浙江大学

ZHEJIANG UNIVERSITY


# 软件工程
# 设计报告

项目名称 动物领养系统

指导老师 尹建伟

学　院 计算机科学与技术学院

项目成员 吴岱阳 金俊一 王宇飞 鲁悦

王嘉成 濮学易 季海川 陶卓

# Contents

**1 简介** **1**

1.1 目的 . . . . . 1

1.2 范围 . . . . . 1

1.2.1 软件名称 . . . . . 1

1.2.2 软件功能 . . . . . 1

1.2.3 软件应用 . . . . . 2

**2 需求说明** **2**

2.1 功能需求描述 . . . . . 2

2.1.1 用户管理功能 . . . . . 2

2.1.2 动物信息管理功能 . . . . . 3

2.1.3 领养申请与审批功能 . . . . . 3

2.1.4 动物地图定位功能 . . . . . 4

2.1.5 动物识图搜索功能 . . . . . 4

2.1.6 领养指南功能 . . . . . 4

2.1.7 其他功能需求 . . . . . 4

2.2 接口需求描述 . . . . . 5

2.3 性能需求描述 . . . . . 5

2.3.1 系统响应时间 . . . . . 5

2.3.2 并发用户处理能力 . . . . . 5

2.3.3 数据处理与存储容量 . . . . . 5

2.3.4 系统可用性与兼容性 . . . . . 5

2.4 安全需求描述 . . . . . 6

2.5 环境需求描述 . . . . . 6

2.5.1 服务端 . . . . . 6

2.5.2 设备要求 . . . . . 8

2.5.3 前端依赖 . . . . . 8

2.5.4 客户端 . . . . . 8

**3 系统体系结构图** **9**

3.1 客户端 . . . . . 9

3.2 后端系统 . . . . . 10

3.3 外部服务 . . . . . 10

**4 数据库设计** **11**

4.1 ER 图 . . . . . 11

4.2 逻辑结构设计 . . . . . 11

4.3 物理结构设计 . . . . . 12

**5 关键过程** **14**

5.1 用户注册登录过程 . . . . . 14

5.1.1 用户注册过程 . . . . . 14

5.1.2 用户登录过程 . . . . . 14

5.2 动物信息管理过程 . . . . . 15

5.2.1 管理员编辑动物信息过程 . . . . . 15

5.2.2 普通用户提交新动物信息过程 . . . . . 15

5.2.3 用户查看动物信息过程 . . . . . 15

5.3 领养申请与审批过程 . . . . . 16

5.3.1 用户提交领养申请过程 . . . . . 16

5.3.2 管理员审核领养申请过程 . . . . . 16

5.4 动物地图定位过程 . . . . . 16

5.4.1 用户查看动物地图过程 . . . . . 16

5.4.2 用户更新动物位置过程 . . . . . 17

5.5 动物识图搜索过程 . . . . . 17

5.6 UI 数据处理与交互过程 . . . . . 18

5.7 数据访问与持久化过程 . . . . . 18

5.8 与外部服务的交互过程 . . . . . 18

**6 界面** **19**

6.1 用户端界面 . . . . . 19

6.1.1 登录注册界面 . . . . . 19

6.1.2 个人信息界面 . . . . . 19

6.1.3 动物信息筛选界面 . . . . . 20

6.1.4 动物信息浏览界面 . . . . . 20

6.1.5 领养申请界面 . . . . . 21

6.1.6 动物地图界面 . . . . . 22

6.2 管理员端界面 . . . . . 23
6.2.1 登录界面 . . . . . 23
6.2.2 动物信息管理界面 . . . . . 23
6.2.3 动物信息浏览界面 . . . . . 24
6.2.4 领养申请审批界面 . . . . . 24
6.2.5 动物地图管理界面 . . . . . 25
6.2.6 领养指南管理界面 . . . . . 26

**7 详细设计 26**
7.1 前端内部接口 . . . . . 26
7.1.1 基础信息模块 . . . . . 26
7.1.2 动物领养申请模块 . . . . . 28
7.1.3 动物地图模块 . . . . . 30
7.2 前后端间交互接口 . . . . . 30
7.2.1 axios 库 . . . . . 30
7.2.2 HttpHandler 接口 . . . . . 30
7.2.3 Express 依赖 . . . . . 32
7.3 后端内部接口 . . . . . 33
7.3.1 基础信息模块 . . . . . 33
7.3.2 动物领养模块 . . . . . 34

**8 可靠性及安全性设计 35**
8.1 系统可靠性设计 . . . . . 35
8.1.1 数据持久化与一致性保障 . . . . . 35
8.1.2 容错与故障恢复设计 . . . . . 35
8.1.3 系统监控与日志记录 . . . . . 35
8.2 系统安全性设计 . . . . . 36
8.2.1 身份认证与授权 . . . . . 36
8.2.2 数据传输安全 . . . . . 36
8.2.3 输入验证 . . . . . 36
8.2.4 安全审计与日志 . . . . . 36
8.3 系统出错设计 . . . . . 36
8.3.1 错误信息分类与提示 . . . . . 36
8.4 补救措施 . . . . . 37

8.5 系统维护设计 . . . . . 37

# 1 简介

## 1.1 目的

本文件旨在系统性地阐述动物领养系统的功能需求点、概要设计和详细设计方案。基于已完成的需求分析和项目计划，本文档将具体描述系统的架构、模块、核心功能、数据库结构、流程、界面与安全性设计。

本文档的主要目的是为系统的编码、测试、部署和后续维护阶段提供清晰、全面的技术指导和规范依据，确保开发团队能够准确理解并高效实现系统各项功能。

本文档的预期阅读者包括项目开发团队成员（如软件工程师、数据库设计师、测试工程师)、项目管理人员以及未来可能参与系统迭代和维护的相关人员。

## 1.2 范围

本设计报告详细定义了动物领养系统的各项设计细节，其主要内容涵盖了系统的功能实现方案、整体架构以及具体的技术细节。

### 1.2.1 软件名称

- 软件中文名称：动物领养系统
- 软件英文名称：ChaMaoMao

### 1.2.2 软件功能

动物领养系统主要着眼于以下功能：动物爱好者和用户的管理、动物信息的管理、领养处理、动物定位、动物识图搜索、领养教程和指南的发布等。在本动物领养平台上，用户可以注册账号，关注自己喜欢的动物，查看相关动物信息，提出领养请求，实现自己的养宠梦想；动保管理员负责审核用户信息，发布和审核领养信息，维护动物具体信息，发布领养教程和指南等。

下面将详细描述核心功能的系统设计：

**用户管理与认证**：包括普通用户和动保管理员两类用户的注册、登录、个人信息管理及权限控制等功能的设计。

**动物信息管理**：涉及待领养动物信息的发布、展示、搜索、修改和审核。详细信息包括动物的种类、品种、年龄、性别、健康状况（如疫苗、绝育情况)、性格特点、照片、当前位置等。普通用户可提交新发现的动物信息，管理员负责审核与维护。

**领养申请与审批**：规范用户提交领养申请、管理员审核申请（包括对申请者条件的评估)、反馈审批结果、更新动物领养状态等一系列流程的设计。

**动物地图与定位**：实现动物地理位置信息的展示与更新。用户可提交或更新已知动物的位置信息，管理员可对地图信息进行管理。

**动物图像识别**：提供通过上传动物图片来尝试识别动物种类或匹配系统中已有动物信息的功能设计，帮助用户快速了解遇到的动物。

**领养指南发布**：允许管理员发布和管理与动物领养相关的知识、教程、注意事项等内容，供用户查阅学习。

#### 1.2.3 软件应用

动物领养系统主要面向以下用户与应用场景：

**动物爱好者和潜在领养者**：以 18 到 35 岁的学生及年轻职场人士为主。本系统可以为广大关爱动物的人士，特别是浙江大学校内及周边社区的潜在领养者，提供一个便捷、可靠的平台来寻找合适的待领养动物，了解领养流程，并获取养宠知识。

**动物保护组织及志愿者**：为动物保护组织和志愿者提供一个信息化的管理工具，帮助他们更有效地管理待领养动物信息，处理领养申请，扩大救助动物的领养渠道，提升工作效率。

# 2 需求说明

## 2.1 功能需求描述

本节将阐释动物领养系统为满足用户需求所应该具备的核心功能。这些功能覆盖了从用户基础操作到动物信息管理、领养流程、地理位置服务、智能识别及知识普及等多个方面。

#### 2.1.1 用户管理功能

用户管理功能是系统的基础，保障用户能够安全、便捷地访问和使用系统资源，并根据其角色（普通用户或动保管理员）获得相应的操作权限。

**对于普通用户**：

用户注册：允许新用户通过提供有效的手机号和设置密码创建个人账号。

用户登录：已注册用户能够通过验证身份信息（手机号/账户名及密码）访问系统。

个人信息查看与修改：用户能够查看并修改其个人资料，如昵称、性别、年龄、联系地址、养宠经验等。

**对于动保管理员**：

管理员账户注册与登录：允许授权的管理员注册并登录管理后台。

用户信息审核与管理：管理员能够查看所有注册用户信息，并对用户提交的某些信息修改（如领养相关的关键信息）进行审核。

管理员个人信息管理：管理员能够查看并修改其自身的账户信息。

### 2.1.2 动物信息管理功能

动物信息管理是系统的核心之一，确保待领养动物的信息能够准确、全面地展示给用户，并由管理员进行有效维护。

**对于普通用户**：

查看动物详细信息：用户能够浏览系统中所有待领养动物的列表，并查看任一动物的详细资料。

提交新动物信息：户在发现新的流浪动物或有待领养动物时，可以向系统提交该动物的信息。

**对于动保管理员**：

发布与编辑动物信息：管理员能够创建、发布新的待领养动物信息，并对已存在的动物信息进行编辑和更新。

审核用户提交的动物信息：管理员能够审核普通用户提交的新动物信息，确保信息的真实性和准确性。

管理动物领养状态：管理员能够及时更新动物的领养状态。

### 2.1.3 领养申请与审批功能

此功能模块是连接领养者与待领养动物的关键桥梁，规范了从领养意向到最终领养确认的整个流程。

**对于普通用户**：

提交领养申请：用户在选定心仪的动物后，能够在线提交领养申请。

查看领养申请状态：用户能够实时追踪自己提交的领养申请的处理进度。

撤销领养申请：在申请被最终处理前，用户应有权撤销自己的领养申请。

**对于动保管理员**：

发布动物领养信息：管理员确认某动物开放领养时，能明确标示并使其对用户可见。

审核领养申请：管理员能够查看、评估所有用户提交的领养申请，并做出审批决定。

管理领养流程：管理员能够管理整个领养过程，包括与申请人的沟通和领养协议签署等环节的记录。

#### 2.1.4 动物地图定位功能

该功能通过地理位置信息，帮助用户了解动物的分布情况，并为寻找或救助特定动物提供便利。

**对于普通用户**：

更新动物位置：用户在发现系统中已记录的某个动物的新位置时，可以提交位置更新。

**对于动保管理员**：

查看与管理全局动物地图：管理员能够查看所有已记录动物的地理分布，并对位置信息进行管理和修正。

#### 2.1.5 动物识图搜索功能

此功能利用图像识别技术，帮助用户通过拍摄或上传动物照片来快速获取动物的相关信息。

**对于所有用户**：

拍照或上传图片识别：用户遇到不熟悉的动物时，可以通过拍摄照片或从本地相册上传动物图片，请求系统进行识别。

获取识别结果：系统能够分析上传的图片，并返回可能的动物种类、品种信息，或匹配系统中已存在的相似动物。

#### 2.1.6 领养指南功能

该功能旨在为用户提供专业、全面的动物领养及养护知识，提升领养成功率。

**对于普通用户**：

浏览和搜索领养指南：用户能够方便地查阅管理员发布的各类领养指南、教程、科普文章等。

**对于动保管理员**：

发布和管理领养指南：管理员能够创建、编辑、发布和删除领养指南及相关知识文章。也需要对指南内容进行有效的组织和分类，方便用户查找。

#### 2.1.7 其他功能需求

系统通知功能：系统应能通过适当方式向用户和管理员发送重要通知。

用户与管理员沟通功能：提供一个机制，允许普通用户就其领养申请或遇到的问题与动保管理员进行在线沟通。

## 2.2 接口需求描述

详见章节 7

## 2.3 性能需求描述

为了确保系统能够为用户提供流畅、高效的使用体验，并在不同负载情况下保持稳定运行，本节将详细阐述系统的各项关键性能指标和要求。这些性能需求是系统设计、开发和测试的重要依据。

### 2.3.1 系统响应时间

**常规操作**（单用户环境下）：

Web 页面（如首页、列表页、详情页）的加载时间：在良好网络条件下，应小于 1 秒

用户基本操作响应时间（如表单提交、信息修改、页面内导航跳转）：应小于 1 秒

**核心功能操作**（单用户环境下）：

动物信息搜索（从提交搜索条件到结果展示）：应小于 3 秒

动物地图加载与交互（如缩放、拖动、查看标记点）：应小于 3 秒

图像识别响应时间（从用户上传图片到系统返回初步识别结果）：应小于 10 秒

### 2.3.2 并发用户处理能力

系统应能稳定支持至少 500 个用户同时在线进行常规浏览、信息查询和提交操作，而不出现明显的性能下降或服务中断。

### 2.3.3 数据处理与存储容量

用户数据：至少支持 10000 条独立用户记录的存储与管理

动物数据：至少支持 10000 条独立的动物信息记录（包括其详细属性和多媒体资料）的存储与管理

关联业务数据：对于领养申请记录、动物位置信息记录、领养指南文章等其他类型的核心业务数据，每种类型应至少支持 10000 条记录的存储与高效检索。

### 2.3.4 系统可用性与兼容性

浏览器兼容性：系统前端界面应能在当前主流的 Web 浏览器上正确显示和正常运行，包括但不限于 Google Chrome、Mozilla Firefox、Microsoft Edge 等的最新稳定版本。

## 2.4 安全需求描述

**数据保密性**：

用户身份认证信息安全：用户的登录凭证（如手机号/账户名和密码）必须得到妥善保护，防止泄露。

个人敏感信息保护：用户的个人身份信息以及领养相关的隐私信息，应受到严格保护。

访问控制：严格控制不同角色用户对系统功能和数据的访问权限。

**数据完整性**：

防止未授权的数据修改：保护系统中的核心业务数据不被非法修改、插入或删除。

数据备份与恢复：系统应具备定期备份数据的能力，并制定有效的数据恢复计划，以应对意外数据丢失或损坏的情况。

输入数据校验：对用户输入的所有数据进行严格校验，防止恶意输入或格式错误的数据污染数据库。

**攻击防御**：

防范常见 Web 攻击：系统应能有效防范常见的 Web 应用程序漏洞和攻击手段，如 SQL 注入防护、文件上传安全。

密码安全策略：引导用户设置强密码，并防范密码暴力破解。

拒绝服务攻击防护：系统应具备一定的抵抗 DoS/DDoS 攻击的能力。

**操作审计与追溯**：

日志记录：对系统中的关键操作和安全事件进行日志记录。

日志保护与审计：确保日志的完整性和不可篡改性，并支持对日志进行查询和分析。

## 2.5 环境需求描述

### 2.5.1 服务端

node.js **版本**：20.15.1

**数据库**：mysql 8.0.41

**依赖要求**：

```
1  <dependencies>
2
3    <dependency>
4      <groupId>mysql</groupId>
```

```
5       <artifactId>mysql-connector-java</artifactId>
6       <version>8.0.31</version>
7     </dependency>
8
9     <dependency>
10      <groupId>org.opengauss</groupId>
11      <artifactId>opengauss-jdbc</artifactId>
12      <version>3.1.0</version>
13      <scope>provided</scope>
14    </dependency>
15
16    <dependency>
17      <groupId>com.microsoft.sqlserver</groupId>
18      <artifactId>mssql-jdbc</artifactId>
19      <version>12.2.0.jre8</version>
20    </dependency>
21
22    <dependency>
23      <groupId>junit</groupId>
24      <artifactId>junit</artifactId>
25      <version>4.13.2</version>
26      <scope>test</scope>
27    </dependency>
28
29    <dependency>
30      <groupId>org.projectlombok</groupId>
31      <artifactId>lombok</artifactId>
32      <version>1.18.24</version>
33      <scope>provided</scope>
34    </dependency>
35
36    <dependency>
37      <groupId>com.google.code.gson</groupId>
38      <artifactId>gson</artifactId>
39      <version>2.9.1</version>
40    </dependency>
41
42    <dependency>
```

```
43      <groupId>org.apache.commons</groupId>
44      <artifactId>commons-lang3</artifactId>
45      <version>3.7</version>
46    </dependency>
47
48  </dependencies>
```

### 2.5.2 设备要求

**浏览器要求**: Chrome、Edge、Firefox 等主流浏览器

Windows: 11

MacOS: 13.0.1

Ubuntu: 22.04

### 2.5.3 前端依赖

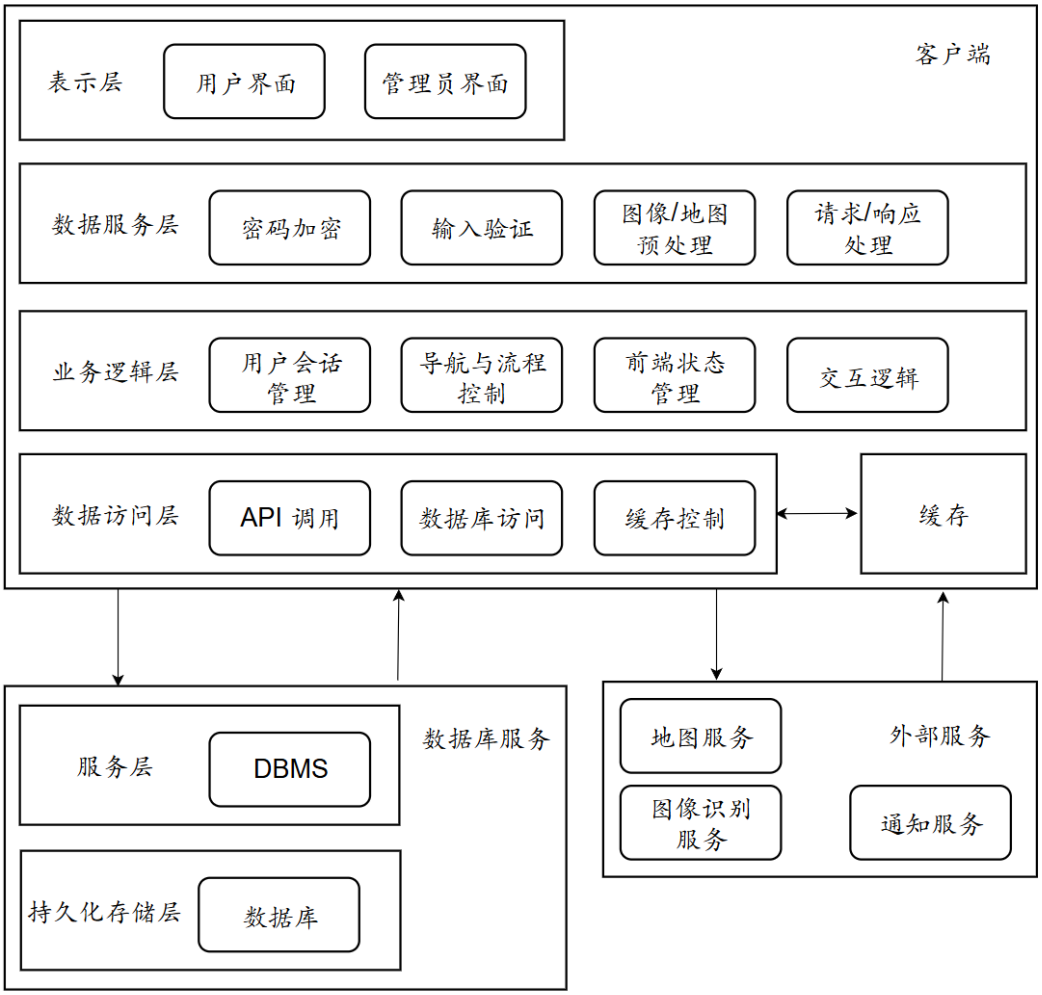
**依赖要求**:

```
1   "dependencies": {
2     "@element-plus/icons-vue": "^2.3.1",
3     "axios": "^1.7.2",
4     "element-plus": "2.6.0",
5     "pinia": "^2.1.7",
6     "vue": "^3.4.29",
7     "vue-router": "^4.2.5",
8     "vuex": "^4.1.0"
9   },
10  "devDependencies": {
11    "@vitejs/plugin-vue": "^5.0.5",
12    "vite": "^5.3.1"
13  }
```

### 2.5.4 客户端

Web 应用，要求使用 Chrome、Edge、Firefox 等主流浏览器访问

# 3 系统体系结构图




## 3.1 客户端

客户端采用分层结构，分为表示层、数据服务层、业务逻辑层和数据访问层。

**表示层**：负责直接与用户交互，展示信息和接收用户输入。组件有用户界面（包含登录页、个人信息页、动物信息页、领养申请页等）和管理员界面（包括用户管理、动物信息管理和领养审核页等）两个主要部分。

**数据服务层**：提供特定的数据处理、转换或封装服务，为表示层或业务逻辑层服务。它不包含核心业务规则，更多的是数据的准备和适配。在其中包括密码加密，在用户密码发送到后端前进行客户端加密；输入数据验证，对用户在表单中输入的数据进行前端校验和初步格式化；图像和地图的预处理，能用户上传动物图片前，可能进行压缩、裁剪或格式检查，也能从地图服务获取的数据或用户输入的位置数据，转换为地图展示库所需的格式；请求/响应处理，封装对后端 API 的调用，处理响应数据的初步解析和错误处理。

**业务逻辑层**：实现客户端的核心业务流程和规则。组件有用户会话管理，处理用户登录状态；导航与流程控制，管理页面跳转逻辑，多步骤表单的流程控制；交互逻辑处理，协调外部服务和界面操作；前端状态管理，管理应用的全局或局部状态。

**数据访问层**：负责客户端与后端数据源的通信。能进行 API 调用和数据库访问，还能控制缓存。

## 3.2 后端系统

在后端系统中，我们将其简化成一个数据库服务。

**服务层**：DBMS，如我们使用的 MySQL。

**持久化存储层**：数据库本身，存储所有应用数据。

## 3.3 外部服务

我们的系统会依赖部分第三方服务，如地图、图像识别和通知服务。

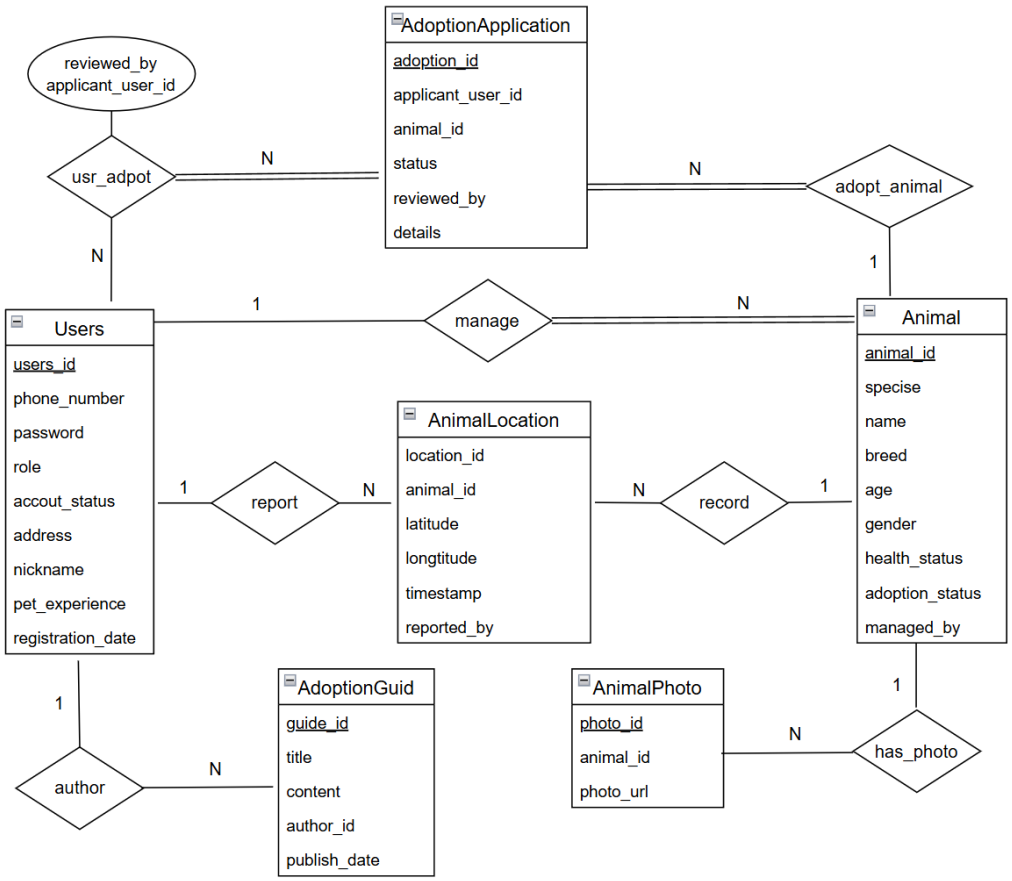
**地图服务**：如高德地图 API 等，用于显示地图、地理编码和位置服务。

**图像识别服务**：第三方 AI 服务，用于识别用户上传的动物图片。

**通知服务**：邮件推送服务，用于发送系统通知。

# 4 数据库设计

## 4.1 ER 图




## 4.2 逻辑结构设计

**用户表**

主键是 `users_id`，是用户唯一的标识符。

`Users(users_id,phone_number,password,role,accout_status,nickname,address,pet_experience,registration_date)`

**动物表**

主键是 `animal_id`，是动物唯一的标识符。外键是 `managed_by`，指向用户表中的 `users_id`，表示该动物由哪个用户管理。

`Animals(animal_id,name,specise,breed,age,gender,health_status,adoption_status,managed_by)`

**动物照片表**

主键是 photo_id，是照片唯一的标识符。外键是 animal_id，指向动物表中的 animal_id，表示该照片属于哪个动物。

AnimalPhotos(photo_id,animal_id,photo_url)

**领养申请表**

主键是 adoption_id，是领养申请唯一的标识符。外键有 animal_id，指向动物表中的 animal_id，表示该申请对应的动物；applicant_user_id，指向用户表中的 users_id，表示申请人；reviewed_by，指向用户表中的 users_id，表示审核人。

AdoptionApplications(adoption_id,animal_id,applicant_user_id,status, reviewed_by,details)

**领养指南表**

主键是 guide_id，是领养指南唯一的标识符。外键是 author_id，指向用户表中的 users_id，表示该指南的作者。

AdoptionGuides(guide_id,title,content,author_id,publish_date)

**动物位置记录表**

主键是 location_id，是位置记录唯一的标识符。外键有 animal_id，指向动物表中的 animal_id，表示该位置记录对应的动物；reported_by，指向用户表中的 users_id，表示报告该位置记录的用户。

AnimalLocations(location_id,animal_id,latitude,longtitude,timestamp, reported_by)

## 4.3 物理结构设计

**用户表**

| 字段名 | 数据类型 | 主键 | 外键 | 备注 |
|---|---|---|---|---|
| users_id | int | 是 | 否 | 用户唯一标识符，自增 |
| phone_number | varchar(20) | 否 | 否 | 用户手机号，用于登录，有唯一性 |
| password | varchar(255) | 否 | 否 | 存储用户密码的哈希值 |
| role | varchar(20) | 否 | 否 | 用户角色 |
| accout_status | varchar(20) | 否 | 否 | 账户状态 |
| nickname | varchar(50) | 否 | 否 | 用户昵称 |
| address | varchar(255) | 否 | 否 | 用户地址 |
| pet_experience | text | 否 | 否 | 用户养宠经验描述 |
| registration_date | timestamp | 否 | 否 | 用户注册日期时间 |

**动物表**

| 字段名 | 数据类型 | 主键 | 外键 | 备注 |
|---|---|---|---|---|
| animal_id | int | 是 | 否 | 动物唯一标识符, 自增 |
| name | varchar(100) | 否 | 否 | 动物名称 |
| specise | varchar(50) | 否 | 否 | 动物种类 |
| breed | varchar(50) | 否 | 否 | 动物具体品种 |
| age | varchar(50) | 否 | 否 | 动物年龄 |
| gender | varchar(10) | 否 | 否 | 动物性别 |
| health_status | text | 否 | 否 | 动物健康状况描述 |
| adoption_status | varchar(20) | 否 | 否 | 领养状态 |
| managed_by | int | 否 | 是 | 管理该动物的用户 |

**动物照片表**

| 字段名 | 数据类型 | 主键 | 外键 | 备注 |
|---|---|---|---|---|
| photo_id | int | 是 | 否 | 照片唯一标识符, 自增 |
| animal_id | int | 否 | 是 | 该照片所属的动物 |
| photo_url | varchar(255) | 否 | 否 | 照片的地址 |

**领养申请表**

| 字段名 | 数据类型 | 主键 | 外键 | 备注 |
|---|---|---|---|---|
| adoption_id | int | 是 | 否 | 领养申请唯一标识符, 自增 |
| animal_id | int | 否 | 是 | 申请领养的动物 |
| applicant_user_id | int | 否 | 是 | 提交申请的用户 |
| status | varchar(20) | 否 | 否 | 申请状态 |
| reviewed_by | int | 否 | 是 | 审核该申请的管理员用户 |
| details | text | 否 | 否 | 申请相关的详细信息 |

**领养指南表**

| 字段名 | 数据类型 | 主键 | 外键 | 备注 |
|---|---|---|---|---|
| guide_id | int | 是 | 否 | 领养指南唯一标识符, 自增 |
| title | varchar(255) | 否 | 否 | 指南标题 |

接下页

| 字段名 | 数据类型 | 主键 | 外键 | 备注 |
|---|---|---|---|---|
| content | text | 否 | 否 | 指南详细内容 |
| author_id | int | 否 | 是 | 指南作者的用户 |
| publish_date | timestamp | 否 | 否 | 指南发布日期时间 |

**动物位置记录表**

| 字段名 | 数据类型 | 主键 | 外键 | 备注 |
|---|---|---|---|---|
| location_id | int | 是 | 否 | 位置记录唯一标识符, 自增 |
| animal_id | int | 否 | 是 | 该位置记录对应的动物 |
| latitude | decimal(9,6) | 否 | 否 | 地理纬度 |
| longtitude | decimal(9,6) | 否 | 否 | 地理经度 |
| timestamp | timestamp | 否 | 否 | 位置记录的时间戳 |
| reported_by | int | 否 | 是 | 报告该位置记录的用户 |

# 5 关键过程

## 5.1 用户注册登录过程

### 5.1.1 用户注册过程

触发条件：用户在未登录的情况下，点击界面的注册选项。主要步骤如下：

(1). 系统导航至用户注册界面

(2). 用户在注册表单中输入必要信息，如手机号、密码、确认密码

(3). 进行前端校验，确保输入信息的有效性

(4). 提交注册请求，客户端将注册信息封装，发送后端

(5). 后端进行处理，判断手机号是否被注册登数据校验，再将密码进行安全的处理，最后数据持久化

(6). 后端向客户端返回注册结果，成功返回消息，引导至登录界面

### 5.1.2 用户登录过程

触发条件：用户在未登录的情况下，点击界面的登录选项。主要步骤如下：

(1). 系统导航至用户登录界面

(2). 用户在登录表单中输入必要信息，如手机号、密码

(3). 进行前端校验，确保输入信息的有效性

(4). 提交登录请求，客户端将登录信息封装，发送后端

(5). 后端进行处理，进行用户查找、密码校验、更新登录状态等

(6). 后端向客户端返回登录结果，引导至个人首页

## 5.2 动物信息管理过程

### 5.2.1 管理员编辑动物信息过程

触发条件：管理员在后台管理界面选择创建新动物信息或编辑现有动物信息。主要步骤如下：

(1). 系统展示一个空的动物信息表单，或根据指定的动物加载现有数据信息

(2). 管理员填写或修改动物的各项信息

(3). 进行前端校验，对必填项、数据格式等进行校验

(4). 提交创建、修改申请，客户端将完整的动物信息封装，发送后端

(5). 后端进行处理，验证权限和数据校验，再进行数据创建与更新，记录操作日志

(6). 服务端返回操作结果

### 5.2.2 普通用户提交新动物信息过程

与管理员创建动物信息过程类似，但是触发者为普通用户。过程有以下区别：

(1). 后端的处理不会直接修改和更新动物信息，而是先临时保存，待管理员进行审核编辑

(2). 管理员审核动物信息，批准和修改后，才会将其正式添加到系统中

### 5.2.3 用户查看动物信息过程

触发条件：用户在动物列表页点击某个动物，或通过搜索/识图结果访问动物详情。主要步骤如下：

(1). 系统展示动物信息，由用户自由点击查看

(2). 客户端根据用户点击或搜索结果的动物 ID，向后端请求动物信息

(3). 后端进行处理，查询动物信息表，返回动物的详细信息

(4). 客户端接收数据，展示动物的详细信息

## 5.3 领养申请与审批过程

### 5.3.1 用户提交领养申请过程

触发条件：用户在动物详情页点击申请领养按钮。主要步骤如下：

(1). 系统导航至该动物的领养申请表单页面，预填部分用户信息

(2). 用户根据提示填写领养申请的详细信息

(3). 进行前端校验，确保输入信息的有效性

(4). 提交申请，客户端将申请信息封装，发送后端

(5). 后端进行处理，验证用户身份，检查动物是否可领养，记录申请信息，再通知管理员进行审核

(6). 后端向客户端返回申请结果，引导用户查看申请状态

### 5.3.2 管理员审核领养申请过程

触发条件：管理员在后台管理界面查看待审核的领养申请。主要步骤如下：

(1). 系统提示待审核的领养申请列表，管理员选择某个申请进行审核

(2). 系统展示该申请的详细信息

(3). 管理员查看申请信息，决定是否批准或拒绝

(4). 提交审批结果，客户端将审核结果封装，发送后端

(5). 后端进行处理，验证权限，更新申请状态，记录操作日志

(6). 服务端向用户返回申请结果

## 5.4 动物地图定位过程

### 5.4.1 用户查看动物地图过程

触发条件：管理员访问系统的动物地图功能模块。主要步骤如下：

(1). 客户端向后端 API 请求在特定区域内或所有动物的位置数据

(2). 后端从动物位置表查询最新的、有效的动物位置记录

(3). 后端将动物的地理坐标列表返回给客户端

(4). 前端展示，客户端初始化地图组件，将从后端获取的位置渲染为标记点，允许用户进行地图交互和查看动物信息

### 5.4.2 用户更新动物位置过程

触发条件：用户在动物详情页点击更新位置按钮。主要步骤如下：

(1). 用户选择点击更新某个具体动物的位置，系统导航至该动物的更新位置表单页面

(2). 用户在地图上通过手动标点、设备 GPS| 定位或输入地址等方式确定动物位置

(3). 进行前端校验，判断位置信息是否有效

(4). 客户端将位置信息发送至后端

(5). 后端进行数据校验，通知管理员进行审核，待审核后更新主表位置信息，进行数据持久化

(6). 后端返回操作结果

## 5.5 动物识图搜索过程

触发条件：用户在系统中选择使用动物识图功能。主要步骤如下：

(1). 用户通过界面提供的入口，拍摄一张动物照片或从设备相册中选择一张图片上传

(2). 客户端可能对图片进行预处理

(3). 客户端将图片文件上传至后端服务器的特定接口

(4). 后端将图片转发给第三方图像识别服务 API

(5). 图像识别服务返回识别结果

(6). 后端根据识别出的动物特征，查询动物信息表，返回匹配的动物列表

(7). 后端将综合的识别结果返回给客户端

(8). 客户端向用户展示识别出的动物信息，并列出系统内找到的相似动物的链接或简要信息

## 5.6 UI 数据处理与交互过程

此过程概括了用户界面层在接受用户输入，展示数据以及与用户交互时的通用处理逻辑。关键方面如下:

- 表单处理与验证: 客户端对所有用户输入的表单数据进行即时校验，提供友好的错误提示
- 数据展示与渲染: 从后端获取的数据需要有效地进行渲染和展示
- 用户反馈与状态管理: 对用户的交互操作，进行即时反馈
- 封装请求和处理错误: 统一封装对后端的请求，需要有响应与处理各种错误情况

## 5.7 数据访问与持久化过程

此过程概括了后端应用服务器在与数据库进行交互以实现数据增删改查时的通用逻辑。关键方面如下:

- 进行数据库连接，获取和释放数据库连接
- 对事物管理时，需要保证原子性的操作确保数据一致性
- 进行数据校验和转换
- 记录数据库的操作处理或错误日志

## 5.8 与外部服务的交互过程

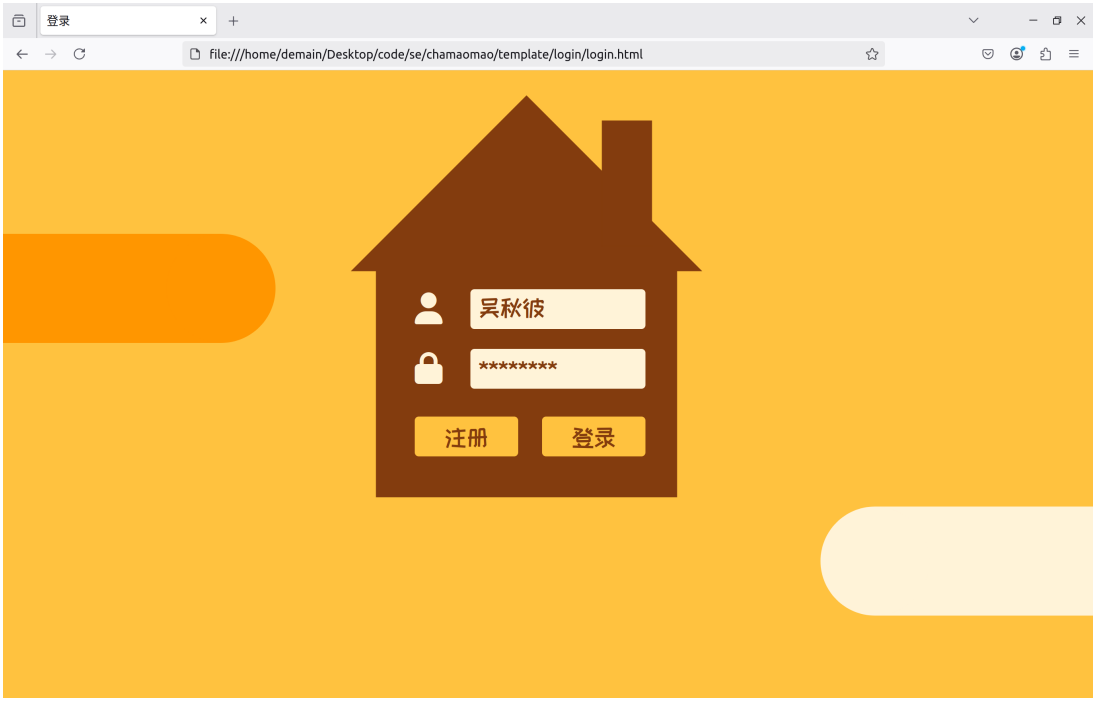
此过程概括了系统与第三方服务（如地图服务、图像识别服务）进行交互的通用模式。关键方面如下:

- 安全地存储和使用调用第三方 API 所需的密钥或凭证
- 处理格式、参数、头部信息对应的请求构建与发送
- 能够解析第三方服务返回的数据
- 将从第三方服务获取的数据转换为系统内部所需的格式

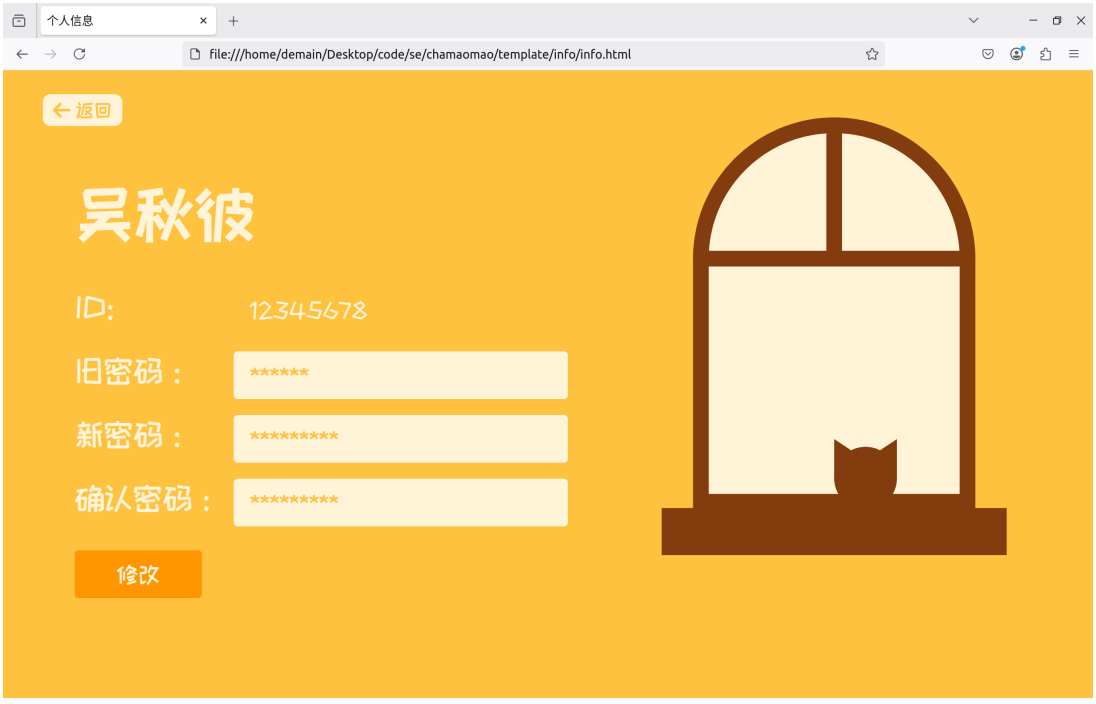
# 6 界面

## 6.1 用户端界面

### 6.1.1 登录注册界面



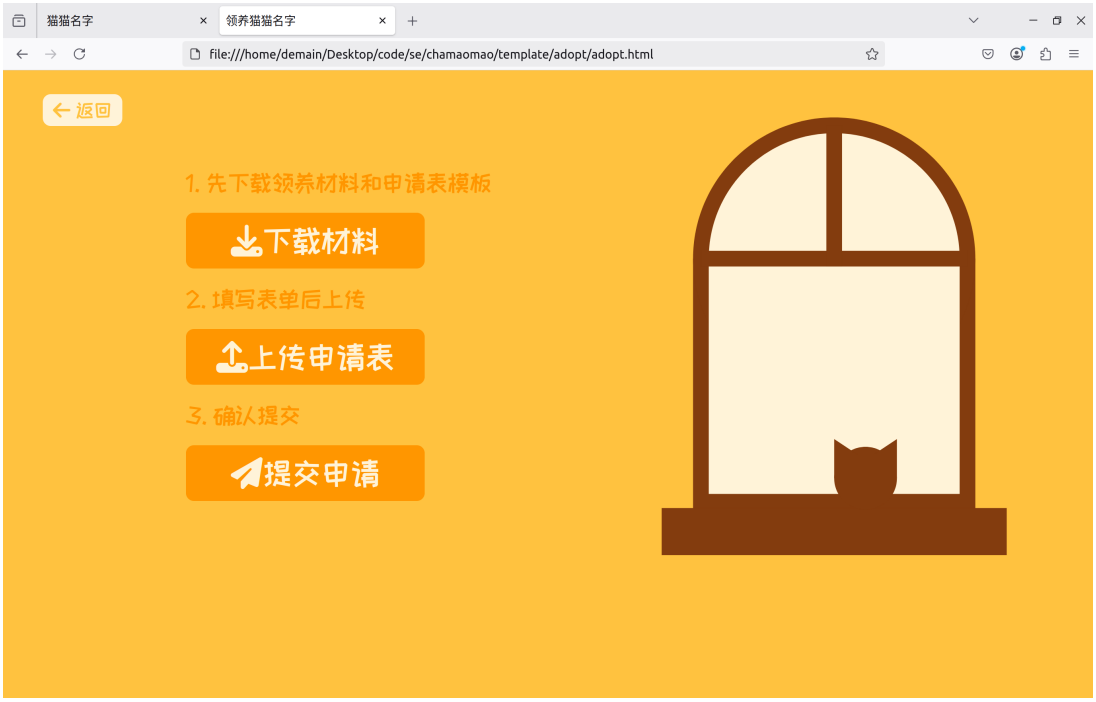

### 6.1.2 个人信息界面

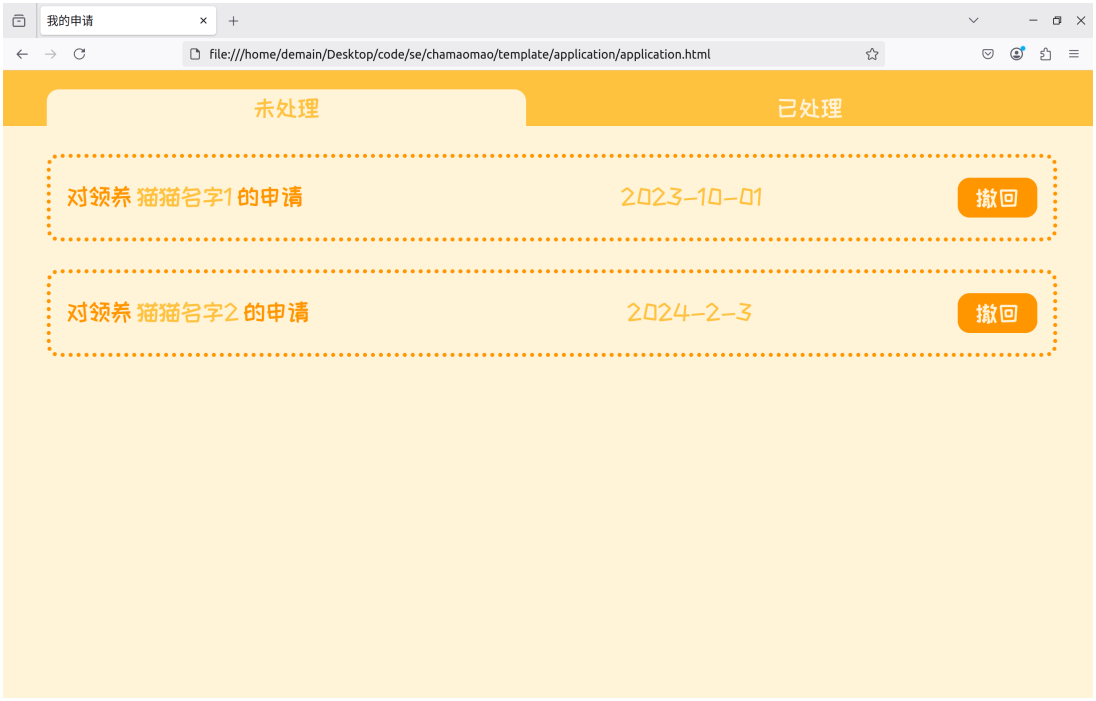


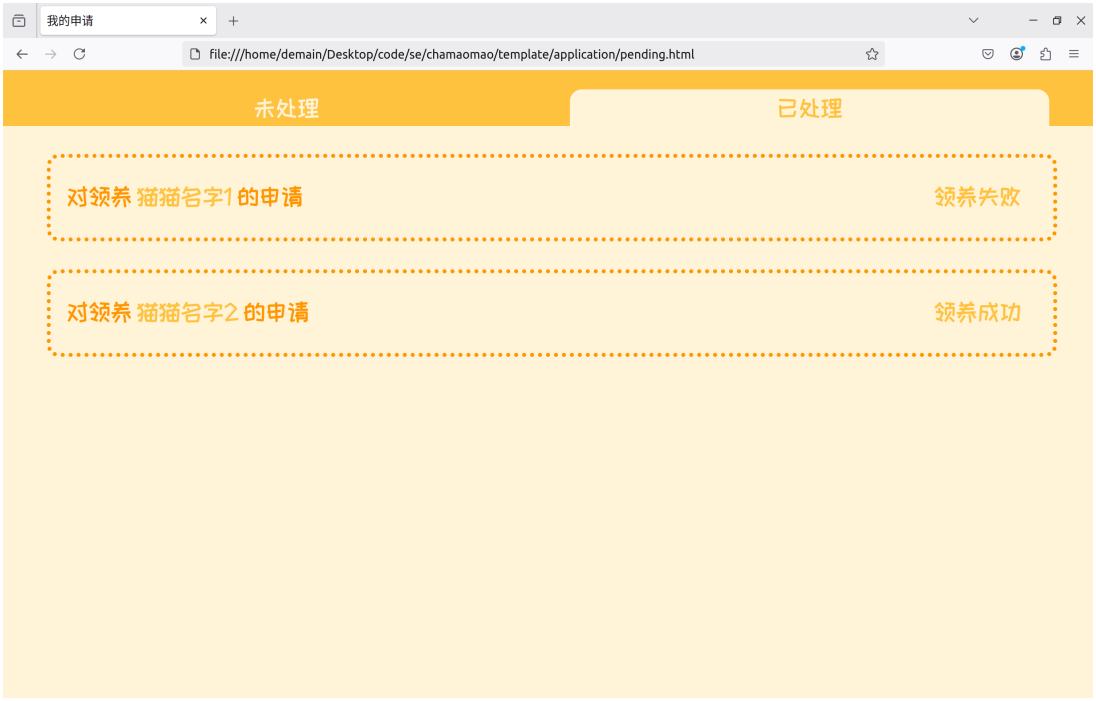
### 6.1.3 动物信息筛选界面


### 6.1.4 动物信息浏览界面

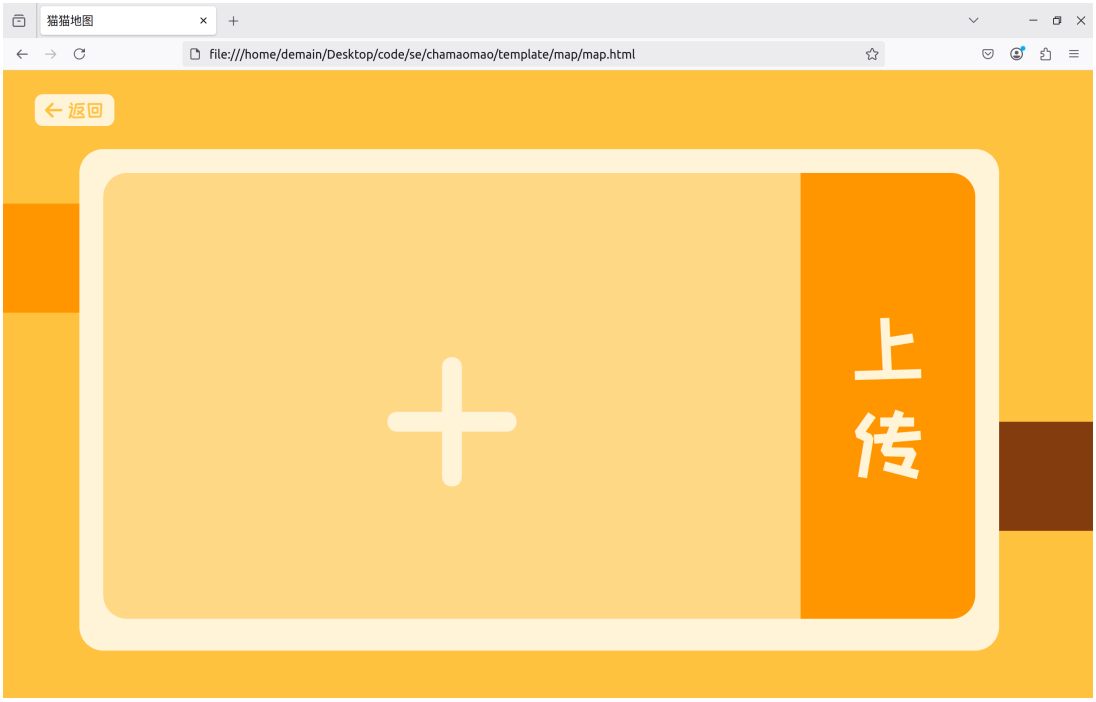
### 6.1.5 领养申请界面


猫猫名字
领养猫猫名字
file:///home/demain/Desktop/code/se/chamaomao/template/adopt/adopt.html
返回
1. 先下载领养材料和申请表模板
下载材料
2. 填写表单后上传
上传申请表
3. 确认提交
提交申请


我的申请
file:///home/demain/Desktop/code/se/chamaomao/template/application/application.html
未处理
已处理
对领养 猫猫名字1 的申请
2023-10-01
撤回
对领养 猫猫名字2 的申请
2024-2-3
撤回

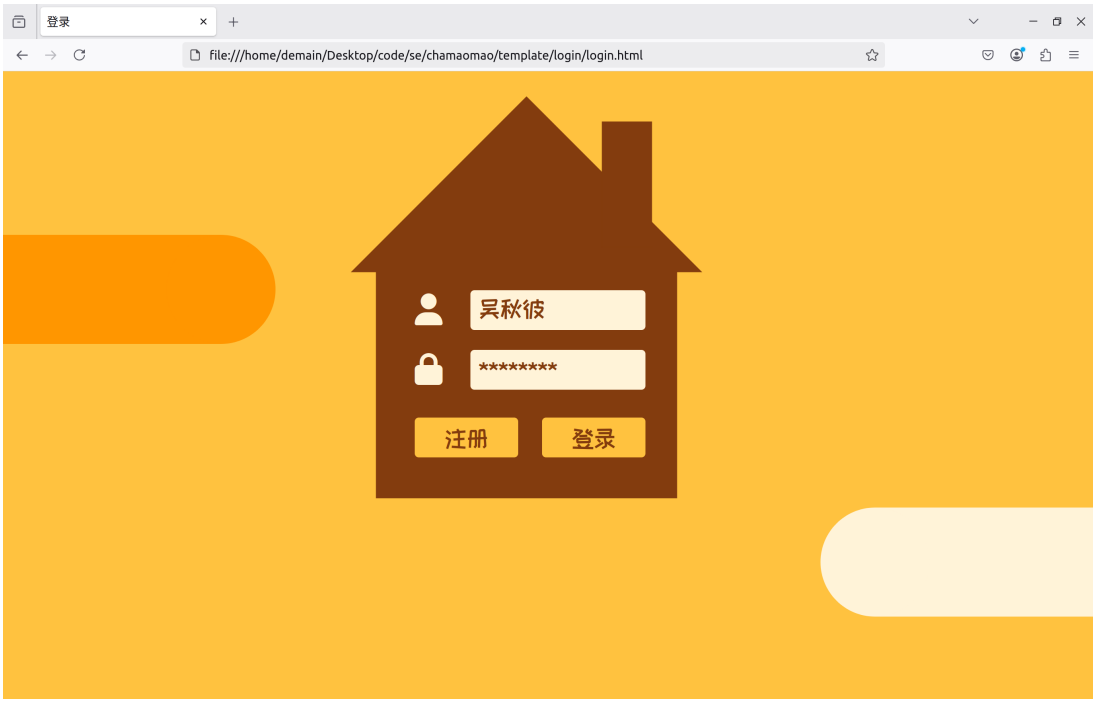

我的申请
file:///home/demain/Desktop/code/se/chamaomao/template/application/pending.html
未处理
已处理
对领养 猫猫名字1 的申请
领养失败
对领养 猫猫名字2 的申请
领养成功

### 6.1.6 动物地图界面


猫猫地图
file:///home/demain/Desktop/code/se/chamaomao/template/map/map.html
返回
上传

## 6.2 管理员端界面

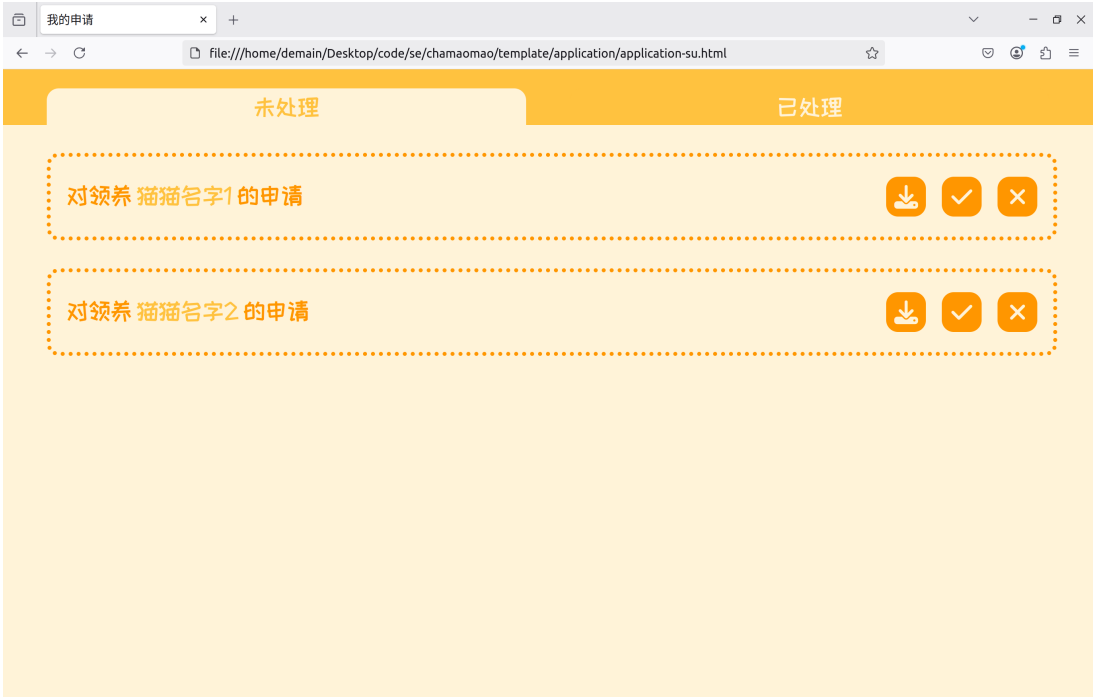
### 6.2.1 登录界面

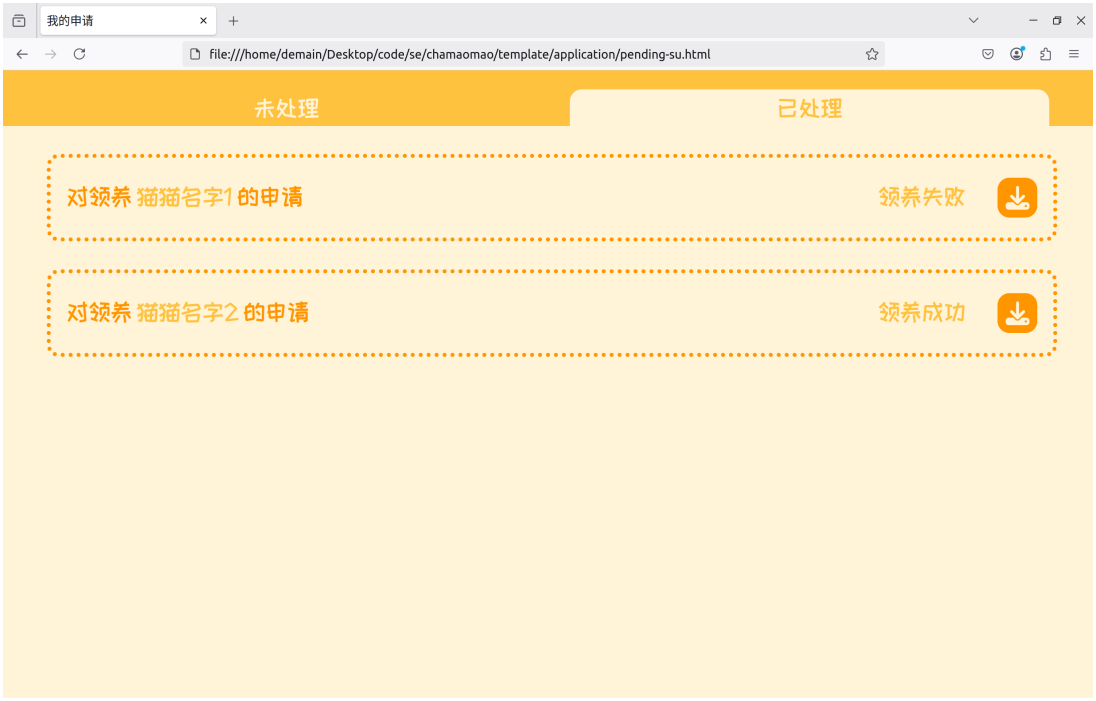



### 6.2.2 动物信息管理界面

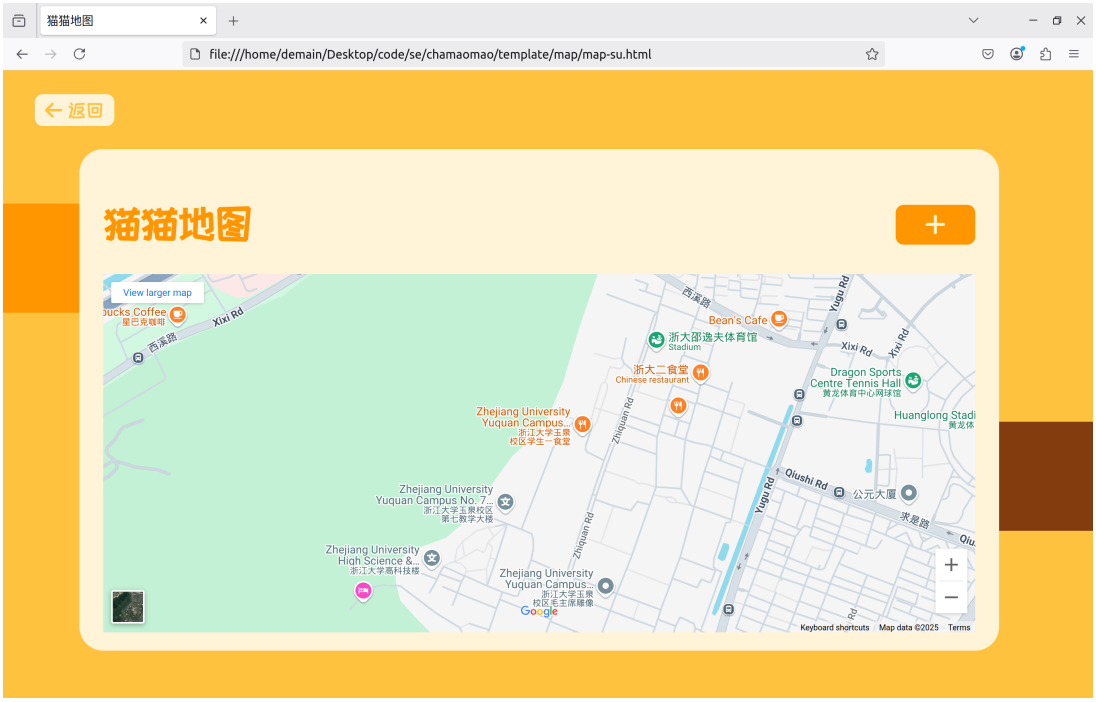
### 6.2.3 动物信息浏览界面


### 6.2.4 领养申请审批界面

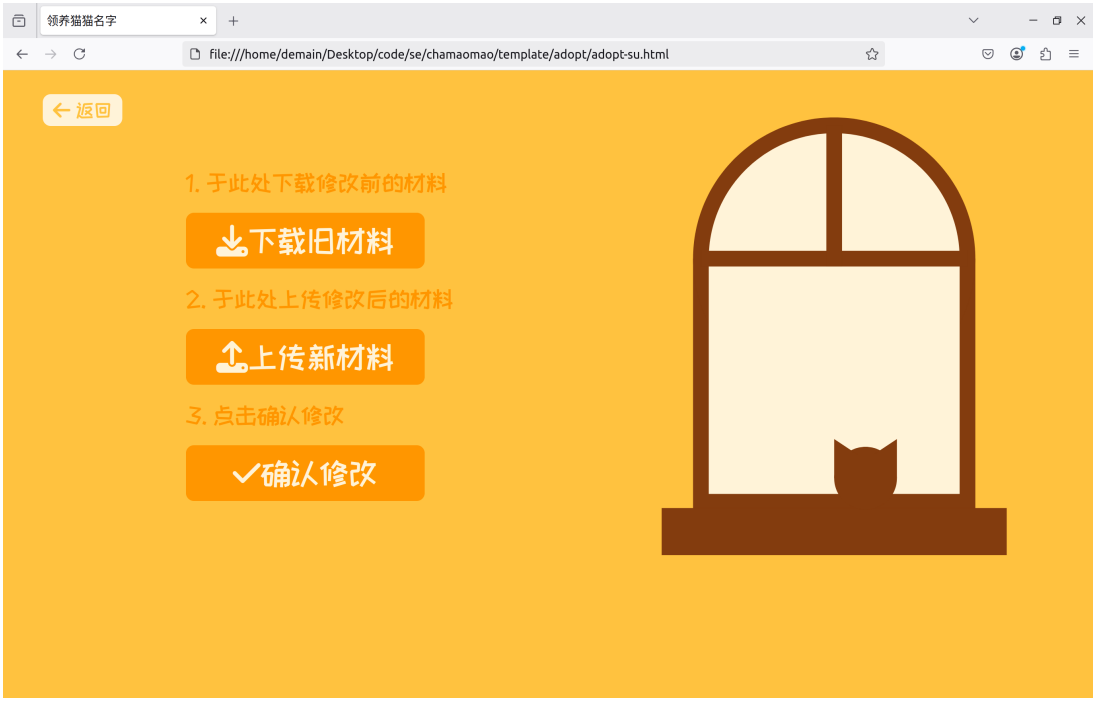





### 6.2.5 动物地图管理界面



#### 6.2.6 领养指南管理界面




# 7 详细设计

本章主要介绍项目前端、后端、以及前后端间定义的各种接口

## 7.1 前端内部接口

主要涉及 Vue 文件中 `script` 块的 `methods` 部分定义的函数

### 7.1.1 基础信息模块

- `Validate`
  - **接口描述：** 用于校验输入账号或密码的格式是否正确
  - **实现思路：** 包括非法字符的识别、长度检测等过程
- `LoginRequest`
  - **接口描述：** 处理登录请求，如果格式不符或者密码错误，给出提示；登陆成功后跳转到系统首页
  - **实现思路：** 先调用 `Validate` 接口校验账号密码格式，校验通过后进行加密，再向后端发送 `POST` 请求；根据后端传回的信息，判断是否登录成功；登陆成功后，判断是普通用户还是管理员，跳转到对应首页

- RegisterRequest
  - **接口描述：**处理注册请求，如果格式不符，给出提示；注册成功后跳转回登录界面
  - **实现思路：**先调用 Validate 接口校验密码格式，校验通过后进行加密，再向后端发送 POST 请求；根据后端传回的信息，判断是否注册成功；注册成功后，跳转回登录界面
- PersonalInfoQuery
  - **接口描述：**获取个人信息用于显示
  - **实现思路：**向后端发送 GET 请求；根据后端传回的数据，将个人信息存到 PersonalInfo 结构体中
- ConfirmPersonalInfoModify
  - **接口描述：**将修改后的个人信息同步到后端数据库，如果修改成功，给出提示
  - **实现思路：**向后端发送 POST 请求，传递 PersonalInfo 结构体；根据后端传回的信息，判断是否修改成功，进行相应显示
- QueryUsers
  - **接口描述：**供管理员查询所有普通用户的个人信息，显示在用户管理界面
  - **实现思路：**向后端发送 GET 请求；得到响应后，将 this.users 清空，创建一个临时变量 users 来获取响应负载，将其中的每个元素加到 this.users 中，页面更新后便会进行显示。
- QueryAnimals
  - **接口描述：**供用户查询动物详细信息
  - **实现思路：**向后端发送 POST 请求，传递 ToQuery 结构体，内部包含动物品种、姓名等查询条件；得到响应后，将 this.animals 清空，创建一个临时变量 animals 来获取响应负载，将其中的每个元素加到 this.animals 中，页面更新后便会进行显示。
- ImageFileValidate
  - **接口描述：**图片上传前，检查类型和大小等
  - **实现思路：**自定义要求进行检查，检查通过后，保存在 ImageFile 对象中，暂时位于浏览器内存中，后续可以用于构造 AnimalInfo 对象，发往后端

- AnimalInfoUpload
  - **接口描述：**新建动物，上传动物信息，待管理员审核通过后再加入数据库
  - **实现思路：**向后端发送 POST 请求，传递 AnimalInfo 结构体；根据后端传回的信息，判断是否上传成功，进行相应显示
- AnimalInfoModify
  - **接口描述：**管理员修改动物信息
  - **实现思路：**向后端发送 POST 请求，传递 AnimalInfo 结构体；根据后端传回的信息，判断是否修改成功，进行相应显示
- AnimalInfoDelete
  - **接口描述：**管理员删除动物信息
  - **实现思路：**向后端发送 DELETE 请求，指明动物 id；根据后端传回的信息，判断是否删除成功，进行相应显示

#### 7.1.2 动物领养申请模块

- FormDownload
  - **接口描述：**用户下载领养申请表
  - **实现思路：**向后端发送 GET 请求，端口号为领养申请表文件服务器对应的端口号
- FormUpload
  - **接口描述：**点击“提交申请”按钮后，用户上传填好的领养申请表
  - **实现思路：**向后端发送 POST 请求，端口号为领养申请表文件服务器对应的端口号，传递 FormData 结构体；根据后端传回的信息，判断是否上传成功
- AdoptionApply
  - **接口描述：**添加一条领养申请
  - **实现思路：**向后端发送 POST 请求，传递 AdoptionApplyInfo 结构体
- QueryAdoptionApply
  - **接口描述：**查询领养记录

  - **实现思路：**向后端发送 GET 请求，附带参数，表明是用户还是管理员的请求；得到响应后，将 this.adoptions 清空，创建一个临时变量 adoptions 来获取响应负载，将其中的每个元素加到 this.adoptions 中，页面更新后便会进行显示。

- ReviewAdoptionApply
  - **接口描述：**管理员审核用户的领养申请
  - **实现思路：**向后端发送 POST 请求，传递 ReviewOpinion 结构体，后端可以据此修改数据库中的审核记录

- AdoptionPost
  - **接口描述：**管理员发布领养信息，同步到数据库中
  - **实现思路：**向后端发送 POST 请求，传递 AdoptionPostInfo 结构体；根据后端传回的信息，判断是否发布成功

- QueryAdoptionPost
  - **接口描述：**查询所有已发布的领养信息，供用户挑选
  - **实现思路：**向后端发送 GET 请求；得到响应后，将 this.adoptions 清空，创建一个临时变量 adoptions 来获取响应负载，将其中的每个元素加到 this.adoptions 中，页面更新后便会进行显示。

- DeleteAdoptionPost
  - **接口描述：**管理员删除领养信息，同步到数据库中
  - **实现思路：**向后端发送 DELETE 请求，传递参数 adoption_post_id；根据后端传回的信息，判断是否删除成功

- GuideDownload
  - **接口描述：**下载领养指南
  - **实现思路：**向后端发送 GET 请求，端口号为领养指南文件服务器对应的端口号

- GuideUpload
  - **接口描述：**点击“确认修改”按钮后，上传修改好的领养指南
  - **实现思路：**向后端发送 POST 请求，端口号为领养指南文件服务器对应的端口号，传递 GuideData 结构体；根据后端传回的信息，判断是否上传成功

### 7.1.3 动物地图模块

- MapShow
  - **接口描述：** 显示动物地图
  - **实现思路：** 主要利用 AMapLoader 库调用高德地图提供的 API 实现。先使用申请好的 Web 端开发者 Key，调用 AMapLoader.load() 方法载入地图；再设置好显示范围、地图中心点；然后向后端发送 GET 请求查询所有动物的位置信息进行缓存；根据位置信息为每个动物创建 Marker 对象，并将它们添加到地图中形成标记

## 7.2 前后端间交互接口

这些接口主要由后端提供，解析来自前端的请求，分配路由，然后查询数据库、返回数据等。在本系统中，我们采用 REST 风格的 API 制定模式。

### 7.2.1 axios 库

axios 是一个基于 promise 的网络请求库，为前端提供与后端交互的接口。本项目中我们通过 axios 向后端发送网络请求并接收响应，其中主要使用 GET 和 POST 方法

### 7.2.2 HttpHandler 接口

在 Java 中，HttpHandler 类是一个接口，它定义了处理 HTTP 请求的方法。本项目中，我们通过重写其中的 handle 方法，来重新实现这一接口，从而处理不同的 HTTP 请求。

具体而言，我们根据 URL 路径的不同，创建不同的类来实现 HttpHandler 接口；然后在 handle 方法中，根据请求类型，会调用 handleGetRequest()、handlePostRequest() 这两个方法，因此我们也需要重写这两个方法，处理对应请求

下面按类进行接口的介绍

- LoginHandler
  - **接口描述：** 处理前端发来的登录的 POST 请求
  - **实现思路：** 调用后端提供的 QueryUser 接口，在用户表中对账号和密码字段进行精确查找，如果找到，向前端发送登陆成功的信息，反之发送登录失败的信息
- RegisterHandler
  - **接口描述：** 处理前端发来的注册的 POST 请求

  - **实现思路：**调用后端提供的 RegisterUser 接口，在用户表中添加一条记录，向前端发送注册成功的信息；其中，users_id 作为自增列，插入时由数据库生成，插入完成后，会一同返回给前端

- PersonalInfoHandler
  - **接口描述：**处理前端发来的个人信息的 GET 和 POST 请求
  - **handleGetRequest() 实现思路：**调用后端提供的 QueryUser 接口，根据 GET 请求的参数 (users_id)，在用户表中进行精确查找，如果找到，向前端发送对应用户数据，反之报错
  - **handlePostRequest() 实现思路：**调用后端提供的 ModifyUser 接口，根据 POST 请求的参数 (PersonalInfo)，在用户表中进行更新，并向前端汇报

- UsersInfoHandler
  - **接口描述：**处理前端发来的所有用户信息的 GET 请求
  - **实现思路：**调用后端提供的 QueryUser 接口，查询条件为空，查询成功后发往前端

- AnimalInfoHandler
  - **接口描述：**处理前端发来的动物信息的 GET、POST、PUT 请求
  - **handleGetRequest() 实现思路：**调用后端提供的 QueryAnimal 接口，根据 GET 请求的参数，在动物表中进行查找，向前端发送查询结果
  - **handlePostRequest() 实现思路：**调用后端提供的 AddAnimal 接口，根据 POST 请求的参数，在动物表中添加一条记录，向前端发送注册成功的信息；其中，animal_id 作为自增列，插入时由数据库生成，插入完成后，会一同返回给前端
  - **handlePutRequest() 实现思路：**调用后端提供的 ModifyAnimal 接口，根据 PUT 请求的参数，在动物表中进行更新，并向前端汇报
  - **handleDeleteRequest() 实现思路：**调用后端提供的 DeleteAnimal 接口，根据 DELETE 请求的参数，在动物表中进行删除，并向前端汇报

- AdoptionApplyHandler
  - **接口描述：**处理前端发来的领养记录的 GET、POST、PUT 请求
  - **handleGetRequest() 实现思路：**调用后端提供的 QueryAdoptionApply 接口，根据 GET 请求的参数，判断是返回所有记录，还是指定用户的记录

  - **handlePostRequest() 实现思路：** 调用后端提供的 AddAdoptionApply 接口，根据 POST 请求的参数，在领养申请表中添加领养申请记录，将申请结果返回前端
  - **handlePutRequest() 实现思路：** 调用后端提供的 ModifyAdoptionApply 接口，根据 PUT 请求的参数，对指定领养记录的“申请状态”与“审核意见”进行修改；如果审核通过，再调用后端提供的 ModifyAnimal 接口，修改该动物的信息表明其已经被领养

- AdoptionPostHandler
  - **接口描述：** 处理前端发来的待领养信息的 GET、POST 请求
  - **handleGetRequest() 实现思路：** 调用后端提供的 QueryAdoptionPost 接口，返回所有待领养信息
  - **handlePostRequest() 实现思路：** 调用后端提供的 AddAdoptionPost 接口，根据 POST 请求的参数，在待领养表中添加记录
  - **handleDeleteRequest() 实现思路：** 调用后端提供的 DeleteAdoptionPost 接口，根据 DELETE 请求的参数，在待领养表中进行删除，并向前端汇报

- AdoptionGuideHandler
  - **接口描述：** 处理前端发来的领养指南的 GET 请求
  - **handleGetRequest() 实现思路：** 调用后端提供的 QueryAdoptionGuide 接口，查询所有已发布的领养指南，返回给前端
  - **handlePostRequest() 实现思路：** 调用后端提供的 AddAdoptionGuide 接口，根据 POST 请求的参数，在领养指南表中添加记录

#### 7.2.3 Express 依赖

这里利用 Node.js 及其常用的 Web 框架 Express，实现文件上传和下载的功能

- post
  - **接口描述：** 处理前端发来的 POST 请求
  - **实现思路：** 首先指定好本地存储位置，定义 multer 中间件，再使用中间件实现文件的保存

- get
  - **接口描述：** 处理前端发来的 GET 请求
  - **实现思路：** 根据提前规定好的文件路径，构造完整文件路径，直接调用 res.download(filepath) 方法即可，会自动将文件发送到浏览器并弹出下载框

## 7.3 后端内部接口

主要涉及针对不同数据库的各种操作

### 7.3.1 基础信息模块

- QueryUser
  - **接口描述:** 根据提供的查询条件查询符合条件的用户，并按照指定排序方式排序。
  - **实现思路:** 根据查询条件构造 SELECT 语句在用户信息数据库中查询即可
- RegisterUser
  - **接口描述:** 根据提供的用户个人信息（主要是账号密码），完成新用户注册。
  - **实现思路:** 构造 INSERT 语句在用户表中插入一条记录
- ModifyUser
  - **接口描述:** 根据提供的新的用户个人信息，完成数据库中个人信息的修改。
  - **实现思路:** 构造 UPDATE 语句在用户表中更新
- AddAnimal
  - **接口描述:** 根据提供的新的动物信息，完成数据库中动物信息的添加。
  - **实现思路:** 构造 INSERT 语句在动物表中插入一条记录
- QueryAnimal
  - **接口描述:** 根据提供的查询条件查询符合条件的动物，并按照指定排序方式排序。
  - **实现思路:** 根据查询条件构造 SELECT 语句在动物信息数据库中查询即可
- ModifyAnimal
  - **接口描述:** 根据提供的新的动物信息，完成数据库中动物信息的修改。
  - **实现思路:** 构造 UPDATE 语句在动物表中更新
- DeleteAnimal
  - **接口描述:** 根据提供的aninmal_id，完成数据库中动物信息的删除。
  - **实现思路:** 首先构造 SELECT 语句在动物表中查询，如果要删除的动物不存在，那么提示删除失败；构造 DELETE 语句在动物表中进行删除

#### 7.3.2 动物领养模块

- QueryAdoptionApply
  - **接口描述：** 根据提供的查询条件查询符合条件的领养记录，并按照指定排序方式排序。
  - **实现思路：** 根据查询条件构造 SELECT 语句在领养申请表中查询即可
- ModifyAdoptionApply
  - **接口描述：** 根据提供的审核意见 ReviewOpinion，完成数据库中领养申请记录的修改。
  - **实现思路：** 构造 UPDATE 语句在领养申请表中更新
- AddAdoptionApply
  - **接口描述：** 根据提供的新的领养信息，完成数据库中领养申请记录的添加。
  - **实现思路：** 首先构造 SELECT 语句在领养申请表中查询，如果该用户对相同动物的领养记录已经存在，那么添加失败；构造 INSERT 语句在动物表中插入一条记录
- QueryAdoptionPost
  - **接口描述：** 根据提供的查询条件查询符合条件的待领养信息，并按照指定排序方式排序。
  - **实现思路：** 根据查询条件构造 SELECT 语句在待领养信息表中查询即可
- AddAdoptionPost
  - **接口描述：** 根据提供的新的待领养信息，完成数据库中待领养记录的添加。
  - **实现思路：** 首先构造 SELECT 语句在待领养信息表中查询，如果相同动物的待领养信息已经存在，那么添加失败；构造 INSERT 语句在待领养信息表中插入一条记录
- DeleteAdoptionPost
  - **接口描述：** 根据提供的adoption_post_id，完成数据库中待领养信息的删除。
  - **实现思路：** 首先构造 SELECT 语句在待领养表中查询，如果要删除的动物不存在，那么提示删除失败；构造 DELETE 语句在待领养表中进行删除

# 8 可靠性及安全性设计

## 8.1 系统可靠性设计

### 8.1.1 数据持久化与一致性保障

**数据库选择与配置**

选用成熟稳定的关系型数据库管理系统，如本系统使用的 MySQL，并进行合理的配置和优化。

**事务管理**

对于涉及多个数据表更新的关键业务操作，须采用数据库事务来确保操作的原子性。如果系列操作中任何一步失败，所有已执行的操作都将回滚，从而避免数据不一致状态。

**数据校验**

在数据写入数据库之前进行严格的数据校验（类型、格式、范围、业务规则），防止非法或无效数据污染数据库。

### 8.1.2 容错与故障恢复设计

**错误处理机制**

系统各模块应包含健壮的错误处理逻辑。对于可预见的错误（如用户输入错误、资源未找到），应向用户返回清晰的提示信息。

对于不可预见的内部错误或异常，系统应能捕获并记录详细的错误日志，同时避免将敏感的系统内部错误信息直接暴露给用户，而是显示通用的错误提示。

**数据备份与恢复**

必须制定并严格执行数据库的定期备份策略，备份数据应存储在与生产数据库物理隔离的安全位置。

### 8.1.3 系统监控与日志记录

**应用性能监控**

在可行的条件下，可以部署应用性能监控工具，实时监控服务器的关键性能指标

**全面的日志记录**

记录详细的应用运行日志、错误日志、用户操作日志和安全审计日志。日志应包含时间戳、事件级别、来源、详细信息等，便于问题排查、性能分析和安全审计。

## 8.2 系统安全性设计

### 8.2.1 身份认证与授权

**强密码策略与存储**

严禁明文存储用户密码，且制用户设置符合一定复杂度要求的密码。

**会话管理**

使用随机且足够长度的会话 ID，提供安全的登出机制，确保服务器端和客户端会话均被有效终止。

**基于角色的访问控制**

严格区分普通用户和管理员的角色。

每个角色拥有明确定义的权限集合，权限应遵循最小权限原则，即仅授予完成其任务所必需的最小权限。

后端 API 接口必须对每个请求进行严格的身份认证和权限校验，确保用户只能访问其被授权的资源和操作。

### 8.2.2 数据传输安全

涉及敏感数据的传输必须使用安全的传输协议，如 HTTPS，以防止中间人攻击和数据泄露。

### 8.2.3 输入验证

对所有来自用户端或外部系统的输入数据，在服务器端进行严格的合法性校验，包括数据类型、格式、长度、取值范围以及特殊字符的过滤或转义。

严禁直接拼接 SQL 语句，避免将用户输入直接作为命令或参数传递给服务端系统执行。

### 8.2.4 安全审计与日志

记录所有与安全相关的事件，包括用户登录、登出、权限变更、敏感数据访问等。

安全日志应受到严格保护，防止未授权的访问和篡改，并定期进行审查。

## 8.3 系统出错设计

### 8.3.1 错误信息分类与提示

**用户操作错误**

如表单填写不符合规范、输入数据格式错误等。

处理方法：前端进行即时校验并给出明确、友好的错误提示，引导用户修正。

**业务逻辑错误**

如申请领养已被领养的动物、用户权限不足等。

处理方法：后端进行业务逻辑校验，返回明确的错误信息，前端展示给用户。

**资源未找到**

如访问一个不存在的动物详情页或用户页面。

处理方法：返回标准的 `404 Not Found` 页面，并提供导航回主页的选项。

**服务器内部错误**

如数据库连接失败、代码缺陷导致的运行时异常等。

处理方法：系统应捕获此类异常，记录详细的错误日志；向用户显示一个通用的、不暴露内部细节的错误页面或提示；将错误信息发送给管理员进行后续处理。

**网络连接问题**

处理方法：客户端应能检测到网络中断，并提示用户检查网络连接。对于需要网络请求的操作，在请求失败时给出相应提示。

## 8.4 补救措施

进行数据恢复，通过定期备份和验证的恢复计划，确保在数据损坏或丢失时能够将数据恢复到最近的一致状态。

将系统重启和恢复服务，通过详细的日志定位和解决问题。

## 8.5 系统维护设计

清晰的文档：完备的需求文档、设计文档、API 文档、运行使用手册，便于快速理解系统和定位问题。

模块化设计：高内聚、低耦合的模块划分，更容易隔离和修复故障。

规范的编码与注释：清晰、可读的代码和必要的注释，有助于开发和维护人员快速理解代码逻辑。

同时，系统维护人员及时更新技术漏洞，通过各种手段防止各种对系统的攻击，增加代码的可靠性。也需要定期维护数据库，包括检查数据库表与日志文件等，确保数据库内数据的正确性。