

**本科毕业论文（设计）**

**宠物社交平台系统设计与实现**

|  |  |  |  |
| --- | --- | --- | --- |
| **学生姓名** | **：** | 谭鑫锋 | |
| **学号** | **：** | 16B543152 | |
| **学院** | **：** | 互联网金融与信息工程学院 | |
| **专业** | **：** | 计算机科学与技术 | |
| **指导教师** | **：** | 刘宇熹 | **职称：副教授** |
| **提交日期** | **：** | 2020年 3 月 9 日 | |

**本科毕业论文（设计）诚信声明**

本人郑重声明：所呈交的本科毕业论文（设计），是本人在指导老师的指导下，独立进行研究工作所取得的成果，成果不存在知识产权争议，除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究作出重要贡献的个人和集体均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

 学生签名：

时间： 年 月 日

**关于论文（设计）使用授权的说明**

本人完全了解广东金融学院关于收集、保存、使用学位论文的规定，即：

1.按照学校要求提交学位论文的印刷本和电子版本；

2.学校有权保存学位论文的印刷本和电子版本，并提供目录检索与阅览服务，在校园网上提供服务；

3.学校可以采用影印、缩印、数字化或其它复制手段保存论文。

本人同意上述规定。

学生签名：

时间： 年 月 日

摘要

随着人们生活水平品质越来越高和移动终端设备的快速更新换代，越来越多的人喜欢养宠物并把宠物的日常生活动态分享到社交媒体上。从上述现象以及需求出发，本设计主要研究基于宠物信息分享以及社交功能的融合，构建基于宠物这一群体的社交平台。

本设计所构建的宠物社交平台分为客户端以及服务端两个部分，服务端采用Koa作为开发框架，Node.js为Web服务器，Mysql作为数据库，客户端采用微信小程序开发平台。

论文首先介绍了平台开发的背景，进行需求分析，确定系统模块设计方案，然后编码实现整个系统，并且在不同系统的真机上进行了测试，最终完成了宠物社交平台。本设计解决了用户想要了解宠物相关方面知识的需求，并满足了用户需要分享宠物动态以及交流养宠经验的需求，还为用户提供了宠物相关服务。而构建的宠物社交平台实现了登录注册功能，聊天功能，定位功能，发布动态功能，宠物信息管理功能，用户信息管理以及信息分享等功能模块。

本设计将宠物社交与微信小程序平台相结合而设计的社交平台增强了宠物社交平台系统的稳定性，为该社交平台提供了巨大的潜在用户，帮助用户解决了大部分有关宠物的问题，满足了用户对宠物信息分享的需求，提高了用户的生活体验。

**[关键词]**：宠物信息分享；社交平台；微信小程序；Node.js

**Abstact**

As people's living standards are getting higher and higher and mobile terminal equipment is being rapidly replaced, more and more people like to keep pets and share their daily lives on social media. Starting from the above phenomena and needs, this design mainly studies the construction of a social platform based on pets based on the integration of pet information sharing and social functions.

The pet social platform constructed by this design is divided into two parts, the client and the server. The server uses Koa as the development framework, Node.js as the web server, Mysql as the database, and the client uses the WeChat applet development platform.

The dissertation first introduces the background of platform development, conducts requirements analysis, determines the system module design plan, and then codes the entire system, and tests it on real machines of different systems, and finally completes the pet social platform. This design solves the needs of users who want to understand pet-related knowledge, and meets the needs of users who need to share pet trends and exchange pet-raising experiences, and also provides users with pet-related services. The constructed pet social platform implements such functional modules as login registration function, chat function, positioning function, release dynamic function, pet information management function, user information management, and information sharing.

The social platform designed by combining pet social and WeChat mini-program platform in this design enhances the stability of the pet social platform system, provides huge potential users for the social platform and improves the user's life experience.

**[Keywords]**: Pet information sharing; Social platform; WeChat mini program; Node.js

目录

[摘要 I](#_Toc34774297)

[**Abstact** II](#_Toc34774298)

[1 绪论 1](#_Toc34774299)

[1.1 研究背景及意义 1](#_Toc34774300)

[1.2 国内外研究现状 2](#_Toc34774301)

[1.2.1 国内研究现状 2](#_Toc34774302)

[1.2.2 国外研究现状 2](#_Toc34774303)

[1.3 研究目的和内容 3](#_Toc34774304)

[1.4 论文结构安排 3](#_Toc34774305)

[2 宠物社交平台关键技术介绍 3](#_Toc34774306)

[2.1 Node.js+Koa框架 3](#_Toc34774307)

[2.2 Vue.js 4](#_Toc34774308)

[2.3 JSON数据格式 4](#_Toc34774309)

[2.4 MySQL数据库 4](#_Toc34774310)

[2.5 C/S架构 5](#_Toc34774311)

[2.6 微信小程序 5](#_Toc34774312)

[2.7 HTML+CSS+JavaScript 5](#_Toc34774313)

[3 系统需求分析 6](#_Toc34774314)

[3.1 需求概述 6](#_Toc34774315)

[3.1.1 产品特点分析 6](#_Toc34774316)

[3.1.2 用户特点分析 7](#_Toc34774317)

[3.2 功能性需求 7](#_Toc34774318)

[3.2.1 用户登录注册 8](#_Toc34774319)

[3.2.2 宠物信息功能 8](#_Toc34774320)

[3.2.3 宠物百科 9](#_Toc34774321)

[3.2.4 宠物常见问题 9](#_Toc34774322)

[3.2.5 宠物服务商家定位 9](#_Toc34774323)

[3.2.6 社区动态需求 9](#_Toc34774324)

[3.2.7 发布动态 9](#_Toc34774325)

[3.2.8 个人中心 9](#_Toc34774326)

[3.3 非功能性需求 10](#_Toc34774327)

[4 系统设计 10](#_Toc34774328)

[4.1 系统设计基本原则 11](#_Toc34774329)

[4.2 系统物理架构设计 11](#_Toc34774330)

[4.3 系统逻辑架构设计 12](#_Toc34774331)

[4.4 系统功能设计 13](#_Toc34774332)

[4.5 系统数据库设计 14](#_Toc34774333)

[5 系统实现 19](#_Toc34774334)

[5.1 开发环境和开发工具 19](#_Toc34774335)

[5.1.1 开发语言 19](#_Toc34774336)

[5.1.2 开发工具 19](#_Toc34774337)

[5.1.3 开发环境 19](#_Toc34774338)

[5.2 系统功能实现 20](#_Toc34774339)

[5.2.1 用户登录和注册功能的实现 20](#_Toc34774340)

[5.2.2 宠物信息管理功能的实现 22](#_Toc34774341)

[5.2.3 社区动态功能的实现 23](#_Toc34774342)

[5.2.4 在线聊天功能的实现 24](#_Toc34774343)

[5.2.5 宠物相关服务功能的实现 25](#_Toc34774344)

[5.2.6 用户信息管理功能的实现 26](#_Toc34774345)

[6 系统测试 28](#_Toc34774346)

[7 结论与展望 28](#_Toc34774347)

[参考文献 30](#_Toc34774348)

[致谢 32](#_Toc34774349)

[附录 程序目录结构展示及部分重要代码 33](#_Toc34774350)

[1前台 33](#_Toc34774351)

[1.1 前台目录结构 33](#_Toc34774352)

[1.2 前台部分重要代码 33](#_Toc34774353)

[2后台 33](#_Toc34774354)

[2.1 后台目录结构 33](#_Toc34774355)

[2.2 后台部分重要代码 33](#_Toc34774356)

**宠物社交平台系统设计与实现**

1 绪论

## 1.1 研究背景及意义

伴随着经济水平的快速增长，人口老龄化现象越来越明显，丁克家庭越来越多，养育宠物逐渐变成人们娱乐休闲、情感寄托的一种热门选择。宠物热不仅一种时尚、审美，也是对生活的热爱，还可以从宠物的发展影射一个国家的经济发展水平。据研究数据表明，在我国，宠物各个方面需求正以15%的速度增加。到2020年，中国宠物商品/服务消费规模有很大机会突破2000亿元。由此可见，我国宠物行业正处于一个飞速发展期，宠物行业存在着巨大的人力和财力需求。[1]

根据《2019年中国移动互联网行业分析报告-市场现状与发展趋势分析》显示，近年来，我国移动互联网发展速度迅速。我国移动互联网接入流量从2016年上半年的37.5GB至2019年上半年的553.9亿GB，增长约15倍。随着移动互联网和移动设备终端的普及，我国网民使用手机上网的比例也逐渐增多。截至2019年6月，我国网民使用手机上网的比例达99.1%，较去年提升0.5个百分点。2019年上半年，手机网民经常使用的APP中，即时通信类APP占用时间最长，比例为14.5%。随着移动互联网的快速发展，使得宠物行业也能够通过移动终端为广大养宠物的用户构建一个宠物交流平台，丰富养育宠物的用户的生活和解决在这过程中遇到的问题。[2]

宠物社交平台，正是在基于养宠物的用户越来越多的大趋势下和移动智能设备普及率越来越高的情况下应运而生的。通过该平台，已经有宠物的用户可以在线与其他用户进行交流，快速定位用户群体，让用户可以与其他用户分享养宠物的经验。同时，对于暂时没有养宠物的用户，平台提供了许多不同种类宠物的信息，让用户可以在养育宠物之前对想养育的宠物品类有一个初步的了解。宠物社交平台的目的就是为了让更多养育宠物的用户进一步了解宠物，让宠物更好地生活以及愉悦自己的生活。

## 1.2 国内外研究现状

### 1.2.1 国内研究现状

相关数据显示，中国宠物狗的数目已经达到2740万只，城市平均养宠率约达11%，为世界排名前三的"养宠大国”。自2012年-2017年，我国宠物行业市场规模持续扩大，年均复合增长率达到34.63%，增长速率十分快速。据统计，2017年中国宠物行业市场营收已达到1470亿，同比增长20.5%。未来，随着宠物饲养观念的标准化和宠物行业延伸服务的普及，我国宠物行业的市场占有率将进一步扩展。未来3到5年，宠物行业经济将会持续增长，到2021年，养宠渗透率不断提升，预计城镇养宠率将达到19%，宠物市场容量有望翻一番，复合年均增长率为16.6%。到2022年，宠物市场规模将有望突破2500亿元。随着国内宠物经济消费升级的进一步推进，消费者用户对高品质线上宠物消费方式的接受程度也将会越来越高。京东大数据显示，2015-2018年京东平台上，宠物消费始终保持着100%以上的增长，且有不断平稳上升的趋势。

### 1.2.2 国外研究现状

宠物行业在发达国家已经有了上百年的发展历史，从宠物经济产值来看，欧美宠物经济产值占全球宠物总经济产值的60%，其中美国37%，欧洲23%。美国是世界上最发达和规模最大的宠物经济体。美国宠物产品协会(APPA)数据显示，2018年美国宠物消费支出金额达到725.6亿美元，与2017年695.1亿美元相比，增长4.4%，预计2020年美国宠物消费支出金额仍将继续保持稳定增长，规模达到753.8亿美元左右。根据APPA官方网站披露，2019-2020年度的全国宠物主人调查统计数据显示，67%的美国家庭拥有至少一只宠物，拥有宠物的家庭数量已到达8490万户。千禧一代是宠物主人中的最大部分，调查结果显示，超过80%的千禧一代宠物主人拥有狗，而50%左右的人拥有猫。[3]

### 1.2.3 现行研究存在的问题及解决方法

现行研究存在的主要问题是快速发展的宠物行业与移动互联网并没有有效地进行结合，移动互联网的快速发展本应该给宠物行业带来更大的收益，为了解决这一问题，应该着力思考如何去将宠物与移动互联网相结合，本论文旨在开发一个宠物社交平台去解决这一问题。

## 1.3 研究目的和内容

通过宠物社交平台，打造一个以宠物相关信息为核心的大环境，让用户通过该平台随时随地了解与分享自己感兴趣的宠物信息，促进用户与用户之间的交流，进而让用户与自己的宠物更好地生活，增添生活的乐趣。

## 1.4 论文结构安排

本论文分为七个章节，第一章是绪论，主要介绍当前养育宠物的历史发展以及国内外的研究现状，同时对论文的研究内容进行概述。

第二章是介绍开发宠物社交平台系统的关键技术，主要从技术以及平台方面进行说明，以及介绍设计中要解决的问题。

第三章是系统需求分析，通过对用户进行需求分析，确定系统所需要的功能。

第四章是系统设计，通过对系统总体设计，系统逻辑设计以及系统详细设计，确定系统的界面功能以及数据需求。

第五章是编码实现宠物社交平台系统。

第六章是对系统各个功能进行测试，检测稳定性以及是否完成预期需求。

第七章是对论文的总结，通过对宠物社交平台的开发过程以及途中遇到的问题进行了总结，同时对未来宠物社交平台的优化以及扩展作出了展望。

2 系统相关技术及要解决的关键问题

## 2.1 系统相关技术介绍

### 2.1.1 Node.js+Koa框架

Node.js是基于Chrome V8引擎的JavaScript运行时环境，Node.js使用事件驱动的非阻塞式I/O模型，使其轻巧高效。[4]Koa框架是建立在Node.js平台上的Web框架，是Express基于Node.js的下一代Web框架,使用Koa编写Web 应用程序，通过组合不同的Generator[5]，可以避免重复嵌套回调函数，大大提高了错误处理的效率。Koa在内核方法中未绑定任何中间件，它仅提供了一个轻量而优雅的函数库，使编写 Web 应用变得得心应手。[6]

### 2.1.2 Vue.js

Vue.js是一套用于构建用户界面的渐进式框架。和其他前端框架一样，Vue.js允许你将一个网页分割成可复用的组件，每个组件都有自己的HTML、CSS和JavaScript来渲染网页中一个对应的位置。如果要构建一个大型应用，就需要先搭建项目，配置一些开发环境等。Vue.js提供了一个命令行工具，它让快速初始化一个真实的项目工程变得非常简单。及各种支持类库结合使用时，Vue也完全能够为复杂的单页应用提供驱动。[7]

### 2.1.3 JSON数据格式

JSON，即 JS 对象标记，其是一种与语言无关的数据交互格式JSON，即 JS 对象标记，其是一种与语言无关的数据交互格式，它的数据格式不仅可以人为读写，也可通过机器来分析和产生。JSON 原本主要用于 JS 间的数据传输，但后来由于其独立于语言的特性逐渐发展成了一种跨语言的数据交换格式。JSON 目前主要被 C、C++、C#、Java 和JavaScript等编程语言广泛使用。JSON 中最常用的数据结构是数组还有对象。对象由键和值组成，称为键值对，该结构在不同的语言中实现也可能不同，如 Java 中用 Map 来表示、C#则用 Dictionary，表示数组和其他语言一样，也是有序的，该结构在实现上也随着语言的不同而不同，比如 C++可能使用链表、而 Java 则使用 List。[8]

### 2.1.4 MySQL数据库

MySQL数据库是一种C/S模型（即客户端和服务端模型），客户端通过账号和密码来连接服务器，连接成功之后才能对数据库进行操作（CRUD：增加、删除、变更、查询）。MySQL的服务端采用IO复用和可伸缩的连接池，完成了网络高并发的经典模型。[9]

### 2.1.5 C/S架构

服务器-客户机，即Client-Server([C/S](https://baike.baidu.com/item/C%2FS))结构。C/S结构一般主要采取两层结构。分别是服务器和客户机，服务器主要负责数据管理，客户机主要负责完成和用户的交互任务。

客户机通过局域网和服务器相连接，客户机接受用户的请求，并通过互联网向服务器发起请求，然后服务器对数据库进行目标操作。服务器收到客户机的请求，然后将数据库查询到的数据发送给客户机，客户机将数据进行预处理并将结果呈现给使用客户机的用户。服务器还要提供完善的安全机制以及对数据完整性的处理等操作，并且允许多个不同的客户机同时访问该服务器，这就对服务器的并发性提出了很高的要求。

在C/S结构中，应用程序分为两个部分:服务器部分还有客户机部分。服务器部分是多个用户共享的功能与数据信息，服务器执行后台服务，比如说控制共享数据库的操作等;客户机部分是用户专有的部分，主要负责执行前台功能，在错误提示、在线指导等方面都有强大的功能，并且可以在下一级程序间自由切换。

### 2.1.6 微信小程序

微信小程序是近几年来微信推出的一种新的轻应用程序，它的特点是用完即走，不需要用户去下载，开发模式也遵循了MVVM模式，数据驱动视图，单向数据流，组件化在小程序里也得到了很好了的体现。[10-11]

### 2.1.7 HTML+CSS+JavaScript

HTML是一种超文本标记语言，可以在文档上编辑标签从而使文档格式统一而又标准化。[12]CSS又称为层叠样式表，它可以修饰HTML标签，从而使文档样式变得丰富多彩，它不仅仅可以修改静态网页，还可以动态地配合各种脚本语言对网页进行修改。[13]JavaScript目前已成为世界上最流行的脚本语言之一，它不仅可以用于HTML和Web,更可以广泛地运用于服务端，客户端。[14-15]

## 2.2 系统要解决的关键问题

### 2.2.1 静态数据问题

系统中有宠物品种信息数据以及很多文章数据，采用人工录入方式肯定是不现实的，所以针对数据方面，系统采用puppeteer前端自动化工具对数据进行爬取。

### 2.2.2 用户鉴权问题

系统没有采用微信小程序自带的微信用户授权方式对用户进行鉴权，而是采用JWT标准对用户进行鉴权，使实现单点登录变得更为简单便捷。

3 系统需求分析

## 3.1 需求概述

在对项目进行设计步骤前，首先要弄清两个方面的信息，一个是产品特点，另一个是用户特征，在此基础上才能对项目进行准确清晰的需求分析。在充分了解了产品所需要的的功能后，对产品进行分析以及之后的设计方案，才是最符合课题要求的。

### 3.1.1 产品特点分析

正如第一章所提及到的，本设计的主旨是为广大养宠物的用户提供一个方便交流与分享宠物信息的社交平台，这句话的三个关键词是：“方便”，“分享”和“社交”。针对这三个方面，可以针对总体的需求从以下几个方面展开。

(1)平台的选择，随着移动设备和网络通讯技术的迅速迭代发展，2017年发展迅速的微信小程序出现在大众视线中，截止到2019年第一季度，微信的使用用户已经达到了11亿，微信小程序以微信这一聊天社交软件作为载体，潜在的使用用户是十分庞大的。微信小程序是一种轻型App，它不需要下载，用户使用完就可以退出关闭，具有发布信息，发布广告和服务功能。[16]它不局限于移动设备的操作系统类型，无论是Android操作系统，ios操作系统还是windows操作系统的移动设备，只要下载了微信，都可以通过微信打开小程序，随用随打开，用完即走。所以使用微信小程序平台作为宠物社交平台的客户端平台，不仅降低了宠物社交平台的开发成本，提高了宠物社交平台运行的稳定性，还为宠物社交平台提供了巨大的潜在用户。

(2)社交平台通常采用的交流方式是在线聊天，宠物社交平台的特点就是为喜欢宠物的用户们提供一个社交平台，通过宠物这一媒介为不同地区的用户交流彼此的养宠经验，分享自己的爱宠动态。所以宠物社交平台在提供在线聊天功能的基础上，还提供了发布动态功能，类似于微信的朋友圈功能，用户不仅可以发布自己宠物的图片或者视频，还能看到其他用户分享的动态内容，通过分享自己宠物动态的形式，和其他用户进行交流，满足了用户想要分享自己爱宠的需求。[17]

(3)有许多想要养宠物的用户想要通过一个平台去了解宠物信息，宠物社交平台针对这一用户群体，特意设计了宠物百科，宠物知识介绍等模块向想要了解宠物信息的用户提供了一个渠道，宠物百科里面包含了常见的宠物种类以及品种，详细介绍了不同种类宠物的特点以及生活习性。在宠物知识介绍里，介绍了一些在日常生活中宠物会出现的一些问题以及如何去解决这个问题的方法。想要养宠物的用户可以通过这些功能模块初步了解宠物信息，进而找到自己想要养的宠物类型。[18-19]

(4)宠物是宠物社交平台的核心，许多用户平时对宠物的花费主要在食物，护理和医疗方面，针对这一方面，宠物社交平台提供了记账本功能，方便用户记录每一次的花费。微信小程序开放了地理位置功能，宠物社交平台可以通过这一点，为用户就近搜索到附件的宠物医院，宠物店等。

### 3.1.2 用户特点分析

宠物社交平台针对的用户主要分为两种，一种是已经有宠物的用户，另外一种是暂时还没有宠物的用户，不同特点的用户也应该区别分析。

(1)有宠物的用户绝大部分的需求是分享动态，交流经验，发布问题等，作为微信小程序形态，所以这些功能不仅需要有，对于排版布局的要求可能也会不同。

(2)对于没有宠物的用户，除了上述等基本功能外，还需要有介绍引导性的功能模块。

## 3.2 功能性需求

通过上述需求分析，确定了用户主要通过平台分享宠物信息，和其他用户进行交流。确定了本系统的用例图如图3-1所示：

图3-1功能需求用例图

### 3.2.1 用户登录注册

新用户进入平台时要注册账号，需要用真实手机号进行注册，需要用手机号接收验证码进行注册。用户登录的方式有两种，分别是手机号密码登录和手机号验证码登录。

### 3.2.2 宠物信息功能

用户进入平台后，可以添加自己的宠物信息，包括宠物的品种，头像，性别，年龄等等。每个用户可以添加多个宠物信息。

### 3.2.3 宠物百科

对常见的宠物进行信息收集，包括犬科，猫科，兔类，鼠类，龟类，鱼类，爬行类，鸟类等。

### 3.2.4 宠物常见问题

针对宠物日常出现的问题，包括健康问题，饮食习惯，性格培养，技能练习收集相关的科普知识文章，

### 3.2.5 宠物服务商家定位

经过用户授权同意后，获取用户的地理位置，对用户所处的周边地区进行宠物服务商家定位，比如宠物医院，宠物护理店，宠物商品店等。

### 3.2.6 社区动态需求

用户在社区可以看到其他用户发布的动态，动态内容可以是文字，图片和视频等类型，社区不限制用户权限，采取每个用户都能看到其他用户动态内容的形式，用户可以评论和点赞其他用户的评论。用户可以看到发布动态的用户的个人信息，可以关注其他用户，可以和其他用户进行一对一聊天。

### 3.2.7 发布动态

每个用户可以在社区模块界面发布动态，动态内容可以是文字，图片和视频。

### 3.2.8 个人中心

用户个人中心包括个人信息，我的宠物，我的动态，我的关注，我的收藏，退出登录等模块，

用户在个人信息模块中修改自己的个人信息，包括昵称，性别和地区。我的宠物模块可以查看，修改或新增用户的宠物信息。我的动态模块中保存了用户之前发布的动态，包括图片动态和视频动态。我的关注模块里可以查看和取消用户已经关注的其他用户，可以和已关注用户进行私聊，可以查看已关注用户的个人信息。我的收藏模块记录了当前用户点赞的其他用户已发布动态，用户可以取消点赞或者查看该点赞动态详情。退出登录模块可以让当前用户退出宠物社交平台，回到登录页面。

## 3.3 非功能性需求

(1)易用性

系统界面设计一定要遵循易用性原则，系统界面要简单易用，不要设计太复杂的界面，

只有简单易用的操作界面才能让用户更方便使用。

(2)可扩展性

系统设计需要具有良好的可扩展性，本系统功能肯定没有涵盖宠物社交平台所有需要具备的功能,在以后系统运行中，会根据不同情况的需求对系统进行修改和完善。由于条件限制，许多功能还暂时没有实现。

(3)性能需求

小程序单个分包不能超过2M,所以要限制源代码包的大小，将一些比较大的静态图片和文件进行压缩，避免用户首次加载小程序的时候出现等待时间过长的现象，对于加载数据量比较大的模块要采取懒加载或者分页加载等形式进行。

4 系统设计

系统设计主要从物理层和逻辑层两方面来考虑。其中物理层决定系统架构，逻辑层决定系统功能实现的思路。系统设计是整个工作的前提和基础，决定了整个系统的框架结构和系统功能的实现步骤与逻辑，设计的优劣会影响系统整体质量和功能实现情况。本系统设计主要从系统总体设计、系统逻辑设计和系统详细设计三个方面展开。[20]

## 4.1系统总体设计

### 4.1.1 系统设计基本原则

(1)简易性原则

简单易操作的界面可以保证用户们操作本系统的正确率和流畅性，大大减少用户因操作上而造成的失误。与此同时，合适的颜色搭配可以提高用户的使用体验。

(2)模块化设计原则

对系统进行细化和分解，将其分解成许多个内在联系强、相对独立的小模块，每个小模块各司其职，实现系统不同的功能，随后再对每个小模块进行单独设计与开发，让设计变得简单。

### 4.1.2 系统架构设计

本系统的物理架构主要由用户移动终端、移动互联网络(2G/3G/4G/5G网络、WIFI等)、微信服务器、互联网、信息交互系统五大部分组成，其主要结构如图4-1所示：

图4-1 物理架构示意图

其中用户智能移动终端不仅仅是手机，像平板电脑和PC端等，只要可以连接移动互联网并且可以登录微信都可以应用于本系统。其中，移动互联网络(2G/3G/4G/5G网络、WIFI等)是用户与微信之间沟通的桥梁，而微信服务器是腾讯公司官方提供的，不需要我们去准备，而微信服务器作为一个“中转站”为用户提供了许多用于登录微信的接口。微信服务器会通过互联网与本系统自身的服务器相连接，因此本系统的数据是直接存储在系统自身的服务器上的。

本系统信息交互流程主要如图4-2所示：

图4-2系统信息交互流程图

从图中可以得知，用户先通过微信客户端发送信息给微信小程序，微信小程序接收到信息后会发送到微信主服务器，主服务器接收到信息后，会自动将信息封装成JSON格式发送给小程序所对应的第三方服务器进行数据处理，完成处理后第三方服务器将信息封装成JSON格式数据返回给微信服务器，再由微信服务器返回给小程序，最终发送给用户的移动终端。

## 4.2 系统逻辑设计

系统的正常运行需要后台的支持和对数据进行预处理，系统总体逻辑框架图如4-3所示：

图4-3系统总体逻辑框架图

系统使用的是C/S架构，主要分为以下三方面。

表示层(UI)。表示层是用户进入本系统的入口，为用户和系统之间提供了接口。在本系统中，表示层主要是微信小程序客户端。

业务逻辑层(BLL)。本系统的业务逻辑层主要由微信服务器和系统服务器两部分组成。

逻辑层不参与微信小程序的前端展示，逻辑层会对前端返回的数据进行处理。在本系统中，业务逻辑层主要是根据用户发送的请求选择对应的模块进行调用。

数据访问层(DAL)。数据访问层是系统中最关键和最重要的一层。数据是系统稳定运行的基础。数据访问层接收到的数据主要分为两类，一类为持久化数据，这类数据需要在数据库中长期保存，另一类是非持久数据，这类数据通常是一些中间数据，作为计算机处理数据的一个中间值。数据访问层主要执行数据的存储和提取等问题。

## 4.3 系统详细设计

### 4.3.1 系统功能模块设计

在需求分析阶段已经了解了系统的任务是什么，根据需求分析的内容，将系统拆解成各个功能模块，明确各个模块之间的联系及接口定义。同时在系统数据需求的基础上，设计出相应的数据库。系统功能模块图如图4-4所示：

图4-4系统功能模块图

(1)登录注册模块

用户初次使用本系统时，需要进行注册，注册需要使用手机号码和手机验证码，验证通过后用户需要设置密码，确认密码后用户就可以登录进入本系统了。已经注册过的用户，可以通过手机号码和密码登录或者手机号码和手机验证码登录，验证通过后进入本系统。

(2)用户信息管理模块

用户基本信息包括用户的昵称，头像，性别，地区等，用户初次登入系统时会随机设置基本信息，用户可以在个人中心修改基本信息。除了基本信息外，还有用户宠物，用户动态，用户收藏，用户关注等信息。

(3)宠物信息管理模块

宠物基本信息包括品种，年龄，性别，头像，是否绝育等基本信息。用户可以添加并修改宠物基本信息。

(4)社区动态模块

用户可以在社区发表动态，动态内容可以是文字，图片和视频，发布后的动态对所有人可见，其他用户可以对动态进行评论和点赞，用户可以关注其他用户。

(5)聊天模块

用户与用户之间可以进行实时聊天，聊天内容可以是文字，也可以是图片。

(6)宠物相关服务模块

在养宠界面，设置了宠物百科，宠物知识文章介绍，宠物花销记事本等模块，开放用户定位服务，可以实时查询地图。

### 4.3.2 系统数据库设计

数据库是按照数据结构来组织、存储和管理数据的仓库。经过对宠物社交平台系统的需求分析和功能模块设计之后，可以对宠物社交平台系统概念模型抽象出用户、宠物、动态，评论等等实体。其中用户实体的E-R图如图4-5所示，一个用户实体主要包括账号，密码，性别，地区，昵称，头像等。

图4-5用户实体关系图

宠物实体主要包括昵称昵称，性别，出生日期，品种，绝育状态，头像等基础属性，宠物实体的E-R图如图4-6所示：

图4-6宠物实体关系图

动态实体主要包括发布用户账号，发布用户头像，发布用户昵称，动态内容，发布时间等基础属性，动态实体的E-R图如图4-7所示：

图4-7动态实体关系图

评论实体主要包括评论用户账户，评论用户头像，评论用户昵称，评论内容，评论时间，回复用户账户，回复用户头像，回复用户昵称等基础属性,评论实体的E-R图如图4-8所示：

图4-8评论实体关系图

根据概念结构设计中用户实体，宠物实体，动态实体，评论实体等实体以及各个实体之间的操作设计出了本系统的数据库表，以下是对关键数据库表中的宠物表、用户表、动态表、评论表，聊天内容表的详细介绍。

表4-1 用户表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 约束 |
| phoneNumber | varchar | 用户账户 | 主键,不能为空 |
| password | varchar | 用户密码 |  |
| gender | varchar | 用户性别 |  |
| name | varchar | 用户昵称 |  |
| avatar | varchar | 用户头像 |  |
| region | varchar | 用户地区 |  |
| bgImage | varchar | 背景图片 |  |

表4-2 宠物表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 约束 |
| pid | int | 宠物ID | 主键,不能为空 |
| uid | int | 用户ID |  |
| name | varchar | 宠物昵称 |  |
| gender | varchar | 宠物性别 |  |
| birthDay | varchar | 宠物生日 |  |
| avatar | varchar | 宠物头像 |  |
| breed | varchar | 宠物品种 |  |
| state | int | 是否 |  |

表4-3 动态表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 约束 |
| post\_id | int | 动态ID | 主键,不能为空 |
| uid | varchar | 用户ID |  |
| name | varchar | 用户昵称 |  |
| avatar | varchar | 用户头像 |  |
| content | mediumtext | 动态内容 |  |
| islike | varchar | 点赞数 |  |
| timestamp | varchar | 发布时间 |  |

表4-4 评论表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 约束 |
| comment\_id | int | 评论ID | 主键,不能为空 |
| post\_id | int | 动态ID |  |
| uid | varchar | 用户ID |  |
| avatar | varchar | 评论用户头像 |  |
| name | varchar | 评论用户昵称 |  |
| content | varchar | 评论内容 |  |
| timestamp | varchar | 评论时间 |  |
| to\_uid | varchar | 回复用户ID |  |
| to\_avatar | varchar | 回复用户头像 |  |
| to\_name | varchar | 回复用户昵称 |  |

表4-5 聊天内容表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 说明 | 约束 |
| m\_id | int | 消息ID | 主键,不能为空 |
| send\_id | int | 发送用户ID |  |
| rec\_id | int | 接收用户ID |  |
| content | varchar | 聊天内容 |  |
| type | int | 消息类型 |  |
| timestamp | varchar | 发送时间 |  |
| image | varchar | 图片路径 |  |

5 系统实现

## 5.1 开发环境和开发工具

### 5.1.1 开发语言

(1)应用Node.js作为开发后台Web服务器的语言。

(2)前端是小程序平台，使用Html+Css+JavaScript配合使用实现。

(3)多种数据库语言用于增删改查数据。

### 5.1.2 开发工具

(1)集成开发环境微信开发者工具(测试版)。

(2) Navicat for MySQL。

(3)Visual Studio Code。

(4)HBuilder X。

### 5.1.3 开发环境

(1)电脑：内存12G，Windows10操作系统的电脑一台。

(2)Web服务器：Koa。

(3)IOS13.3.1系统手机2部。

## 5.2 系统功能实现

### 5.2.1 用户登录和注册功能的实现

经过慎重考虑后，登录功能没有使用微信小程序官方的授权登录，而是使用的手机号码注册和登录的方式。用户注册后可以设置密码，设置密码后直接登录进入宠物社交平台，已经注册过的用户可以选择手机号码和手机验证码登录或者手机号码和密码登录。登录成功后，后台会返回token用于前端进行授权接口。请求注册登录界面如图5-1，图5-2所示：



图5-1 注册界面

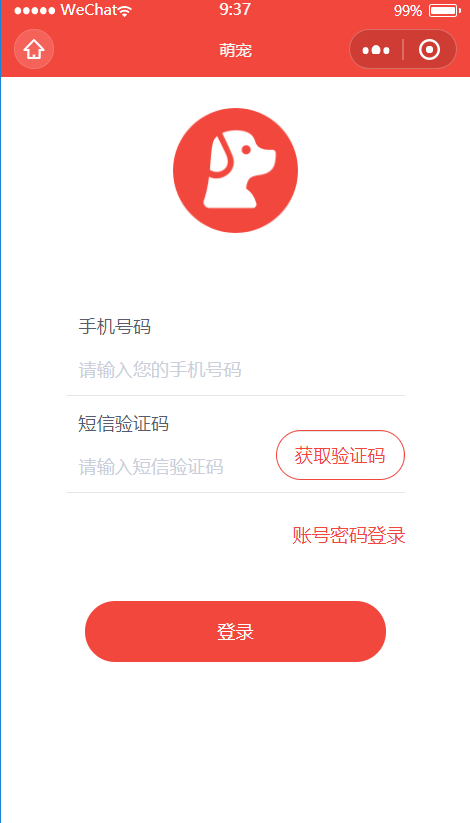


图5-2 两种登录方式界面

用户登录成功后会进入到系统初始页面“养宠”选项卡，“养宠”界面中主要由宠物信息，记账本，宠物百科，同城服务，精选文章等五个功能组成。如图5-3所示：



图5-3登录后初始页面

### 5.2.2 宠物信息管理功能的实现

用户登录宠物社交平台后，可以添加宠物信息，可以添加多个宠物，添加宠物信息界面如图5-4所示：



图5-4添加宠物信息界面

### 5.2.3 社区动态功能的实现

用户点击“社区”选项卡可以进入到“社区”界面，用户在社区界面里可以看到社区动态列表，动态列表里可以看到其他用户发布的动态，当前用户也可以发布动态，在界面顶部可以看到两个图标，点击“视频”图标可以进入发布视频动态界面，点击“相机”图标可以进入发布图片动态界面。社区动态功能具体操作界面如图5-5所示：

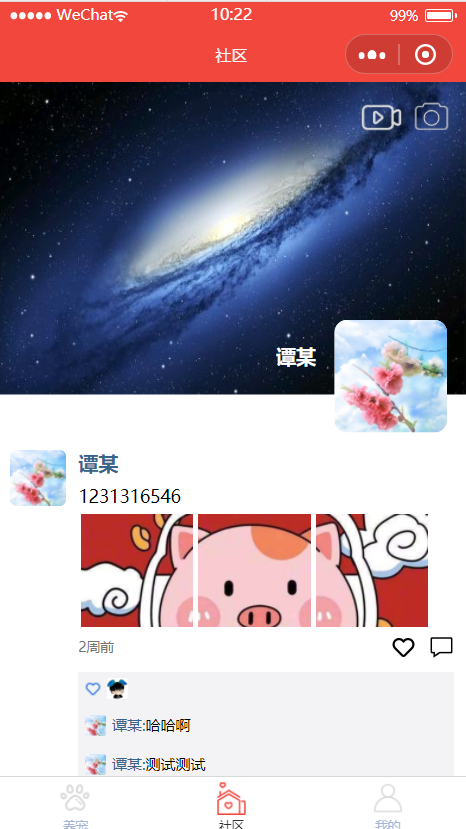


图5-5发布社区动态

### 5.2.4 在线聊天功能的实现

用户在“社区”界面点击其他用户的头像可以查看其它用户的个人信息，可以关注该用户，也可以和该用户在线进行一对一聊天，聊天内容可以是图片或者文字。在线聊天界面如图5-6所示：



图5-6在线聊天界面

### 5.2.5 宠物相关服务功能的实现

用户在“萌宠界面”点击宠物百科，地图定位，生活记事本等模块的图片时，可以进入对应功能模块的界面。下面只详细介绍宠物百科功能，用户点击宠物百科模块后，会进入宠物种类列表，用户可以选择任意一个种类列表进入该种类品种列表，进入种类品种列表后，可以点击任意一个品种查看该品种详细信息。宠物百科具体界面如图5-7所示：

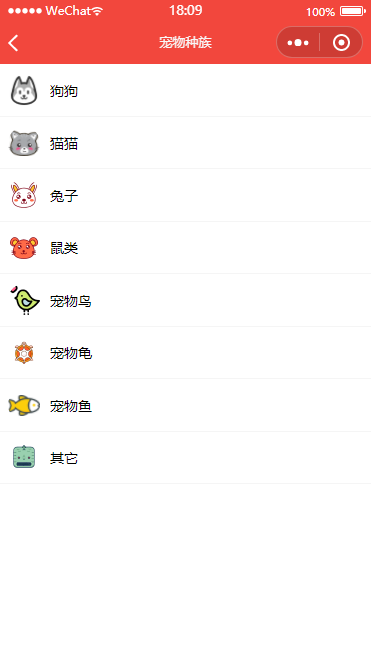


图5-7宠物百科界面

### 5.2.6 用户信息管理功能的实现

用户在“我的”选项卡界面增删改查自己的信息，包括个人资料，我的宠物，我的关注，我的动态，我的收藏，退出登录等功能。用户信息管理界面如图5-8所示：

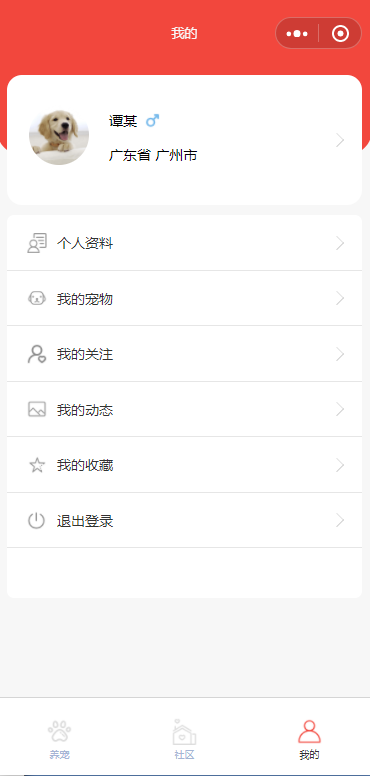


图5-8用户信息管理界面

用户点击任意一个模块都可以进入对应的界面，下面只详细介绍个人资料界面，个人资料包括姓名，性别，地区等基础属性。个人资料界面如图5-9所示：



图5-9个人资料界面

6 系统测试

## 6.1 系统测试环境

系统测试环境如下：

(1)服务器：内存12G，硬盘大小为500GB。

(2)操作系统：Windows10。

(3)开发工具：Visual Studio Code、HBuilder X、Navicat for MySQL、微信开发者工具(测试版)。

(4)数据库：MySQL Server5.7

(5)Web服务器：Koa

(6)IOS系统版本：IOS13.3.1

(7)测试网络：WIFI、4G。

## 6.2 系统功能测试

系统功能测试又称黑盒测试，主要是测试系统的各个功能模块能否正常的使用。系统功能测试步骤主要如下：第一步，根据系统实现的功能模块来确定设计对应的测试用例；第二步，对系统进行测试，由于条件有限，本系统采取人工测试，在测试过程中找出系统存在的问题；第三部，对测试结果进行分析而后生成测试报告。

本系统功能测试主要测试系统的各功能模块以及各功能子模块之间的接口是否正常运行，然后测试系统每个功能界面之间的跳转和数据连接是否正常，以及服务器与移动终端之间的通信是否正常。下面主要测试各功能模块的核心功能。

(1)登录注册功能测试用例如表6-1所示，主要是为了验证登录注册功能是否能正常使用。

表6-1 登录注册功能测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 测试内容 | 预期结果 | 实际测试结果 |
| 1 | 在注册页面输入手机号码和正确的手机验证码注册，进入设置密码页面 | 进入到设置密码页面 | 与预期结果一致 |
| 2 | 在登录页面输入手机号码和正确的手机验证码登录，进入养宠界面 | 提示登录成功并能够根据用户登录的信息显示用户对应的数据 | 与预期结果一致 |

(2)宠物信息管理功能测试用例如表6-2所示，主要是为了验证宠物信息管理功能是否正常使用，数据是否有异常。

表6-2 宠物信息管理功能测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 测试内容 | 预期结果 | 实际测试结果 |
| 1 | 在新增宠物信息页面输入宠物信息并保存 | 提示保存成功并返回养宠界面，宠物信息列表刷新，新增了一条宠物信息 | 与预期结果一致 |
| 2 | 在修改宠物信息页面修改宠物信息并保存 | 提示修改成功并返回养宠界面，宠物信息列表刷新，该宠物信息已修改 | 与预期结果一致 |
| 3 | 删除某条宠物信息 | 提示删除成功并刷新宠物信息列表，宠物信息列表数量减少一条 | 与预期结果一致 |

(3)社区动态功能测试用例如表6-3所示，主要是为了验证发布动态和评论点赞功能是否正常使用，数据是否有异常。

表6-3 社区动态功能测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 测试内容 | 预期结果 | 实际测试结果 |
| 1 | 在发布动态页面输入文字，上传图片或者视频并发送 | 提示发布成功并返回社区动态列表，最新发送的社区动态置顶显示 | 与预期结果一致 |
| 2 | 在社区动态列表中任意评论点赞一条动态内容 | 在该动态内容下会显示评论内容或点赞标志 | 与预期结果一致 |

(4)个人中心功能测试用例如表6-4所示，主要是为了验证我的动态，我的关注，我的收藏等功能是否正常使用。

表6-4 个人中心功能测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 测试内容 | 预期结果 | 实际测试结果 |
| 1 | 在个人中心界面中进入我的动态页面，查看我的动态信息 | 根据用户的账户显示对应的动态信息列表 | 与预期结果一致 |
| 2 | 在个人中心界面中进入我的关注页面，查看我的关注信息 | 根据用户的账户显示对应的关注信息列表 | 与预期结果一致 |
| 3 | 在个人中心界面中进入我的收藏页面，查看我的收藏信息 | 根据用户的账户显示对应的收藏信息列表 | 与预期结果一致 |

(5)在线聊天功能测试用例如表6-5所示，主要是为了验证聊天信息是否能够实时发送。

表6-5 在线聊天功能测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 测试内容 | 预期结果 | 实际测试结果 |
| 1 | 在聊天界面和某用户在线聊天，发送图片或信息 | 用户和对方都可以实时接收到信息 | 与预期结果一致 |
| 2 | 在聊天界面上拉加载历史聊天记录 | 在聊天界面中上拉加载能够显示历史聊天记录 | 与预期结果一致 |

(6)宠物相关服务功能测试用例如表6-6所示，主要是为了验证数据是否正确。

表6-6 宠物相关服务功能测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 测试内容 | 预期结果 | 实际测试结果 |
| 1 | 进入同城服务界面查看地图 | 显示与宠物服务相关的门店 | 与预期结果一致 |
| 2 | 进入宠物百科界面查看品种资料 | 显示对应的品种详情信息 | 与预期结果一致 |
| 3 | 进入记账本查看宠物花费支出 | 显示宠物总计花费以及不同种类花费 | 与预期结果一致 |

## 6.3 系统性能测试

本系统是基于微信小程序平台开发的，所以用微信开发者工具里自带的Audits工具对系统进行性能测试。测试系统启动时间低于3秒，系统内存峰值正常，没有内存泄漏的情况出现；系统网络连接正常，在无网络连接时，能够提示错误，用户体验良好；系统的登陆时间和页面跳转时间都在用户的承受范围之内。

## 6.4 系统兼容性测试

兼容性测试主要测试不同系统移动终端的UI适配性、系统稳定性。兼容性测试结果如表6-7所示。

表6-7 系统兼容性测试结果表

|  |  |  |
| --- | --- | --- |
| 移动设备型号 | UI与机型适配性 | 稳定性 |
| iPhone 7 Plus | 测试通过 | 测试通过 |
| iPhone 7 | 测试通过 | 测试通过 |
| iPad Air3 | 测试通过 | 测试通过 |
| 魅蓝 Note5 | 测试通过 | 测试通过 |

从兼容性测试结果可见，不同系统移动终端都可以稳定运行使用本系统。

7 结论与展望

本系统从需求分析，到设计和编码测试，共经历了3个月的时间，期间进行了多次需求方面的修改。

经过此次项目编程和学习，实现了登录注册模块，用户信息管理模块，宠物信息管理模块，宠物相关服务模块，社区动态模块，在线聊天模块等功能模块的全部预期功能。在功能方面完成度较高。整个系统根据MVC模式进行开发，经过模块化的编程，后台接口和前台页面相对应，对于后期维护来说是十分方便的。数据库设计方面也是达到了预期的要求。

在实现系统的过程中，学习到了诸如MySql，Koa，WebSocket等前后台一系列新的知识和技术。为了优化用户体验，在前台UI界面优化和后台数据处理方面进行了大量的编码工作。比如说上拉加载，加载提示，通讯录功能，显示大图，预览图片等等。实际体验效果也确实得到了较大的提升。

在整个系统中，难点最大的是实现在线聊天功能，如何能够和用户及时的进行信息交互。在本系统中最终使用了WebSocket实现该功能，一改以往IM工具的轮询请求，而是建立全双工通道，使用户双方能够快速进行信息交互。

但是，由于时间关系和技术方面的原因，部分功能没有实现，比如说消息的推送功能还有商城功能。除此以外在编码方面还是存在一些代码冗余的问题，在数据库优化方面还有很大提升的空间，在往后的项目开发过程中应该注意。

参考文献

[1] 朱淑婷. “互联网+”背景下的宠物APP[J]. 中国工作犬业, 2019(02): 62-64.

[2] 陈伟才. 中国宠物行业现状和发展趋势[J]. 中国洗涤用品工2019(08): 56-59.

[3] 安信证券. 美国宠物行业深度: 医食为主，创新迭出[M]. 深圳: 安信证券研究中心, 2018.

[4] 朴灵. 深入浅出Node.js[M]. 北京: 人民邮电出版社, 2013.

[5] 阮一峰. ES6标准入门(第3版)[M]. 北京: 电子工业出版社, 2017.

[6] 忽如寄. Node.js开发实战[M]. 北京: 清华大学出版社, 2018.

[7] 刘博文. 深入浅出Vue.js[M]. 北京: 人民邮电出版社, 2019.

[8] 高静, 段会川. 基于移动设备的JSON数据传输效率研究[J]. 信息技术与信息化, 2011(01): 13-16.

[9] 刘晓光. 基于MySQL的分布式SQL数据库的设计与实现[D]. 中国科学院大学(工程管理与信息技术学院), 2016.

[10] 张帆. 微信小程序项目开发实战——用WePY、mpvue、Taro打造高效的小程序[M]. 北京: 人民邮电出版社, 2019.

[11] 闫小坤. 微信小程序开发详解[M]. 北京: 清华大学出版社, 2017.

[12] Adam Freeman. HTML5权威指南[M]. 谢廷晟等译. 北京: 人民邮电出版社, 2014.

[13] Eric A.Meyer, Estelle Weyl. CSS:The Definitive Guide(4th edition)[M]. New York: O’Reilly Media,Inc, 2019.

[14] Nicholas C.Zaka. JavaScript高级程序设计(第3版)[M]. 曹力等译. 北京: 人民邮电出版社, 2012.

[15] Flanagan D. JavaScript:The Definitive Guide(6th edition)[M]. New York: O’Reilly Media,Inc, 2012.

[16] 田志友, 周元敏, 田雨. 微信小程序的媒体价值[J]. 新媒体研究, 2018, 4(01): 47-49.

[17] 周银. 移动短视频社交应用的发展研究[D]. 南昌大学, 2016.

[18] 徐恒. 社会化网络数据获取技术研究与实现[D]. 吉林大学, 2016.

[19] 赵振东. 定向互联网站点数据爬虫及应用服务接口的设计与实现[D]. 南京大学, 2014.

[20] 刘伟主编. 设计模式[M]. 北京: 清华大学出版社, 2018.

致谢

日月如梭，在本文完成之际也代表着我在广东金融学院互联网金融与信息工程学院的学习生涯即将结束，但在这两年的时光里我收获的知识以及心得体会将会使我受益终身。期间，互联网金融与信息工程学院的老师对我的谆谆教诲不仅丰富我的知识底蕴，更培养了我热衷探索新知识的性格。通过这两年的学习，我相信我能更好地适应社会。

首先我要感谢我的导师刘宇熹副教授，本论文是在他的精心指导和关注下完成的，其次还要感谢我的室友和其他教授我课程的老师为我解疑答惑，解决了本项目中的许多难点。

最后，衷心感谢所有广东金融学院的同学和老师，让我的大学生涯圆满落下帷幕。

附录 程序目录结构展示及部分重要代码

1 前台

## 1.1 前台目录结构

前台目录结构如图1-1所示：

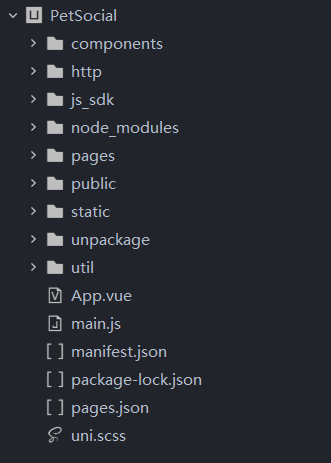


图1-1 前台目录结构

## 1.2 前台部分重要代码

### 1.2.1 接口拦截器方法

为防止非本系统用户非法请求后台接口，后台需要对来自客户端的请求进行授权认证，如果客户端请求没有携带后台返回的token值，则在请求方法中进行拦截。接口拦截器方法如下。

import axios from '@/js\_sdk/gangdiedao-uni-axios'

axios.interceptors.request.use(config => {

let token = uni.getStorageSync('token');

if (token) {

config.headers.Authorization = token;

}

return config;

}, err => {

return Promise.reject(err);

})

export default axios

### 1.2.2 接口请求方法封装

本系统前台需要对后台接口进行频繁的请求，所以将请求方法封装，避免重复编码，接口请求方法代码如下。

import axios from './interceptor.js'

let ApiUrl = 'http://192.168.30.1:3000/API';

export default {

get(url, params) {

return new Promise((resolve, reject) => {

axios.get(ApiUrl + url, {

params: params,

}).then(res => {

resolve(res.data);

}).catch(err => {

reject(err);

})

})

},

post(url, params) {

const config = {

headers: {

'Content-Type': 'application/json'

}

}

return new Promise((resolve, reject) => {

axios.post(ApiUrl + url, params, config).then(res => {

resolve(res.data);

}).catch(err => {

reject(err);

})

})

},

put(url, data) {

let config = {

headers: {

// 'Content-Type': 'application/x-www-form-urlencoded'

'Content-Type': 'application/json'

}

}

return new Promise((resolve, reject) => {

axios.put(ApiUrl + url, data, config).then(res => {

resolve(res.data)

}).catch(err => {

reject(err)

})

})

},

delete(url, data = {}) {

let config = {

headers: {

'Content-Type': 'application/x-www-form-urlencoded'

}

}

return new Promise((resolve, reject) => {

axios.delete(ApiUrl + url, {

params: data

}, config).then(res => {

resolve(res.data)

}).catch(err => {

reject(err)

})

})

},

// 上传文件

uploadFile(file) {

let url = 'http://192.168.30.1:3000/image';

return new Promise((resolve, reject) => {

uni.uploadFile({

url: url,

filePath: file,

name: 'file',

header: {

"Content-Type": "multipart/form-data",

},

success: res => {

console.log(res);

resolve(JSON.parse(res.data));

},

fail: err => {

reject(err);

}

})

})

}

}

2 后台

## 2.1 后台目录结构

后台目录结构如图2-1所示：

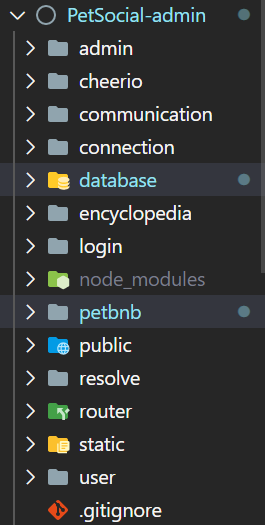


图2-1 后台目录结构

## 2.2 后台部分重要代码

### 2.2.1 数据库CURD方法封装

为了减少每次对数据库进行操作的代码量，将数据库CURD方法封装成了公用函数。代码如下。

const mysql = require('mysql')

class Connection {

constructor() {}

// 初始化 mysql 连接

connection() {

return mysql.createConnection({

host: 'localhost',

user: 'root',

password: 'root',

database: 'pet\_miniprogram',

})

}

// 新增

insert(addSql,addSqlParams) {

let con = this.connection();

return new Promise((resolve,reject) => {

con.connect();

con.query(addSql,addSqlParams,function (err, result) {

if (err) {

reject(err);

}

resolve(result);

})

con.end();

})

}

// 删除

delete(deleteSql) {

let con = this.connection();

return new Promise((resolve,reject) => {

con.connect();

con.query(deleteSql,function(err,result){

if(err){

reject(err);

}

resolve(result);

})

con.end();

})

}

// 修改

update(updateSql,updateSqlParams) {

let con = this.connection();

return new Promise((resolve,reject) => {

con.connect();

con.query(updateSql,updateSqlParams,function (err, result) {

if (err) {

reject(err);

}

resolve(result);

})

con.end();

})

}

// 查询

select(sql) {

let con = this.connection();

return new Promise((resolve,reject) => {

con.connect();

con.query(sql,function (err, result) {

if (err) {

reject(err);

}

resolve(result);

})

con.end();

})

}

}

module.exports = Connection;

### 2.2.2 Node.js服务器

Node.js内置服务器，启动服务器代码如下。

const Koa = require('koa')

const Router = require('koa-router')

const router = **new** *Router*();

const bodyParser = require('koa-bodyparser')

const app = **new** *Koa*();

const koajwt = require('koa-jwt')

const static = require('koa-static')

const path = require('path')

const router\_login = require('../router/login/login');

const router\_petbnb = require('../router/petbnb/petbnb');

const router\_user = require('../router/user/user');

const router\_upload = require('../public/uploadFile');

const router\_community = require('../router/communication/communication')

// 错误处理

app.use((*ctx*, *next*) => {

  return next().catch((*err*) => {

    if (err.status === 401) {

      ctx.status = 401;

      ctx.body = 'Protected resource, use Authorization header to get access\n';

    } else {

      throw err;

    }

  })

})

app.use(static(path.join(path.dirname(\_\_dirname), './static')))

app.use(koajwt({

  secret: 'my\_token'

}).unless({

  path: [/\/API\/Login\/login/, /\/API\/Login\/register/, /\/API\/Login\/captcha/, /\/API\/Login\/setPassword/, /\/image/]

}));

app.use(bodyParser())

app.use(router\_login.routes());   //导入登录注册模块路由

app.use(router\_petbnb.routes());   //导入养宠模块路由

app.use(router\_user.routes());   //导入个人中心模块路由

app.use(router\_upload.routes()); //导入上传图片模块路由

app.use(router\_community.routes()); //导入社区模块路由

app.use(router.routes()).listen(3000, () => {

*console*.log('3000端口已开启');

})

### 2.2.3 路由方法

每个路由对应一个接口，下面只列出登录路由的方法。

const Router = require('koa-router')

const router = new Router();

const bodyParser = require('koa-bodyparser')

const Login = require('../../login/index')

const login = new Login(); // 登录类

const jwt = require('jsonwebtoken')

// 获取验证码

const captcha = async ctx => {

await login.getCaptcha(ctx.request.query.account, ctx.request.query.type).then(res => {

if (res.code === 0) {

ctx.body = {

code: 1,

data: {

},

message: ''

}

} else {

ctx.body = {

code: -1,

data: {

},

message: res.data

}

}

})

}

// 注册获取token

const register = async ctx => {

let {

account,

msgCaptcha

} = ctx.request.body

await login.register(account, msgCaptcha).then(res => {

const token = jwt.sign({

account: res

}, 'my\_token', {

expiresIn: '4h'

});

ctx.body = {

code: 1,

data: {

token: token

},

message: '注册成功!'

}

})

.catch(err => {

console.log(err);

ctx.body = err;

})

}

// 设置密码

const setPassword = async ctx => {

let {

password,

confirmPassword

} = ctx.request.body;

await login.setPassword(password, confirmPassword).then(res => {

const token = jwt.sign({

account: res

}, 'my\_token', {

expiresIn: '4h'

});

ctx.body = {

code: 1,

data: {

token: token

},

message: '登录成功!'

}

})

.catch(err => {

console.log(err);

ctx.body = err;

})

}

// 登录

const getLogin = async ctx => {

let {

account,

password,

msgCaptcha

} = ctx.request.body;

await login.login(account, password, msgCaptcha).then(res => {

const token = jwt.sign({

account: res

}, 'my\_token', {

expiresIn: '4h'

});

ctx.body = {

code: 1,

data: {

token: token

},

message: '登录成功!'

}

})

.catch(err => {

console.log(err);

ctx.body = err;

})

}

router.get('/API/Login/captcha', captcha)

router.post('/API/Login/register', register)

router.post('/API/Login/login', getLogin)

router.post('/API/Login/setPassword', setPassword)

module.exports = router;