30 11:14:24	详细信息 修改				
成都站点	摩洛哥	猪鼻子奶牛猫	捡到一只2月大猪鼻子小奶牛，很有个性	猫咪	2020-04-10 19:06:39	2020-04-30 10:28:24	详细信息 修改				
成都站点	33	猪鼻	捡到一只2月大猪鼻子小奶牛，很有个性	猫咪	2020-04-10 19:06:39	2020-04-30 10:28:19	详细信息 修改				

显示第 1 到第 3 条记录，总共 3 条记录

图 3-32 待领养宠物文章管理页面

获取数据列表代码如下：

```
@RequestMapping("/data")
@ResponseBody
public R data(Integer deptId, String title, Page<Adoption> page) {
    QueryWrapper<Adoption> param = new QueryWrapper<>();
    if(deptId != 1) {
        param.eq("dept_id", deptId);
    }
    if(!StringUtils.isEmpty(title)) {
        param.like("title", title);
    }
    param.orderByDesc("update_time");
    page = adoptService.page(page, param);
    return R.ok(page);
}
```

3.3.6.2 同城领养文章管理

同城领养文章管理页面如图 3-33 所示。

审核

删除

搜索

搜索

<input type="checkbox"/>	发布人	城市	待领养宠物	标题	介绍信息	标签	发布时间	修改时间	状态	审核	操作
<input type="checkbox"/>	wxmm	成都市	澳大利亚	小猫领养	小猫你是不是有许多的问号???	小猫咪	2020-04-13 18:06:40	2020-04-30 16:36:42	待领养	未审核	<div><div>详细信息</div><div>修改</div></div>
<input type="checkbox"/>	柠檬小鱼鱼	成都市	旺仔小包子	旺仔小包子大放送~	爱哭爱笑爱闹 领养不吃亏		2020-04-17 18:21:44	2020-04-17 18:21:44	待领养	已审核	<div><div>详细信息</div><div>修改</div></div>
<input type="checkbox"/>	qse3	成都市	测试适应性	测试适应性	test	test	2020-04-14 23:53:46	2020-04-16 15:46:38	待领养	已审核	<div><div>详细信息</div><div>修改</div></div>
<input type="checkbox"/>	qse3	成都市	测试适应性	测试适应性	test	test	2020-04-14 23:41:47	2020-04-17 17:15:28	待领养	已审核	<div><div>详细信息</div><div>修改</div></div>
<input type="checkbox"/>	wxmm	成都市	测试	测试	测试	测试	2020-04-14 14:18:16	2020-04-14 14:21:44	待领养	已审核	<div><div>详细信息</div><div>修改</div></div>
<input type="checkbox"/>	wxmm	成都市	汤姆	汤姆和杰瑞	猫和老鼠	猫和老鼠	2020-04-13 20:19:29	2020-04-13 20:19:29	待领养	已审核	<div><div>详细信息</div><div>修改</div></div>

显示第 1 到第 6 条记录, 总共 6 条记录

图 3-33 同城领养文章管理页面

获取数据列表代码如下：

```
@RequestMapping("/data")
@ResponseBody
public R data(String title, Page<Cityadopt> page) {
    QueryWrapper<Cityadopt> param = new QueryWrapper<>();
    if(!StringUtils.isEmpty(title)) {
        param.like("title", title);
    }
    param.orderByAsc("verify");
    param.orderByDesc("create_time");
    page = cdService.page(page, param);
    return R.ok(page);
}
```

审核功能代码如下：

```
@RequestMapping("/accept/{cityadoptId}")
@ResponseBody
public R accept(@PathVariable Integer cityadoptId) {
    Cityadopt cd = cdService.getById(cityadoptId);
    cd.setVerify(1);
    cdService.updateById(cd);
    return R.ok();
}
```

3.3.6.3 养宠小知识文章管理

养宠小知识文章管理页面如图 3-34 所示。

+ 新增 删除		搜索 搜索				
<input type="checkbox"/>	标题	介绍信息	标签	发布时间	修改时间	操作
<input type="checkbox"/>	资深铲屎官告诉你养猫的小知识!	现在养猫的人越来越多,或许很多人还不知道该如何养猫,或许还有很多人想养猫对于猫却非常的迷茫,小编今天为大家来讲讲	喵星人	2020-04-30 16:19:16	2020-04-30 16:21:51	i 详细信息 修改
<input type="checkbox"/>	测试测试	测试测试	测试测试	2020-04-16 16:02:37	2020-04-17 12:38:55	i 详细信息 修改

显示第 1 到第 2 条记录, 总共 2 条记录

图 3-34 养宠小知识文章管理页面

获取数据列表代码如下:

```
@RequestMapping("/data")
@ResponseBody
public R data(String title, Page<Tips> page) {
    QueryWrapper<Tips> param = new QueryWrapper<>();
    if(!StringUtils.isEmpty(title)) {
        param.like("title", title);
    }
    param.orderByDesc("create_time");
    page = tipsService.page(page, param);
    return R.ok(page);
}
```

3.3.6.4 拯救毛孩子文章管理

拯救毛孩子文章管理页面如图 3-35 所示。

审核

删除

搜索

搜索

<input type="checkbox"/>	标题	发布人	城市	介绍信息	标签	发布时间	修改时间	状态	审核	操作
<input type="checkbox"/>	网红桥下的小可爱!	枚枚	成都市	网红桥墩上出现一只猫咪,长毛猫,看起来还很小,位置很难营救,需要大家的帮助。	救助困难	2020-04-30 16:09:29	2020-04-30 16:39:49	待救助	未审核	<div><div>详细信息</div><div>修改</div></div>
<input type="checkbox"/>	222	枚枚	成都市	222	222	2020-04-15 19:41:39	2020-04-17 11:56:55	救助中	已审核	<div><div>详细信息</div><div>修改</div></div>
<input type="checkbox"/>	111	枚枚	成都市	111	111	2020-04-15 19:39:08	2020-04-16 16:44:09	已救助	已审核	<div><div>详细信息</div><div>修改</div></div>

显示第 1 到第 3 条记录, 总共 3 条记录

图 3-35 拯救毛孩子文章管理页面

管理员需要根据实际救助情况点击状态列中的下拉按钮修改文章状态, 便于对救助活动进行监督。

获取数据列表代码如下:

```
@RequestMapping("/data")
@ResponseBody
public R data(String title, Integer deptId, Page<Rescue> page) {
    QueryWrapper<Rescue> param = new QueryWrapper<>();
    if(!StringUtils.isEmpty(title)) {
        param.like("title", title);
    }
    if(deptId != 1) {
        Dept dept = deptService.getById(deptId);
    }
}
```

```

        param.eq("province", dept.getPareaId());
        param.eq("city", dept.getCareaId());
    }
    param.orderByAsc("verify");
    param.orderByAsc("status");
    param.orderByDesc("create_time");
    page = rescueService.page(page, param);
    return R.ok(page);
}

```

审核功能代码如下：

```

@RequestMapping("/accept/{rescueId}")
@ResponseBody
public R accept(@PathVariable Integer rescueId) {
    Rescue rescue = rescueService.getById(rescueId);
    rescue.setVerify(1);
    rescueService.updateById(rescue);
    return R.ok();
}

```

状态修改功能代码如下：

```

@RequestMapping("/status")
@ResponseBody
public R status(Integer cityadoptId, Integer status) {
    Rescue rescue = rescueService.getById(cityadoptId);
    if(rescue.getVerify() == 0) {
        return R.error("审核中...");
    }
    else {
        rescue.setStatus(status);
        rescueService.updateById(rescue);
        return R.ok();
    }
}

```

4 系统测试

在项目开发到部署运行的整个过程中，系统测试是非常关键的一步，不仅可以通过测试检验系统功能、系统性能是否符合预期，也可以测试出系统是否存在没有注意到的 BUG。测试用例可以逻辑清晰并全面地测试系统的各项功能，因此设计出好的测试用例是非常关键的。

4.1 测试的目的及意义

在编码过程中，造成 BUG 的原因往往不是语法错误而是逻辑错误，而 IDE 工具是无法帮助编码人员检查出逻辑错误的。因此，测试的目的就在于检验程序中的功能是否符合设计的逻辑，是否有编码人员没有注意到的逻辑错误造成的 BUG。但是发现 BUG 并不是测试的唯一目的，当整个项目编码完成后，完整地使用项目的所用功能且没有发现错误是项目质量的重要保障，也是让编码人员找寻改进灵感的重要途径之一。

4.2 测试环境

- ◆ 操作系统：Windows 10
- ◆ JDK：JDK1.8.0_191
- ◆ Maven：apache-maven-3.6.3
- ◆ Web 服务器：Tomcat 8.5
- ◆ 浏览器：Google Chrome 81.0.4044.129（64 位）
- ◆ 数据库：MySQL5.7.26
- ◆ 服务器配置：1 核 2GB 1Mbps

4.3 测试用例设计

本系统采用的是黑盒测试，设计了 11 个测试用例，覆盖了用户端和管理端的主要功能，测试用例如下表所示。

表 4-1 测试用户端注册和登录

用例编号	001
测试内容	测试用户能否正常注册和登录
测试步骤	1、用户访问用户端登录页面 2、点击立即注册链接跳转到注册页面 3、输入用户名、密码、邮箱、电话等信息 4、点击注册按钮 5、注册成功后跳转到登录页面 6、输入注册的用户和密码 7、点击登录按钮
实际结果	用户可以正常注册和登录
测试结论	测试通过

表 4-2 测试查看文章列表及详情

用例编号	002
测试内容	测试用户能否正常显示文章列表并使用条件搜索，是否能查看文章详情
测试步骤	1、用户依次点击左侧领养宠物、同城领养、养宠小知识、拯救毛孩子链接 2、查看各功能文章列表 3、使用条件搜索，如选择不同站点、不同城市 4、搜索文章标题 5、点击文章题目查看文章详情
实际结果	用户可以正常查看相关信息
测试结论	测试通过

表 4-3 测试发布文章

用例编号	003
测试内容	测试用户能否正常发布文章
测试步骤	1、用户依次点击同城领养、拯救毛孩子的我要发布按钮 2、编辑文章内容 3、点击提交按钮
实际结果	用户可以正常发布文章
测试结论	测试通过

表 4-4 测试修改我的文章

用例编号	004
测试内容	测试用户能否正常修改自己发布的文章
测试步骤	1、用户依次点击我的文章菜单下的两个链接 2、修改文章状态 3、删除文章 4、点击文章标题 5、修改文章内容 6、点击提交按钮
实际结果	用户可以正常修改文章
测试结论	测试通过

表 4-5 测试注册志愿者

用例编号	005
测试内容	测试用户能否正常注册志愿者
测试步骤	1、用户点击拯救毛孩子链接 2、输入真实姓名、性别、联系方式、身份证号等信息 3、点击提交按钮
实际结果	用户可以正常注册为志愿者
测试结论	测试通过

表 4-6 测试管理端登录

用例编号	006
测试内容	测试管理员能否正常登录管理端
测试步骤	1、管理员输入用户名和密码 2、点击登录按钮
实际结果	管理员可以正常登录
测试结论	测试通过

表 4-7 测试系统管理

用例编号	007
测试内容	测试超级管理员能否正常使用系统管理模块
测试步骤	1、操作管理员信息 2、分配角色 3、操作角色信息 4、分配资源 5、操作资源信息
实际结果	超级管理员可以正常使用系统管理模块
测试结论	测试通过

表 4-8 测试系统管理

用例编号	008
测试内容	测试管理员能否正常使用站点管理模块
测试步骤	1、操作站点信息 2、操作物资信息 3、入库、补仓、申请、出库物资 4、操作物资记录信息 5、审核、驳回、确认收货物资记录
实际结果	管理员可以正常使用站点管理模块
测试结论	测试通过

表 4-9 测试用户管理

用例编号	009
测试内容	测试管理员能否正常使用用户管理模块
测试步骤	1、操作普通用户信息 2、操作志愿者信息
实际结果	管理员可以正常使用用户管理模块
测试结论	测试通过

表 4-10 测试宠物管理

用例编号	010
测试内容	测试管理员能否正常使用宠物管理模块
测试步骤	1、操作宠物信息 2、操作契约协议信息 3、操作回访记录信息
实际结果	管理员可以正常使用宠物管理模块
测试结论	测试通过

表 4-11 测试文章管理

用例编号	011
测试内容	测试管理员能否正常使用文章管理模块
测试步骤	1、操作待领养宠物文章信息 2、操作同城领养文章信息 3、审核同城领养文章 4、操作养宠小知识文章信息 5、操作拯救毛孩子文章信息 6、审核拯救毛孩子文章
实际结果	管理员可以正常使用文章管理模块
测试结论	测试通过

通过这 11 个测试用例，覆盖了本系统用户端和管理端从注册、登录到各个模块的所有功能，测试的结果是全部测试通过，表明本系统的所有功能均能正常使用，能完成相应的业务流程，达到了系统设计要求。

5 总结与展望

5.1 总结

本文对流浪猫犬救助领养平台的建设进行了需求、可行性和设计等方面的分析,深入探讨了系统所使用的技术及其在系统设计中的应用。

从设计目标来看,系统的各项功能基本达到了设计要求,普通用户能在该平台通过领养的方式领养到心仪的宠物,爱心人士可以通过该平台为救助流浪猫犬的事业出一份力,救助组织则可以使用本系统高效地开展工作。总的来说,本系统为流浪猫犬救助领养活动提供了较高的技术支持,也为该类型救助组织的宣传和后续发展奠定了很好的基础。

从技术手段来看,系统将多种技术合理地结合,弥补了单一技术的不足,实现了更多功能,既减轻了开发人员的工作量,也方便了系统的维护和二次开发。

系统基本上达到了设计的预期,但仍然有不足的地方,主要是以下方面:

(1) 系统的功能设计仍然不够成熟,还没有接受实践的检验,需要根据用户更为细致、准确的需求进一步完善。

(2) 系统运行时可能会面临各种各样的情况,如何应对这些状况是需要在开发时全面考虑的。

5.2 展望

流浪宠物的问题已经持续了很多年,近年来更是加剧恶化,虽然受到了社会的高度重视,但在短时间内仍然是无法有效解决的。许许多多爱心人士为了救助流浪宠物牺牲自己的时间、精力甚至金钱,不仅仅是因为可怜它们,还因为想要营造一个更美好的社会环境。但随着互联网时代的到来,以往的救助方式不再适合这个新时代,顺应时代的改变是必要的。本系统的完善和推广,必将在很大程度上这方面的问题加以有效解决。

参考文献

- [1] 朱九超,孙泉云,夏炉明,卢军.流浪宠物的成因、危害及其防控对策[J].上海畜牧兽医通讯,2015(03):47-49.
- [2] Kvac Martin,Hofmannova Lada,Ortega Ynes,Holubova Nikola,Horcickova Michaela,Kicia Marta,Hlaskova Lenka,Kvetonova Dana,Sak Bohumil,McEvoy John. Stray cats are more frequently infected with zoonotic protists than pet cats.[J]. Folia parasitologica,2017,64.
- [3] 康楠.拯救流浪动物路在何方[J].经济,2012(05):125-127.
- [4] 陈红岩,何琳,李萌.城市流浪动物——亟待安置的生命[J].生命世界,2008(12):22-25.
- [5] 何倩.救助流浪动物,他们在行动[J].温州瞭望,2009(13):70-72.
- [6] 李嘉雯,黄群山.城市流浪犬猫的成因、危害与对策[J].广东畜牧兽医科技,2012,37(04):37-39.
- [7] 张秋雨.流浪动物救助实践困境与路径优化——基于四川省宜宾市的实证分析[J].法制与社会,2017(15):188-190.
- [8] 邓志强,邓林强.Maven 在 Java 项目开发中的应用[J].电子元器件与信息技术,2019,3(05):1-4.
- [9] 陈甫.Bootstrap3 在 Java Web 项目中的应用[J].电脑编程技巧与维护,2014(17):27-28+32.
- [10] 翟剑锐.Spring 框架技术分析及应用研究[D].中国科学院大学(工程管理与信息技术学院),2013.
- [11] 温立辉.Spring 框架在模型层的应用及原理[J].福建电脑,2017,33(05):147-148.
- [12] 丁振凡.Spring 3.x 的事务处理机制的研究比较[J].微型机与应用,2012,31(10):4-6.
- [13] 葛萌,黄素萍,欧阳宏基.基于 Spring MVC 框架的 Java Web 应用[J].计算机与现代化,2018(08):97-101.
- [14] 荣艳冬.关于 Mybatis 持久层框架的应用研究[J].信息安全与技术,2015,6(12):86-88
- [15] 陈宇收,饶宏博,徐亮.基于 Shiro 的权限管理机制研究[J].电脑编程技巧与维护,2019(06):39-40.
- [16] 赵素萍.MD5 加密算法的改进及应用[J].现代计算机(专业版),2017(15):60-62.
- [17] 冯振兴.Ajax 技术在 Web 系统中的应用研究[D].北京林业大学,2008.
- [18] 魏鑫.Mybatis 逆向工程功能扩展实现[J].电脑编程技巧与维护,2019(11):38-41.
- [19] Wojciech Muła,Daniel Lemire. Base64 encoding and decoding at almost the speed of a memory copy[J]. Software: Practice and Experience,2020,50(2).
- [20] David Insa,Josep Silva,Sergio López. Using the DOM Tree for Content Extraction[J]. Electronic Proceedings in Theoretical Computer Science,2012,98(Proc. WWV 2012).

致 谢

时光匆匆而过，转眼间大学生涯即将结束，此时此刻，我满怀感激之情，向所有指导我、关心我、传授我知识的老师和学院领导表示由衷的感谢。

首先，我要向指导我、督促我的范敏导师表达诚挚的感谢，感谢范老师半年来从选题、开题、设计编码、论文写作、答辩等多个环节给予了我精心的指导，让我能够顺利完成我的毕业设计和论文。范老师以严谨的态度、严格的要求、负责的精神为我做榜样，不仅教授了我知识，也教授了我做人的道理。我深刻体会到了范老师的用心良苦，再次向范老师致以崇高的敬意和真挚的感谢。

其次，在大学四年里，我得到了学院许多老师和领导的关心、帮助和信任，感谢你们为学院创造了良好学习环境和学习氛围，才使得我能在这里获取到更多的知识和技能。

最后，感谢我的家人、同学和朋友，是你们始终支持我、鼓励我、帮助我，让我在求学过程中充满信心和动力。也要感谢我自己，感谢自己的坚持，感谢自己的热爱，感谢自己的汗水。感谢所有的论文评审专家和老师在百忙之中抽出时间评审论文。

卿三思

2020 年 4 月