太原理工大学
TAIYUAN UNIVERSITY OF TECHNOLOGY

2025 届本科生毕业设计（论文）

基于 Spring Boot 的同城上门喂宠预约系统的设计与实现

学　　号：2021005509

姓　　名：霍劲羽

学　　院：软件学院

专　　业：软件工程

班　　级：软件 2114 班

指导教师：魏振楠 人工智能开发工程师

闫　琛　　讲师

完成日期：2025 年 6 月

声明及论文使用的授权

本人郑重声明：所呈交的毕业设计（论文）是本人在指导教师的指导下取得的研究成果，毕业设计（论文）写作严格遵循学术规范。除了文中特别加以标注和致谢的地方外，毕业设计（论文）中不包含其他人已经发表或撰写的研究成果。因本毕业设计（论文）引起的法律结果完全由本人承担，太原理工大学享有本毕业设计（论文）的研究成果。

论文作者签名 霍勐羽　　2025 年 6 月 3 日

本毕业设计（论文）作者和指导教师同意太原理工大学保留使用毕业设计（论文）的规定，即：学校有权保留送交毕业设计（论文）的复印件，允许毕业设计（论文）被查阅和借阅；学校可以上网公布全部内容，可以采用影印、缩印或其他复制手段保存毕业设计（论文）。

论文作者签名：霍勐羽　　指导教师签名：魏振楠 汪琼

签字日期：2025 年 6 月 3 日　　签字日期：2025 年 6 月 3 日

基于 Spring Boot 的同城上门喂宠预约系统的设计与实现

摘　要

随着现代都市生活节奏加快，越来越多的宠物主因工作繁忙或临时外出而面临宠物照料难题。当前市场上出现一批宠物上门服务，但目前上门宠物服务大多依赖传统的中介平台或社交媒体，存在服务单一、流程繁琐、覆盖有限等痛点。本研究设计开发了一套宠物上门服务系统。基于 Spring Boot 技术架构搭配微信小程序，采用现代化的软件工程管理方法，确保系统具有良好的可维护性和用户体验。最终实现了一套高效的宠物服务预约系统，支持服务预约、订单管理、服务推荐等功能。该系统不仅满足了宠物主人的实际需求，还为宠物服务行业的数字化转型提供了可行的解决方案。具有显著的社会效益和商业价值。

关键词：Spring Boot；微信小程序；预约系统

Design and Implementation of Local Pet Care Scheduling System Based on Spring Boot

Abstract

With the accelerating pace of modern urban life, an increasing number of pet owners face challenges in pet care due to busy work schedules or unexpected trips. While a range of pet home services have emerged in the market, most current offerings still rely on traditional intermediary platforms or social media, suffering from issues such as limited service options, cumbersome processes, and restricted coverage.

To address these pain points, this study designed and implemented a pet home service system. Based on the Spring Boot technical architecture paired with WeChat Mini Programs, the project adopted modern software engineering management methods to ensure excellent maintainability and user experience. The final product is an efficient pet service booking system that supports functions including service reservation, order management, and service recommendations.

This system not only meets the practical needs of pet owners but also provides a viable digital transformation solution for the pet service industry. It demonstrates significant social benefits and commercial value.

Key words: Spring Boot; WeChat Mini Program; Scheduling System

目　录

1 绪论

1.1 研究背景及意义

随着我国经济的飞速发展，人们的生活节奏也在加快，工作与生活的分离愈发明显，宠物逐渐成为现代家庭中情感寄托的重要角色。相关数据显示，截至 2024 年，中国城镇养宠家庭已超过 7000 万户，宠物数量与日俱增，宠物经济呈现出快速发展的趋势[1]。在这种背景下，养宠不仅是一种生活方式，更是一种情感需求的体现。

另一方面，随着社会节奏的加快，宠物主人在日常生活中面临繁重的工作压力与时间限制，往往无暇顾及对宠物进行定时喂养、洗护、陪伴等基本照料，这不仅影响宠物的身心健康，也成为宠物主人长期困扰的问题。目前市面上虽存在部分第三方宠物服务平台，但多依赖于传统中介模式或社交平台信息发布，存在信息不对称、服务质量不可控、响应效率低下等问题，难以为宠物主人提供及时、高效、可信赖的服务支持，无法真正解决宠物日常照护的核心难题。所以开发一个适用性强的预约系统具有一定研究意义。

本论题的设计就是为了实现宠物服务的数字化建设，更好满足社区居民的快节奏生活需求，为我国数字化建设做出贡献。本系统的出现，由专业的宠物师对宠物进行照料，一方面满足居民的看宠需求，解放了宠物主人的双手，同时在一定程度上提升了居民的幸福度。小程序的预约系统可以保证预约的高效性，操作者的大部分工作可在手机上完成，宠物主只需在微信客户端操作即可完成预约。这样的预约系统能够满足用户的绝大多数需求。本系统数字化的工作方式能够最大程度帮助服务平台管理人员避免因工作量大、内容重复、效率低下的大量收集审核的劳动。减少了人力成本的同时又降低因人工行为导致的失误。

1.2 国内外研究现状

宠物服务属于家政行业，而国内外对家政服务平台的研究都在逐渐增多。从商业角度分析，目前家政行业有直营模式、经纪人直营模式、平台模式三种主要商业模式[2]。直营模式下，互联网作为中介平台，直接匹配用户需求。经纪人直营模式类似传统家政模式，经纪人担任中介，用户通过互联网寻找中介。平台模式主要是家政中介公司提供服务。其中，互联网特征更直观的是直营模式，需要整合更多资源，难度较大。

直营模式的家政平台有天鹅到家、轻喜到家、Handy 等。该模式下服务人员属于家

政平台，平台统一安排工具及工作。该模式采用去中介化的方式，用户使用手机客户端预约服务，平台通过需求分配服务人员。如天鹅到家，平台基于用户地理位置匹配服务人员，采用抢单模式，范围内的服务人员都可以查看订单并按照自身需求进行接单。其缺点也很明显，服务人员只考虑到自身需求，做出了局部最优的选择，可能导致整体效率低下，以及存在大量订单无人接单造成质量无法保证的问题。

经纪人直营模式最典型的平台如 58 到家。服务人员由平台认证的经纪人直接管理，经纪人负责招募、培训、调度及服务质量监督，平台则提供订单分发、工具支持及品牌背书。用户通过平台直接下单，节省在挑选服务人员所耗费的时间，但订单分配由经纪人介入，避免纯抢单模式的局部最优问题。但存在潜在问题：调度效率和服务质量高度依赖经纪人管理水平，若经纪人经验不足或负荷过重，可能导致派单延迟或纠纷。

平台模式最典型的是 2013 年在中国成立的云家政平台，为中小微家政公司提供免费的 SaaS 云平台，让线上和线下结合，线上信息交流，方便用户满足需求，让线下服务人员得到有利的工作机会，合理利用资源[3]。但也存在一些缺点，很难做到平台的标准化服务。

此外，学者刘艺溥认为当前供不应求的服务造成了家政行业的许多乱象和不健康的发展状况。为了防止一些恶劣事件的发生，从平台上加以安全防护是实现从业人员监督管理的有效方式。同时，随着互联网的飞速发展，人工智能技术渗入到了人们生活和工作的各个方面。人脸识别作为一种生物识别技术，已成为了计算机视觉领域的重要分支。学者刘艺溥设计并实现了一套基于人脸识别的家政服务平台，在传统家政服务平台上引入了人脸识别功能，并对人脸识别算法部分研究了基于深度网络的 VGGNet 和 ResNet 模型改进，并应用在家政平台[4]。

而对于微信小程序平台，已有大量对该应用进行预约平台的开发，如学者刘一澎认为随着互联网的兴起与发展，留学咨询的方式从传统的线下中介延伸至线上平台。虽然留学中介提供了一定的留学咨询服务，但其资讯的真实有效性得不到保证，且行业内还存在留学信息不对称及线下服务不规范等问题。开发了一套基于微信小程序的专家预约系统，满足学生对专家的筛选及预约服务的需求[5]。

微信小程序主要服务于中国大陆用户，相关的直接研究多集中在国内。但从技术架构和功能特性来看，微信小程序本质上属于移动互联网应用的一种形式，其核心理念与海外广泛研究的 Progressive Web App (PWA)等轻量级移动应用存在诸多相似之处。学者

Andreas Biorn-Hansen 将 PWA 与其他跨平台开发方法（如 React Native、Xamarin 等）进行了比较，两者均使用标准的 Web 技术（HTML、CSS、JavaScript），降低了学习成本；无需通过应用商店发布，用户可以直接从浏览器访问或安装；开发者可以随时更新应用，无需用户手动更新。然而，作者也指出，PWA 在访问设备硬件功能方面仍存在一定的限制，这可能影响某些特定应用的开发[6]。

1.3 本文主要研究内容

本论题在我国宠物服务平台研究和发展基础上，分析服务平台的建设和宠物从业者工作所遇到的问题，并对国内大部分在线预约平台进行分析，再结合国外预约系统的发展过程和未来发展方向，结合我国国情。利用软件工程专业知识和 Spring Boot 框架搭配微信小程序，设计了一款适用于我国国情的，功能相对完善的宠物上门服务平台。并完成了整部论文的撰写工作。

(1) 本论题采用 B/S 与 C/S 混合结构，开发了两个子系统，分别为用户端及管理端。用户端可进行预约等操作，管理端可进行订单管理等操作。

(2) 本论题主要工作是对预约系统的设计与实现。在设计时，严格遵守 Java 面向对象模式。根据我国宠物服务业发展现状，对预约系统进行了较为全面的需求分析，使用系统部署图和顺序图表达系统架构。在实现部分，采用前后端分离方式，即后端采用 Spring Boot 框架，前端采用微信小程序技术。使用 MySQL 数据库和 MyBatis-Plus 技术。在系统使用界面，用户只需在微信小程序进行可视化操作即可，降低系统的学习成本和使用门槛，以便更好地使用和普及。

(3) 在开发过程中，也进行了功能与非功能测试。编写功能测试用例，对平台主要模块的功能进行验证，同时也对系统响应时间和并发性能进行测试。

1.4 论文的组织结构

论文总分为 6 章，各章结构及内容概述如下：

第1章 绪论。本章首先阐述了宠物服务预约系统的开发背景，并根据目前国内外宠物服务预约系统的基本研究状况。对预约系统的基本思想及研究意义进行介绍，并对本课题的论文结构进行概述。

第2章 系统开发相关技术介绍。本章对系统开发过程中使用的开发工具进行了介绍，详细描述了该系统所用到的开发框架、相关技术等开发工具。并对这些框架的原理和使用方法进行了概述。最终确定了本系统开发所需的工具和相关技术支持等。

第3章 系统需求分析。对预约系统的需求进行全面分析与概述，包括功能性和非功能需求分析，之后分析了系统的开发可行性。

第4章 系统设计。对系统各个关键技术，前后端的设计进行介绍。其中重点设计了系统的注册与单点登录模块。之后介绍了数据库的设计。

第5章 系统实现。首先介绍了系统开发及运行环境，详细介绍了核心模块的实现。最后对系统进行功能与性能测试，并作出测试分析。

第6章 结论。总结了论文完成的工作以及存在的问题，并展望系统下一步研究工作。

2 系统相关开发技术介绍

宠物服务预约系统采用了微信小程序框架，本质上属于 B/S 结构。总体处于前后端分离的开发模式，业务逻辑主要发生在服务端，为了便于用户操作和对系统进行维护等操作，客户端载体为微信小程序。采用前后端分离开发模式好处可以避免双端高耦合，有效减轻服务端压力。

2.1 Spring Boot 框架

Spring Boot 是一个基于 Spring 框架的开源应用程序开发框架，通过预组装了 Spring 的一系列套件，提供自动配置和约定大于配置的方式，以便尽可能少的代码和配置来开发基于 Spring 的 Java 应用程序。Spring Boot 的运行原理是通过 Spring 框架的依赖注入和控制反转机制，以降低程序的耦合性[7]。

Spring 与 Spring Boot 之间的关系可以用如下架构图 2-1 描述：

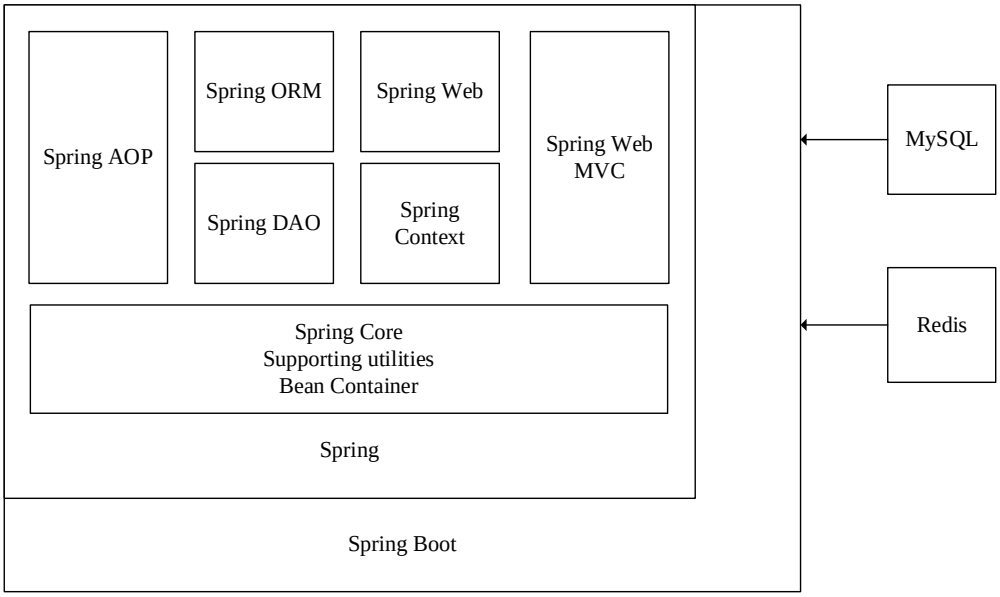

图 2-1 Spring Boot 架构图

2.1.1 Spring Boot 的发展

早期 Java Web 应用雏形是 Servlet，随着框架技术水平的不断提升，经过数次的更新，最终形成了相对完备的开源套件。在这些完善的套件中，Spring 框架克服了传统重量级框架臃肿、低效的劣势，极大简化了项目开发中的复杂性，受到开发者的青睐。

虽然 Spring 框架是轻量级的，但它的配置是重量级的。Spring 早期版本专注于 XML 配置，开发一个程序需要各种的 XML 配置文件。随着实际生产中敏捷开发的需要，以

及 Spring 注解的大量出现和功能改进，从 Spring 4.x 版本开始，项目开发以基本可以脱离 XML 配置文件进行。因此，在 Spring 中使用注解开发逐渐占据了主流地位。与此同时，Pivotal 团队在原有 Spring 框架的基础上通过注解的方式进一步简化了 Spring 框架的使用，同时基于 Spring 框架开发了全新的 Spring Boot 框架，并于 2014 年 4 月推出了 Spring Boot 1.0 版本，后在 2018 年 3 月推出了 Spring Boot 2.0 版本。Spring Boot 2.x 在 Spring Boot 1.x 的基础上进行了很多功能的改进和扩展，同时进行了大量代码重构[8]。2022 年 11 月，发布了 Spring Boot 3.0 版本，引入 Java 17 为最低版本，以便利用最新的语言特性和性能改进。3.0 版本还对大量依赖项进行升级，主要为增强系统安全性和提高应用可观察性。使得开发人员能够更加轻松的进行开发和部署。因此，Spring Boot 已成为了目前最受欢迎的 Java Web 开发框架之一。

2.1.2 Spring Boot 框架特点

Spring Boot 框架作为 Java 生态中现代微服务开发的核心工具，如前文所提到的，Spring Boot 框架能使得开发者摆脱大多数开发过程中涉及到繁琐的 XML 配置及注解工作。同时，由于 Spring Boot 的 starter 特性可以禁止重复依赖，所以开发过程中也无需担心版本冲突。Spring Boot 具有如下几个显著特点：

(1) 约定大于配置：Spring Boot 通过自动配置、默认值优化、简化 XML 方式，极大程度上降低了开发工作量。

(2) 快速启动，独立运行：Spring Boot 提供了一系列的起步依赖，内嵌了 Tomcat 服务器，无需部署 war 包。再通过打包生成可执行 jar 包，打包即服务。

(3) 无需代码生成：采用注解和约定的方式，减少代码编写量，避免代码生成的问题，比如代码重复，降低了维护难度。

(4) 多环境支持：项目内有多种配置文件，比如 application.yml，可根据不同环境配置不同参数。

总之，Spring Boot 是一个便捷的框架，可以帮助开发者搭建一个高效的应用程序。

2.1.3 Spring Boot 运行原理

Spring Boot 运行原理基于约定大于配置的核心思想，通过嵌入式容器技术，实现了传统 Spring 应用的部署和快速启动。其运行原理如下：

(1) 加载 Spring Boot 核心配置文件：Spring Boot 会加载默认的配置文件 application.yml。内含默认配置项，如 jwt 令牌、数据库连接信息等。

(2) 执行自动装配：Spring Boot 会从 AutoConfigure 中读取候选配置，将符合条件的配置类中的@Bean 方法注册到容器。

(3) 启动嵌入式容器：Spring Boot 会通过 ServletWebServerFactory 创建服务器实例，初始化 DispatcherServlet 并注册到容器，绑定配置文件中的端口号。启动应用程序。

整个初始化中，Spring Boot 读取核心文件，自动设置运行环境，根据程序类型准备容器，最后执行用户自定义的初始化任务，完成了整个启动过程。

2.2 微信小程序

小程序是一种全新的连接用户与服务的方式，它可以在微信内被便捷地获取和传播，同时具有出色的使用体验。主要开发语言是 JavaScript。与普通的网页开发有不少相似之处，但与传统前端开发相比，二者还是有些许区别的。

网页开发中，渲染任务和脚本任务是互斥的，这也会导致长时间的脚本运行可能会导致页面失去响应；而在小程序中，二者是分开的，分别运行在不同的线程中。网页开发者可以使用到各种浏览器暴露出来的 DOM API，进行 DOM 选中和操作。小程序的逻辑层和渲染层是分开的，逻辑层运行在不同于渲染层的独立 JS 运行时中，因此并不能直接使用 DOM API 和 BOM API。这一区别导致了前端开发非常熟悉的一些库，例如 jQuery、Zepto 等，在小程序中是无法运行的。同时逻辑层的 JS 运行时与 NodeJS 环境也不尽相同，所以一些 NPM 的包在小程序中也是无法运行的[9]。

网页开发者需要面对的环境是各式各样的浏览器，桌面端需要面对 Firefox、Chrome 等各种不同内核的浏览器，在移动端需要面对 Safari、Chrome 以及 iOS、Android 系统中的各式 WebView 。而小程序开发过程中需要面对的仅仅是两大操作系统 iOS 和 Android 的微信客户端，小程序中两大运行环境也是有所区别的，如表 2-1 所示。

表 2-1 小程序的运行环境

运行环境	逻辑层	渲染层
iOS	JavaScriptCore	WKWebView
Android	V8	Chromium 定制内核

2.2.1 小程序主要代码构成

微信小程序中有不同类型的文件，包含一个描述整体文件的 app 和多个描述各自页面的 page，主体部分由三个文件组成：

表 2-2 小程序主体文件

文件	是否必需	作用
app.js	是	小程序逻辑
app.json	是	小程序公共配置
app.wxss	否	小程序公共样式表

一个小程序页面由四个文件组成，分别是：

表 2-3 小程序页面文件

文件类型	是否必需	作用
json	否	页面配置
wxml	是	页面结构
wxss	否	页面样式表
js	是	页面逻辑

(1) JSON 配置文件：JSON 是一种数据格式，而非编程语言，在小程序中，JSON 用于扮演静态配置的角色。本文分析重要的 JSON 文件：小程序配置 app.json：是当前小程序的全局配置，包括了小程序的所有页面路径、界面表现、网络超时时间、底部 tab 等。工具配置 project.config.json：工具上做的任何配置都会写入到这个文件，当开发者重新安装工具或者换电脑工作时，只要载入同一个项目的代码包，工具就自动恢复到开发项目时的个性化配置，包括编辑器的颜色、代码上传时自动压缩等一系列选项。页面配置 page.json：用于配置小程序页面的一些基本属性，例如顶部颜色、是否允许下拉刷新功能。此外，JSON 文件无法使用注释，否则会引起报错信息。

(2) WXML 模板：小程序中 WXML 充当网页编程中的 HTML 的角色。且与 HTML 非常类似，WXML 由标签、属性等构成。但也有不同之处，例如 WXML 中标签使用 view，button，text 等，以及 wx:if 等表达式，用于变量绑定。

(3) WXSS 样式：具有 CSS 的大部分特性，同时也做了一些扩展与补充。比如新增尺寸单位 rpx，小程序环境可直接换算对应的屏幕像素比。新增了全局与局部样式，用于控制整体与部分的样式。

(4) JS 交互逻辑：用于处理用户操作。例如点击 button 按钮时，调用对应的逻辑以响应请求。同时也可在 JS 中调用小程序提供的 API，以便调起微信提供的能力，例如直接使用微信用户登录，获取手机号等。

2.2.2 小程序宿主环境

微信客户端提供给小程序的环境称为宿主环境。小程序借助宿主环境提供的能力，可完成一些平台网页无法完成的功能。

小程序的运行环境分为渲染层与逻辑层，其中 WXML 模板和 WXSS 样式工作在渲染层，JS 脚本工作在逻辑层。其中，渲染层的界面使用 WebView 进行渲染；逻辑层采用 JsCore 线程运行 JS 脚本。一个小程序存在多个界面，所以渲染层存在多个 WebView 线程，这两个线程会经由微信客户端做中转，渲染层发送的网络请求也会经由客户端转发，小程序的通信模型如下图 2-2 所示。

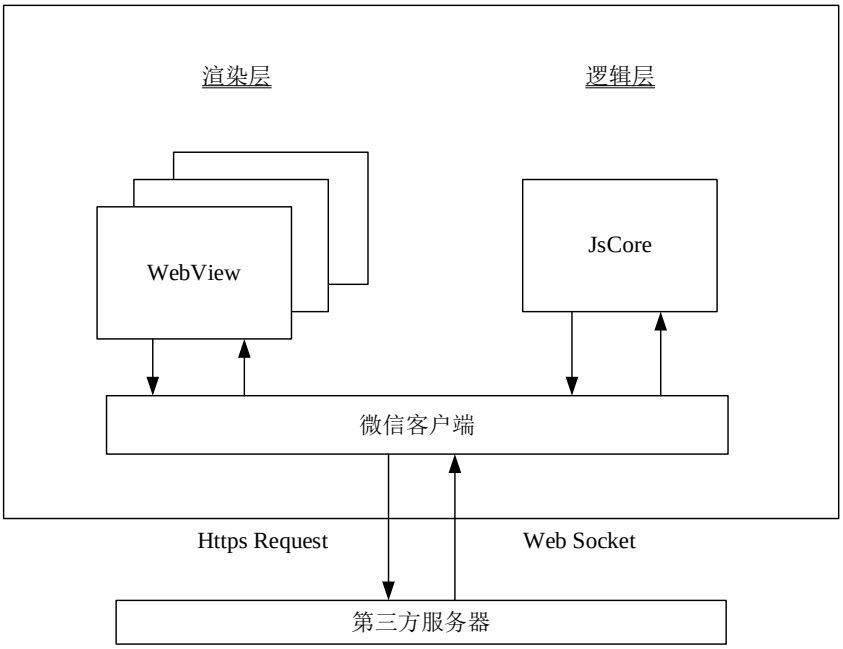

图 2-2 小程序通信模型图

2.3 数据库技术

数据库系统可追溯到 20 世纪 60 年代，计算机管理的对象规模越来越大，应用范围越来越广泛，数据量急剧增长，同时多种应用、多种语言互相覆盖地共享数据集合的要求越来越强烈。出现了统一管理数据的专门系统软件，即数据库管理系统。在 20 世纪 70 年代，关系型数据库管理系统出现，建立在严格的数学概念基础之上，使用 SQL（结构化查询语言）进行数据的查询与管理。在 20 世纪 90 年代，随着互联网的兴起，数据库技术得到广泛应用[10]。数据库技术涉及到更多领域，如数据仓库，数据挖掘，大数据等。同时，非关系型数据库（NoSQL）也开始出现，提供了一种不同于传统关系型数据库的数据存储和管理方式，如 HBase，专用于大数据的存储与管理。三次信息化浪潮，数据库的发展方向也从数据存储改变为更加高效的满足大数据处理和分析需求。

2.3.1 MySQL 数据库

本论题采用 MySQL 数据库，MySQL 是一种广泛使用的关系型数据库管理系统，

可在多个操作系统运行。而且支持多种语言，如 C++、Java、Python 等，因此在开发 Web 应用程序时，MySQL 也是明智的选择。

MySQL 凭借其高性能和可扩展的特性，可在各种生产环境下，使用各种引擎都可以高效运行。同时，其支持事务的特性，又保证了数据的可靠性。而且，其开源免费带来了庞大的社区支持，拥有活跃的开发者生态。

在选择 MySQL 作为数据库系统时，事务处理的 ACID 特性作为决策时的关键因素。MySQL 通过其灵活的事务处理机制和高效的并发控制策略，为处理高并发场景提供了有力支撑。同时，也关注数据库性能优化，针对特定查询进行索引优化，查询缓存策略的调整，选择了 InnoDB 作为存储引擎，以便最优利用服务器资源。通过这些策略，MySQL 可以有效地处理大批量并发请求，同时保证数据的一致性和完整性。

2.3.2 MyBatis 框架

MyBatis 是一种 Java 持久层框架，将面向对象语言中的对象与 SQL 语句耦合在一起，使用 XML 语言为主。框架源自 iBatis 团队，设计理念是将 SQL 语句与 Java 代码分离，使得开发人员能够更好的维护与管理应用程序。

MyBatis 核心组件由 Mapper 和 Configuration 组成。Mapper 负责 MyBatis 的映射，将 Java 对象映射到数据库表中，通常用 XML 或者注解的方式配置。Configuration 是 MyBatis 的配置对象，管理全局配置信息，如数据源和插件等。

本论题采用 MyBatis 的增强工具：MyBatis-Plus。在 MyBatis 的基础上只做增强不做改变，达到了简化开发、提高效率。启动即会自动注入基本的增删改查操作，直接面向对象操作，少量配置即可实现大部分操作，满足绝大多数需求。同时内置了代码生成器，采用 Maven 插件可快速生成 Mapper、Model、Service、Controller 层代码。优点包括灵活性、易用性、易与调试。是一种强大的 MyBatis 扩展工具，被广泛运用于各种类型的 Java 程序中。

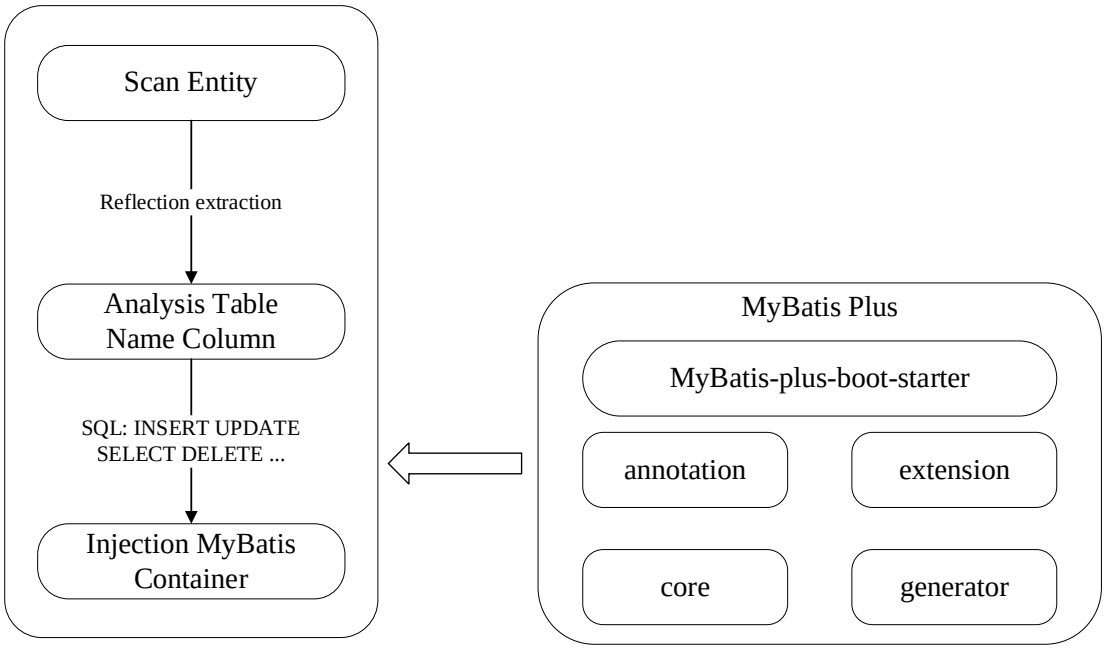

图 2-3 MyBatis Plus 框架图

2.4 本章小结

本章介绍了在开发预约系统过程中所用到的技术框架，并对这些框架的原理和优势进行了介绍，主要有：前后端分离开发技术、后端框架 Spring Boot、前端页面微信小程序、MySQL 数据库以及持久层框架 MyBatis。通过对各种技术的介绍，为后期项目开发提供理论铺垫和技术可行性基础。

3 系统需求分析

需求是用户对系统的要求，在开发过程中，首先要做的是完成对系统的需求进行分析。只有在明确了需求之后才能开展有针对性的软件开发工作，对软件需求的深入理解是软件开发工作获得成功的前提条件[11]。下面介绍系统的需求概述、功能性需求、非功能需求以及系统的可行性分析。

3.1 系统需求概述

近年来，国内宠物服务行业已逐步实现数字化转型，出现了多个成功的线上宠物服务预约平台。许多传统线下宠物服务开始向数字化服务转型，通过线上平台提供包括上门喂食、遛狗陪伴、宠物洗护等全方位的宠物服务。宠物服务数字化已取得显著进展。

然而，目前大多数宠物服务预约仍依赖社交平台或电话沟通，效率低下。特别是在工作日等高峰期，宠物主人常常面临预约难、服务响应慢等问题，而服务提供者则难以有效管理订单。这种低效的服务模式影响了用户体验，也制约了行业的规范化发展。

因此，本论题针对以上情况，作如下的需求分析。

3.2 功能性需求分析

本文旨在设计一个在线宠物服务预约系统，为宠物主人和宠物师提供预约平台。用户的两大主要角色为普通用户与管理员，其中管理员包含核销员（宠物师）、普通管理员和超级管理员。具体的用户角色权限如表 3-1 所示：

表 3-1 角色权限表

序号	角色	权限
1	普通用户	浏览平台界面
		用户注册登录
		个人信息修改
		创建和取消订单
		浏览、收藏、转发文章
2	核销员	订单查看及确认
		导出预约用户名单
3	普通管理员	用户信息管理
		文章新增与删除
		预约项目管理
4	超级管理员	管理端的最高权限

(1) 普通用户：是在平台进行注册登录后的用户，可以体验到完整的预约流程，具有以下权限：用户可以使用自己的账户名与密码登录系统。可以在进入系统后修改自己

的密码。在进入系统后，可以在个人资料页面将自己的姓名、地址、宠物信息进行修改。在系统首页，可以选择阅读文章，文章可以收藏、分享给自己的微信好友。首页还设置有服务推荐，用户可点击查看预约项目，在设置服务项目与时间后，可向核销员展示核销二维码进行核销。此时完成一次预约，用户可在个人中心页面查看已预约项目。

(2) 核销员：是线下直接面对宠物主人的角色，可更改自己的个人资料，查看预约项目的用户信息与核销操作。

(3) 普通管理员：管理员拥有系统的二级权限，可以管理预约中心，包括预约项目的发布与删除、用户预约名单的查看与导出、对广告文章的编辑与更新。也包括对所有注册过的用户进行查看、编辑、删除操作。管理员还可以对某些项目进行置顶操作，以加大用户访问。同时也包括对自身个人信息的编辑操作。

(4) 超级管理员：拥有管理端的最高权限，除上述管理员权限，还包括对管理人员的新建与删除。

在整体用例分析中，为了简化角色，直观展示系统用例，将核销员、普通管理员、超级管理员三者合为管理员一个角色。本系统用例图如下图所示：

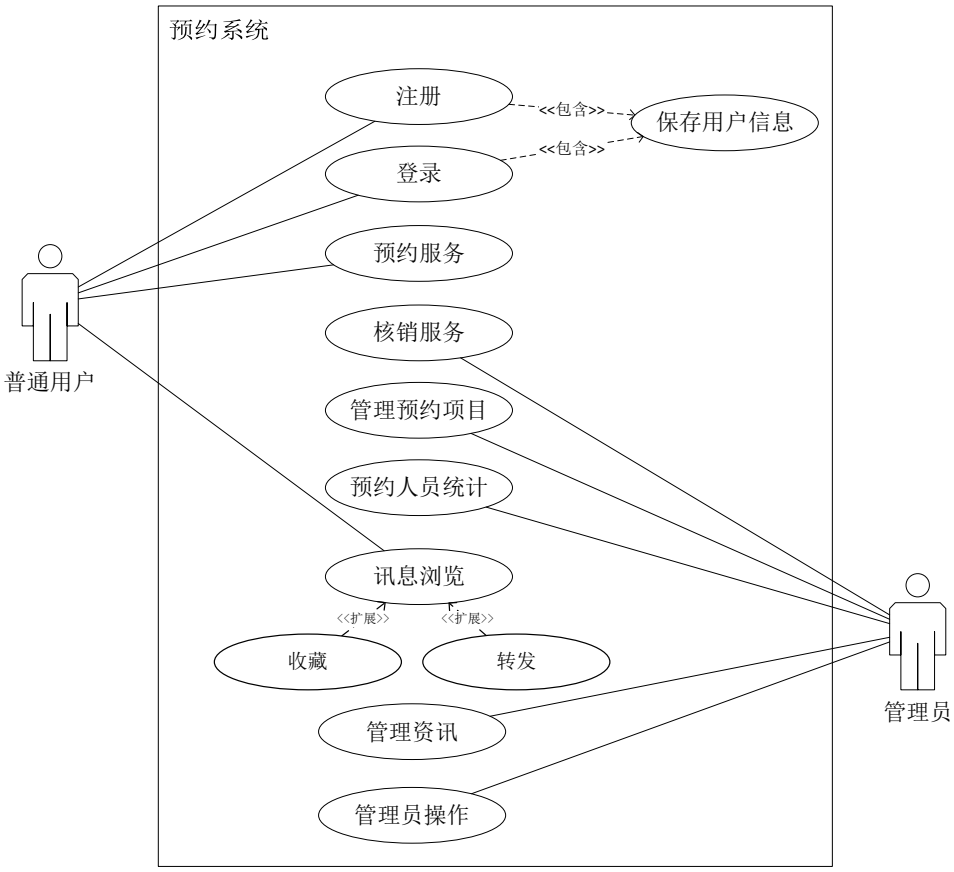

图 3-1 宠物服务预约系统用例图

普通用户是系统的主要用户，他们的目的主要是使用系统来预约宠物服务，次要目的是阅读新闻资讯。以下是普通用户的主要用例：

注册和登录：用户需要注册并登录系统才能使用系统的各项功能。

查看新闻资讯：用户可以对新闻资讯进行查阅、分享、转发。

宠物服务预约：用户可以阅览并预约服务项目，如喂猫、遛狗和洗护等。

系统管理员是系统的管理者，他们的目的是发布及管理预约项目。次要目的是撰写新闻资讯。以下是系统管理员的主要用例：

登录和权限管理：管理员需要登录系统并管理用户权限，以确保只有授权用户才能访问系统。

资讯管理：管理员可以管理公告和通知等信息，包括发布、修改和删除等。

预约项目管理：管理员可以管理各项预约项目，包括发布、修改和删除。

统计分析：管理员可以通过系统对预约人员的各种信息进行统计分析，如服务请求处理情况等。

3.3 非功能性需求分析

对于任何系统而言，系统的非功能需求与功能性需求同等重要，因为非功能性需求关乎到用户的使用体验和系统的稳定性。所以在满足上述功能性需求后，还需要满足如下的非功能性需求。

(1) 性能需求：系统需支持 100+并发用户同时操作，且页面响应时间不超过 2s，批量导出表格文件不应超过 10s。

(2) 安全性需求：用户密码采用加密存储，对数据进行保护，防止未授权的访问或恶意攻击。

(3) 可扩展需求：系统具有良好的可扩展性，能随着用户量增加而扩展功能。

(4) 可用性需求：系统应 7*24 不间断运行，避免出现停机问题。

(5) 可维护需求：系统具有良好的可维护性，保证系统的可靠性。

3.4 可行性分析

宠物服务预约系统是一个面向宠物主人和宠物服务提供者的微信小程序，旨在提高宠物服务行业的预约效率和质量。基于 Java 语言的广泛应用和微信庞大的用户数量，开发此系统有充分的可行性。下面从技术可行性、经济可行性以及操作可行性综合分析。

从技术可行性来看，基于 Spring Boot 和微信小程序的开发具备较高的可行性。

Spring Boot 作为成熟的 Java 框架，提供了自动配置、嵌入式容器等特性，能够快速构建稳定的后端服务，并支持高并发场景。微信小程序作为前端载体，具有跨平台、即用即走的优势，且开发工具链完善。系统采用的数据库 MySQL 为业界主流方案，技术风险较低。此外，RESTful API 设计和小程序的原生能力能够满足业务需求，技术实现路径明确。本系统后端程序运行在微软 Azure 云服务器，前端运行在腾讯微信小程序服务器，两者公司规模庞大，具有一定的安全性，也满足系统稳定、长期运行的需求。

从经济可行性来看，该系统开发成本可控，具备良好的投入产出比。开发所需的技术栈均为开源或免费方案，降低了软件授权成本。系统上线后，可通过广告投放等模式实现盈利。同时，数字化管理能够显著降低传统宠物服务的中介成本和信息不对称带来的效率损失，提升整体行业效益。考虑到宠物经济持续增长的市场趋势，该系统具备长期运营的商业价值。

从操作可行性来看，本系统设计简洁的 UI 界面，使宠物主人可以完成快速注册到预约，服务提供者也可轻松使用管理端对服务项目、预约人员进行管理。此外，微信小程序的设计，对于用户友好，无需下载 App，仅需扫描小程序码即可运行本系统，符合用户随时预约的习惯。

3.5 本章小结

在本章中，对预约系统的需求进行全面分析与概述。首先对预约系统进行总体概述，以确定系统所需的功能与角色。之后分析了系统的用例，以确定系统的角色权限。这些用例包括用户注册模块、预约模块、最新通知模块等。在确定用例后，进一步确定了各个用例的具体功能。此外，还分析了系统的非功能需求，包括系统的性能与安全性等方面，以满足用户的需求与增强用户满意度。最后，针对系统的可行性进行了分析，得出本系统在各方面均具备可行性，具有广泛的应用前景。

4 系统设计

上一章已完成对预约系统的需求分析，接下来根据各个模块的需求分析进行相应的功能设计，并针对主要功能点进行详细阐述，具体说明每个模块的操作逻辑和设计方式。

4.1 系统概要设计

预约系统本质是 B/S 架构的软件系统，但由于微信小程序的特殊性，也有部分 C/S 架构的特性。由三个主要组件组成：微信小程序、Web 服务器和数据库服务器。这种架构使用户可以轻松使用微信小程序访问系统，并与服务器进行通信。Web 服务器充当中介角色，接受来自微信小程序发来的请求并转发至数据库服务器进行处理。而数据库负责存储和管理系统中所有的数据。这种架构优点在于可以实现跨平台的兼容性，无论是 Android 或者 iOS 设备，仅需微信即可完成所有工作，并且提供高度稳定性，也保证用户数据得到保护。

4.1.1 系统架构设计

本系统的总体设计在于保障各个模块功能相互之间协调工作，以实现系统的功能特性，图 4-1 为系统的总体结构，共有五个层次：

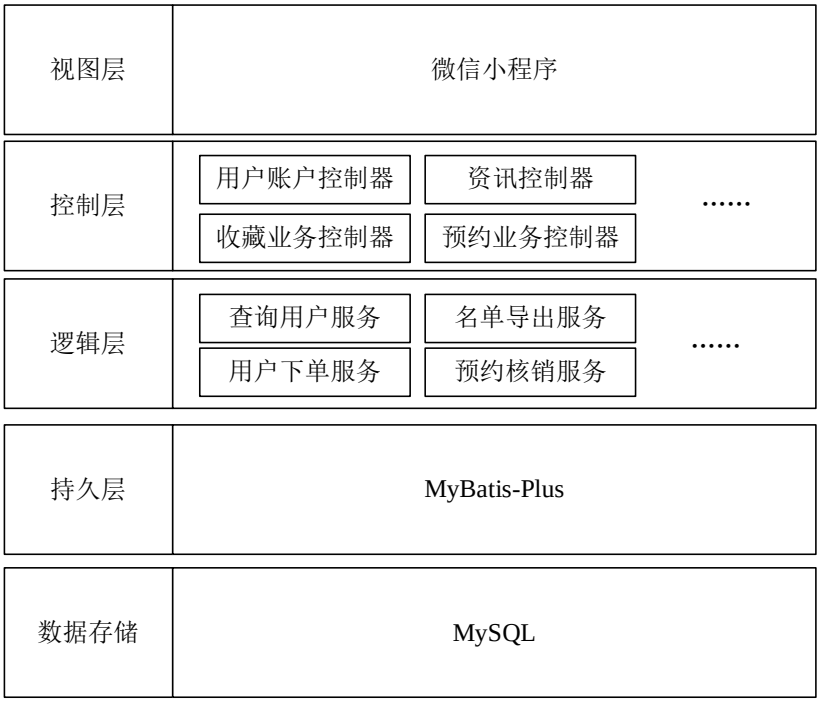

图 4-1 预约系统总体结构

视图层负责微信小程序前端页面，主要处理用户请求和响应，将用户操作转化为控

制层的指令。控制层负责处理前端传来的响应，将数据转化为逻辑层可以处理的格式，如用户账户控制器、资讯控制器、收藏业务控制器等。业务逻辑是整个系统的核心，处理所有的业务，如密码校验、用户管理、扫码核销验证等，主要是数据的计算以及存储。持久层用于数据的存储与访问，将 Java 系统与数据库联系，存储例如用户账户信息等数据，也包括写入、更新数据。

这五个层次相互独立，各自有不同的工作任务，通过接口进行通信。这种分层使得系统易于扩展而且维护性强。例如后续若想新增功能，仅需在相应逻辑层中添加新代码，无需修改他层代码。而且这种分层开发便于不同开发者负责相应的层次，提高了开发效率和代码质量。

4.1.2 系统功能模块设计

本系统分为用户端和管理端。用户端面向宠物主人，提供从注册登录、宠物信息管理、服务预约等操作。管理端则为运营方提供了全面的后台管理功能，包括服务项目管理、预约订单处理、用户信息维护、资讯内容发布等操作。

图 4-2 展示了系统用户端的具体功能结构。

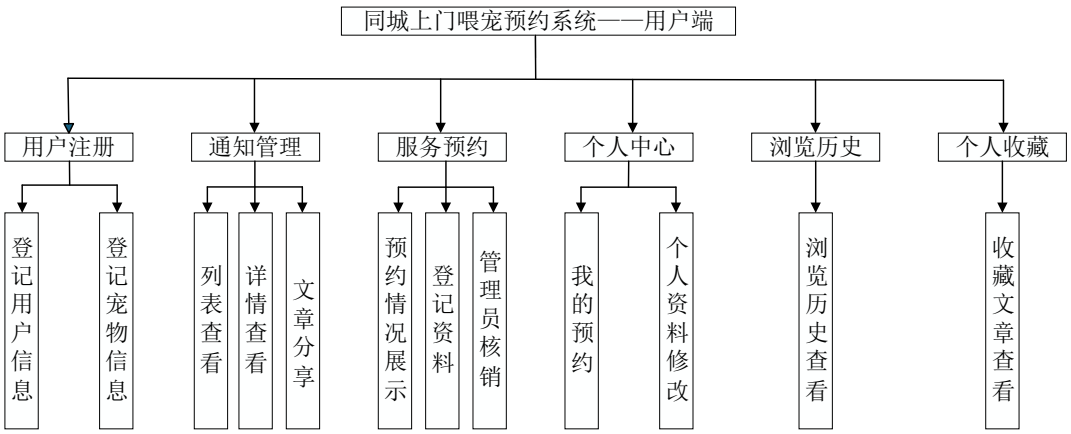

图 4-2 预约系统用户端

(1) 用户注册模块：用户在此页面输入个人用户名与密码，并且完善一定的个人资料，如姓名、宠物类型、住址。

(2) 通知管理模块：用户可以在列表查看文章简介，点击后阅读由管理员撰写的文章，可以分享给微信好友。

(3) 服务预约模块：作为系统核心模块。用户在注册模块登记宠物详细信息后，在服务列表页面，用户可以根据宠物类型选择合适的服务。预约流程支持精确到 1 小时的

时间段选择。预约完成后，待服务人员上门核销，用户出示核销码，完成本次服务。

(4) 个人中心模块：集中管理用户的各类数据和操作。用户可以在此修改个人资料、管理常用地址，并查看所有历史预约记录。订单状态实时更新，支持紧急取消功能，服务完成后用户可对服务进行评价。

(5) 浏览历史模块：自动记录用户查看过的服务和文章，方便快速找回内容。

(6) 个人收藏模块：允许用户收藏有价值的养宠知识文章。

图 4-3 展示了系统管理端的具体功能结构。

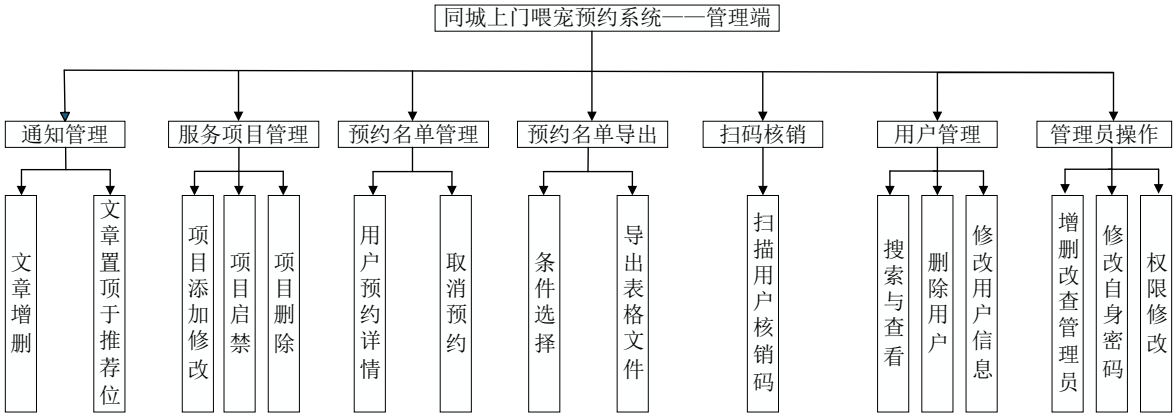

图 4-3 预约系统管理端

(1) 通知管理模块：管理员可以撰写通知文章，支持上传图片以丰富内容，同时，对于已发表的文章，可以进行置顶到推荐位，以增加曝光度。如果遇到不满意之处，也可将文章进行删除。

(2) 服务项目管理模块：预约系统的核心功能。管理员可以对项目进行新建，创建项目后，可根据实际需要对服务项目进行启禁操作，被禁止的项目不可被预约，也可对项目进行删除操作。

(3) 预约名单管理模块：管理员可以查看已预约用户的资料，包括服务时间和项目，也可对已预约的用户进行取消操作。

(4) 预约名单导出模块：管理员可对筛选某一时间段所有预约服务的用户，并可以导出为 xlsx 表格文件进行查看。

(5) 扫码核销模块：核销员（宠物师）在到达预定地点后，通过扫描用户端独一无二的 QR Code 进行核销。

(6) 用户管理模块：管理端可以搜索符合条件的用户，修改或者删除某一用户信息。

(7) 管理员操作模块：超级管理员可以对低权限管理员进行增删改查操作，也可以

修改自己的密码，还可以对低权限管理员升级或降级权限操作。

4.2 系统详细设计

本部分挑选系统部分模块进行详细设计，如用户注册和登录模块：单点登录服务完成用户登录、登出和 token 校验的功能；预约模块作为系统主要提供服务的模块，包括用户预约、管理员对预约项目的增删改查、用户管理、扫码核销功能。简要介绍非核心功能模块。

4.2.1 注册及登录模块

用户需先在个人中心页面进行注册。为保护用户信息，用户密码将采用 MD5 相对安全的加密方式存储于数据库中。用户点击注册按钮后，查询数据库中是否存在该用户名，如果找不到匹配的条目，在数据库中插入新的数据，并且跳转至登陆页面。注册业务流程设计如图 4-4 所示：

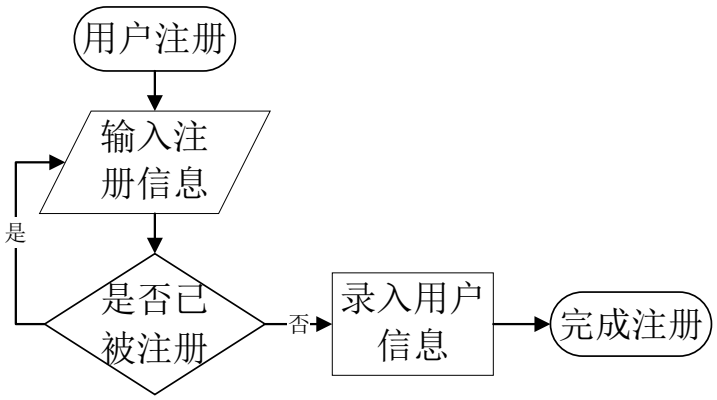

图 4-4 用户注册流程

用户注册以及登录功能的业务流程包括账户控制器 PassportController、账户服务 PassportService 等。在注册时，需要传入 account、name、password 等参数。微信小程序先在前端使用 js 检验各项参数是否合规，以及两次密码是否一致，通过后传入控制器，控制器再次确认各项参数是否符合数据类型，合规后调用 service 中的 register 方法。在 service 中，首先将传来的 account 参数调用 mapper 中 exists 方法检测是否存在，若存在则报告“该用户名已经存在，请更换!”给前端，拒绝注册。若不存在则将各项数据调用 Mapper 的 add 方法存入数据库中，在存入时，将 password 参数使用 md5Hex 方法重新赋值，完成本次注册。注册时序如图 4-5 所示：

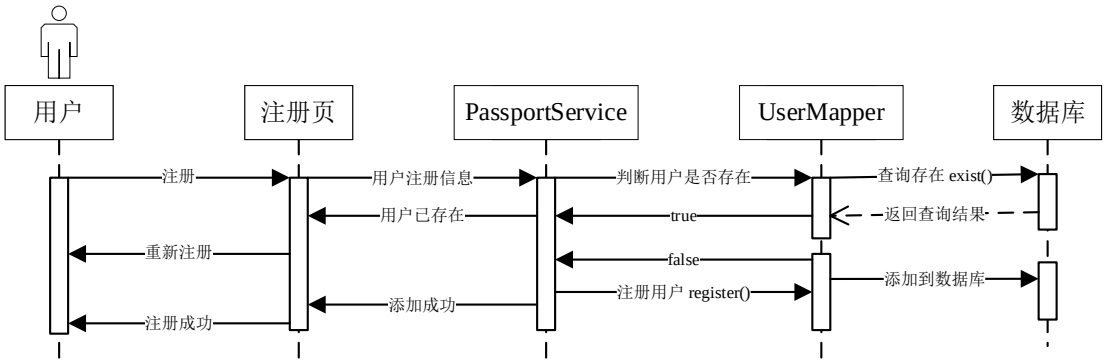

图 4-5 用户注册时序图

用户在完成注册操作后，便会自动登录，无需再次登录。而对于用户退出系统后再入系统，需要登录后才能访问各项功能。对于登录模块的设计，集成 CAS(Central Authentication Server) 实现了 SSO(Single Sign On) 单点登录。CAS 为本系统提供用户登录功能和内部服务 API 保护功能。确保用户一次登录即可使用所有关联服务，同时也保护系统内部 API 免受未授权访问。CAS 单点登录的时序如下：

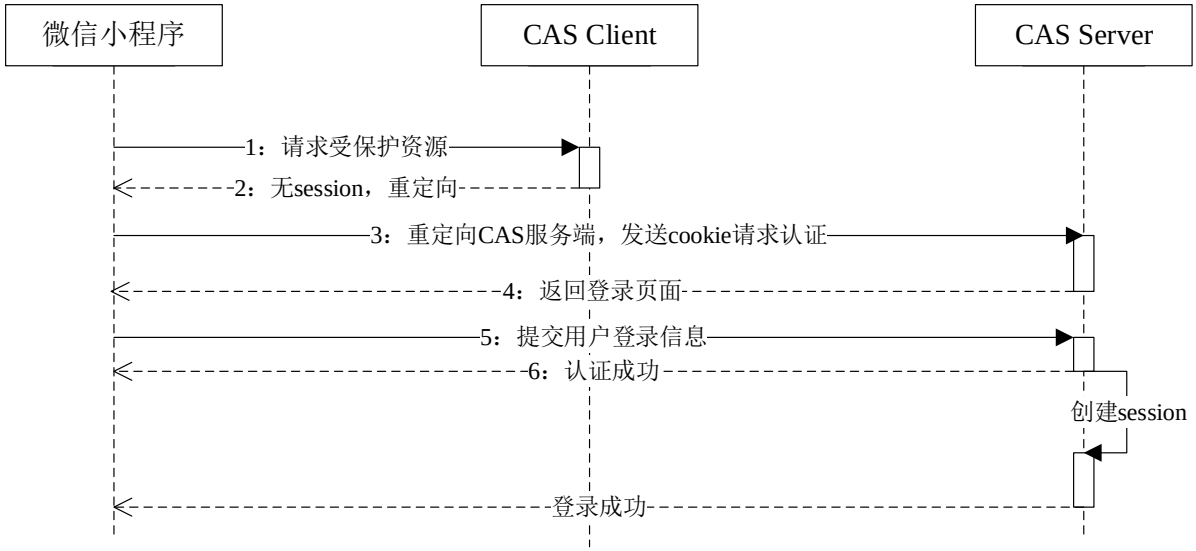

图 4-6 CAS 时序

(1) 登录：客户端要访问系统的功能，首先要与单点登录服务进行身份校验，校验方式使用传统的用户名和密码方式。用户名和密码校验通过后生成一个全局唯一且加密的 token，此 token 作为当前登录用户在当前客户端上的令牌，用于进行 API 访问权限校验。登录成功后向客户端返回用户信息和 token，客户端访问系统内其他服务有权限控制的 API 时需要携带此 token 信息。登录流程如图 4-7 所示：

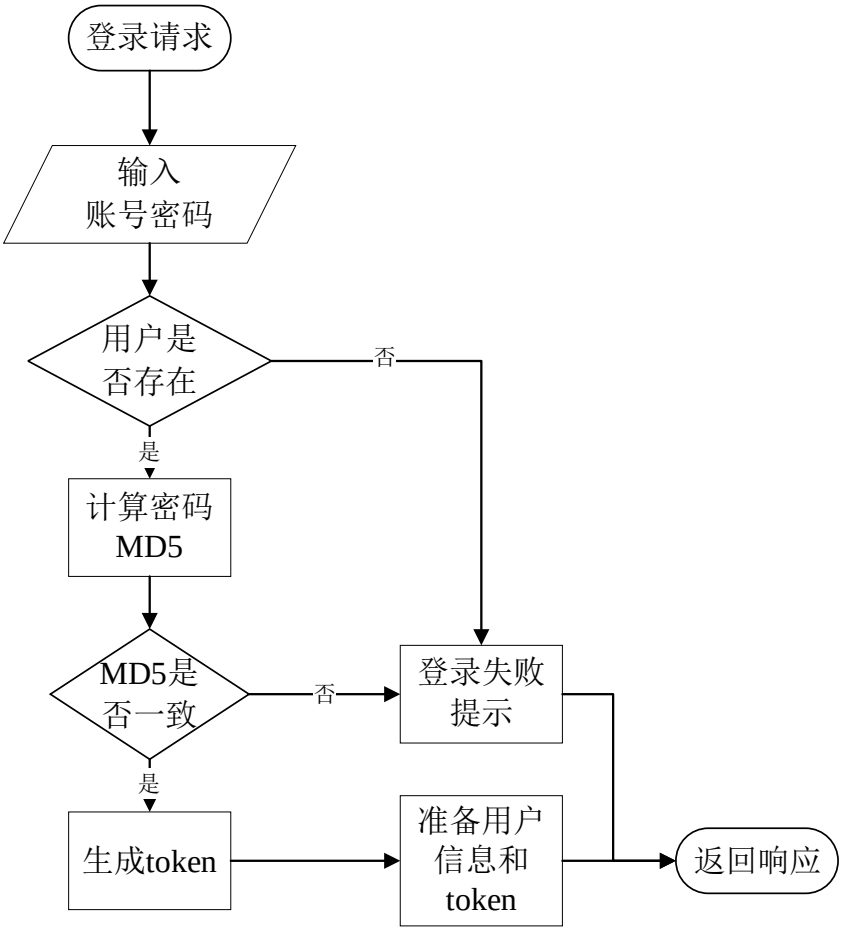

图 4-7 用户登录流程

(2) token 校验：系统内的其他服务如果需要对用户的操作进行确权：读取用户请求中包含的 token 信息，向单点登录系统询问此 token 是否有效；此时单点登录系统对 token 进行校验，若 token 对应了一个用户登录状态，则向其他服务返回此 token 对应的用户信息，若 token 无效则通知其他服务此 token 无效。Token 校验过程如图 4-8 所示：

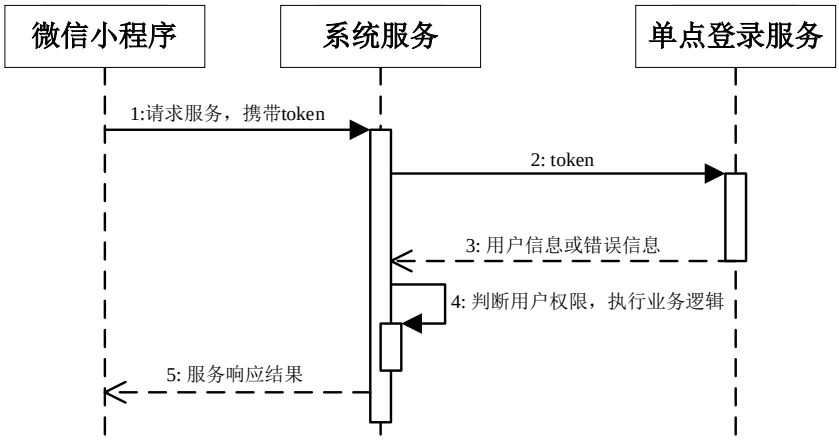

图 4-8 token 校验时序图

(3) 登出：提供了注销接口，由用户点击按钮，使得 token 失效以保证账户安全。

4.2.2 项目浏览模块

无论用户是否注册，打开小程序都将会跳转到小程序首页。首页展示有系统公告、服务项目、推荐服务。用户在此页面可点击浏览文章或项目。首页控制器是 HomeController，通过调用 Service 中的 getHomeList()方法实现列表查看项目，该方法通过返回 Map 来获取数据。例如首页有四个不同的服务类型：上门喂猫，上门遛狗，上门洗护和其他服务，按照四种服务的属性值分类，以进入不同的服务类型。

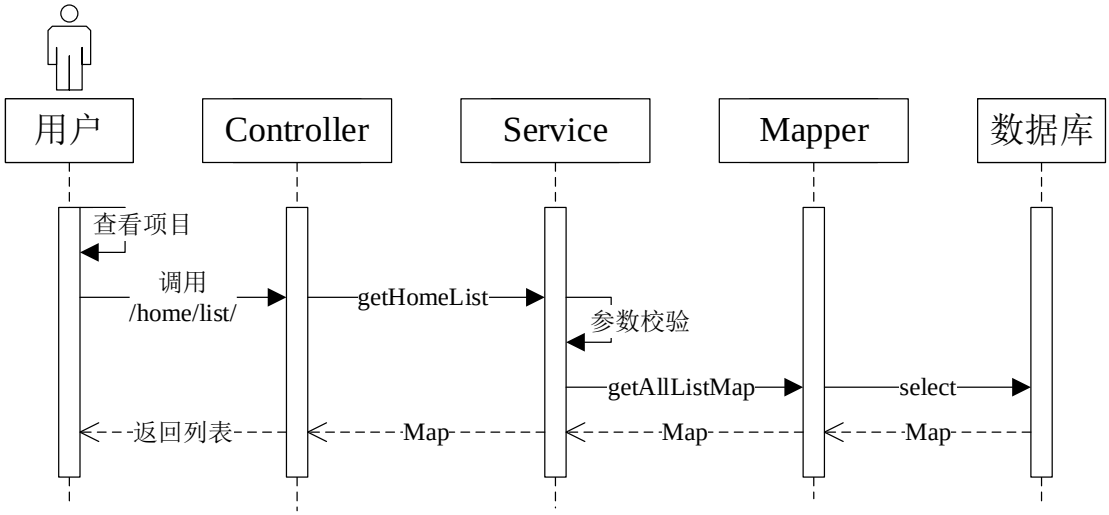

图 4-9 项目浏览时序

4.2.3 服务预约模块

用户端服务预约模块控制器是 MeetController。用户点击服务项目后，会弹出该项目的详情页面，内含预约须知与可预约时段，该服务项目由管理端在后台设置。用户只能在在未过期的日期选择服务项目，即仅可预约当日及当日之后的服务，不可预约之前的服务。系统使用 System.currentTimeMillis()获取当前时间戳，并转为“yyyy-MM-dd”格式，与数据库中服务项目的持续时间作比较，若该时间戳超出某一段项目时间，则不再展示，仅展示在当前时间戳之后的项目。若通过非法手段更改日期使得项目出现并且预约，Mapper 将抛出日期异常。用户点击时段后进入下单页面，用户主动传入 name 与 phone，经由 Service 将该数值打包为 JSON 数据格式传给 Mapper 存入数据库的 OBJ 列中，以便后续数据导出。此外，系统为每个预约服务生成了一个 15 位长的随机字符串，用于产生核销二维码。此为完成一次预约。用户预约简要流程如下：

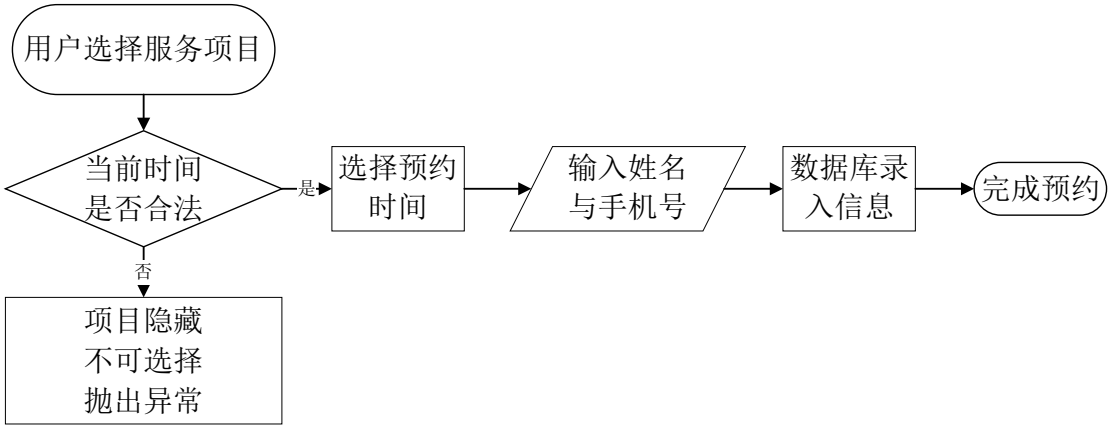

图 4-10 用户预约简要流程

用户在预约后，点击底部预约日历，找到对应日期可查看到当天的预约项目。也可在个人中心页面查看所有预约，该页面依然采用 list 方法展示，同上文浏览模块，此处不再赘述。点击预约项目后，调用 Controller 中的 getMyMeetJoinDetail()方法，用户可在此处将日程加入到手机日历中，调用了小程序的 bindTap 功能 wx.addPhoneCalendar，在 Android 与 iOS 发起默认日历 App 调用，将活动添加到日程中。下方展示的 QR Code 源自数据库存储的 15 位字符串，将调用开源的 weapp-qrcode 工具，将字符串转为 QR Code，管理端进行扫码核验。在扫码正确完成后，完成一次服务。

在扫描用户端的 QR Code 时，会调用控制器中 checkinMeetJoin()方法，检验此处扫描的 QR Code 与用户数据库中的字符串是否对应，给出相应的结果。该模块设计了双重核销机制，既支持后台管理员手动核销，也提供扫码核销功能。每次核销操作都会记录精确的时间戳，并严格校验预约记录的状态，防止重复核销等异常情况。系统还能智能识别带#分隔符的日期范围查询，方便管理员快速筛选特定时间段的报名记录。这些功能通过多重状态校验和事务性操作，保障了核销流程的可靠性和安全性。

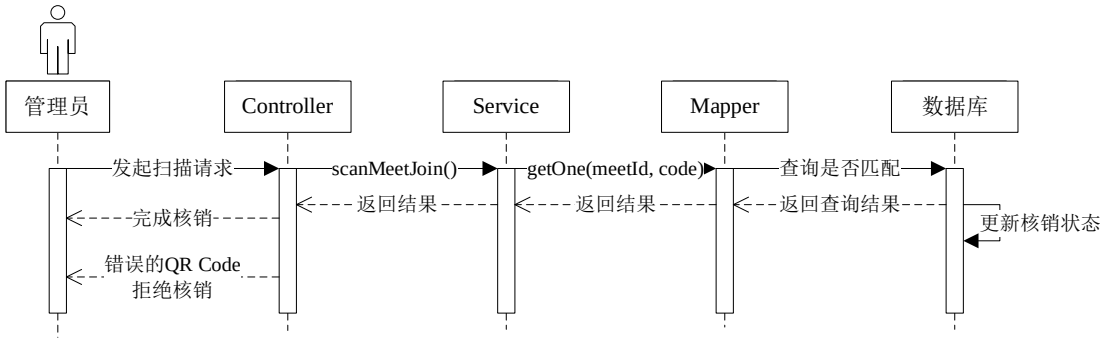

图 4-11 扫码核验时序

上文提到了用户端的完整预约操作以及管理端扫码核销的实现，以下将设计管理端对服务项目的管理操作。

服务项目的管理，是宠物预约系统的核心业务处理层，本模块使用分层架构和模块化设计。在数据访问层，基于 MyBatis-Plus 框架进行封装，通过 Where 条件构造器动态生成查询语句，结合 UpdateWhere 对象实现字段级更新，既保证了查询灵活性又避免了全表更新风险。分页查询采用 PageParams 参数封装，与 PageResult 结果集构成页。

业务处理层实现了宠物预约活动的全生命周期管理。在项目创建环节，系统会先检查标题是否重复，避免数据冗余；创建后进入编辑日期操作，会保留原有活动日期等核心配置；此外，状态变更则会联动影响前端展示和用户能否预约。这些操作都通过条件校验和字段控制，确保业务数据的准确性和完整性。

此处以添加项目为例：管理员在管理端的“添加预约项目”功能中，通过填写标题、选择服务分类、设置排序号等操作，可以发布一个服务项目。在发布项目时，可上传图片文件以丰富内容。

该部分表单较多，因此采用了 TPL 模板引擎来创建表单，这些表单全部存放于 meet_form_tpl.wxml 中。在填写所有信息后，点击提交，微信小程序将所有数据预先做合法性检验并打包为 Map 格式，调用 Controller 中的 insertMeet()方法，将数据进行传入，再次进行后端合法性检验。通过检验后，调用 Service 的 insert 方法，将通过校验的数据传入。即完成了预约项目的初步创建。此时回到管理员主页，点击项目管理，将预约项目设置日期。日期设置完毕后，在用户端即可正常进行预约操作。

综上，本系统预约模块的总流程如下图所示：

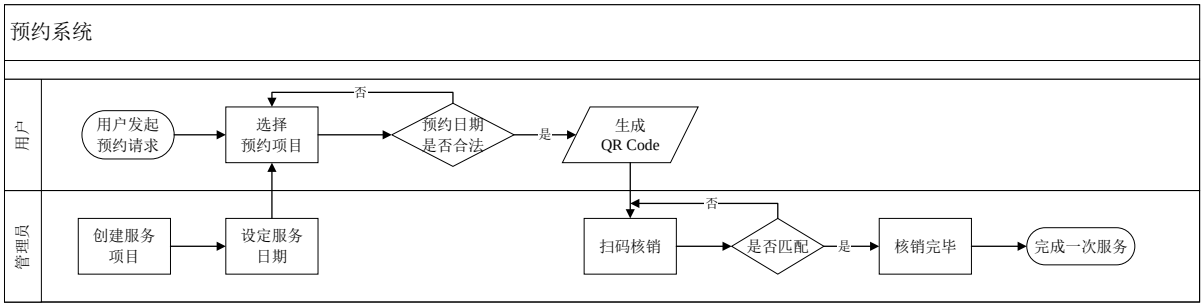

图 4-12 预约流程

在安全控制方面，系统实现了多层次的防护机制。继承自 BaseMyAdminService 的权限校验能力确保操作合法性，关键业务操作都包含存在性验证，敏感字段会自动脱敏

处理。业务规则上设置了排序值范围限制、核销防重校验等约束条件，有效防范异常操作。这些措施共同构建起系统的安全防护体系。

系统通过配置化的查询参数支持多维度复合查询，FileHelper 抽象层实现文件服务的统一管理，完善的日志记录为问题追踪提供依据。

4.2.4 数据导出模块

本系统的数据导出模块支持将用户数据和预约项目数据以文档文件 Excel 导出，便于管理员进行数据分析。下图为导出模块的工作流程：

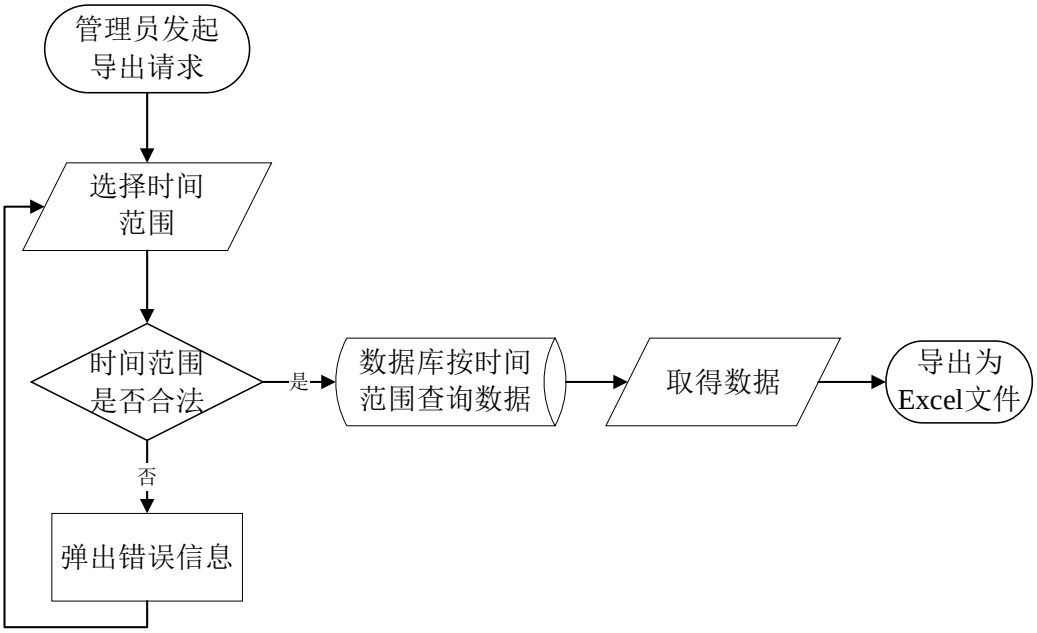

图 4-13 数据导出流程

首先，前端发起导出请求，调用 Controller 层的 exportMeetJoinExcel 接口，传入预约 ID（meetId）、开始日期（start）和结束日期（end）的 JSON 参数。Controller 接口会对这三个参数进行严格校验，确保格式正确且符合业务规则。

通过校验后，请求进入 Service 层的同名方法。该方法首先根据项目配置确定文件存储路径，使用 Excel 工具初始化一个包含 7 列的工作表，设置好"序号"、"姓名"、"手机"等表头，并调整各列宽度。接着构建数据库查询条件，筛选指定预约 ID 且在选定日期范围内的预约记录，并按日期和 ID 降序排列。

考虑到数据量可能很大，系统采用分页查询机制，每次从数据库获取 100 条记录。对于每条记录，会提取其 JSON 格式的表单信息，解析出姓名、手机号等字段，连同预约日期、时间段、添加时间以及签到状态（已签到/未签到）一并写入 Excel 行。导出时

动态生成包含 7 列数据的 Excel 表格，自动从 JSON 格式的 meetJoinObj 字段中提取用户姓名和手机号等关键信息，文件存储路径通过 ProjectConfig 统一配置。处理完当前页数据后，继续获取下一页，直到所有符合条件的记录都处理完毕。

全部数据处理完成后，关闭 Excel 写入器，生成文件的访问 URL。最后将包含下载链接和总记录数的结果返回给 Controller，再由 Controller 封装成统一格式的 API 响应返回前端。

4.2.5 其他非核心模块

(1) 资讯模块用于发布文章，基于 MyBatis-Plus 框架实现了资讯数据的全生命周期管理。在数据持久化方面，通过继承 BaseMyAdminService 获得基础数据访问能力，并注入 NewsMapper 实现具体的数据库操作。新增资讯时，系统会先检查标题是否已存在，避免数据重复，然后将资讯状态默认为 NORMAL 后入库；修改资讯时同样进行标题查重校验，并通过 UpdateWhere 对象实现字段级更新，避免全字段覆盖。

资讯查询功能支持多条件组合检索，包括基于关键字的模糊搜索（同时匹配标题和内容对象）、按分类 ID 筛选、按状态过滤等。排序方面支持默认排序（按 ORDER 字段升序和 ID 降序）和自定义排序规则，满足不同场景下的展示需求。分页查询采用 MyBatis-Plus 的分页插件实现，与统一的 PageParams 参数和 PageResult 结果集配合，确保分页数据的高效获取。

状态管理功能提供了资讯的上下架控制（statusNews）、排序权重调整（orderNews）和首页推荐位设置（vouchNews）。这些操作都通过 UpdateWhere 实现精准的字段更新，避免不必要的全字段修改。在删除资讯时，系统会先验证记录是否存在，防止无效操作。

整体来看，系统提供了完整的资讯管理能力。使用 Where 和 UpdateWhere 构建器实现灵活的查询条件和更新操作，结合 MyBatis-Plus 的分页功能，在保证功能完备性的同时兼顾了性能表现。异常处理方面通过 appError 方法抛出业务异常，由统一异常处理器捕获处理。

(2) 用户收藏模块：基于 MyBatis-Plus 框架构建数据访问层。在收藏状态判断方面，通过组合用户 ID 和对象 ID 构建查询条件，使用 exists 方法高效检查收藏关系是否存在。收藏更新操作部分，当检测到已收藏时执行取消操作，未收藏时则新建记录。

收藏数据管理方面，实现了完整的增删改查操作。新增收藏时封装了标题、对象 ID、类型和路径等信息，便于后续查询展示；取消收藏则根据用户 ID 和对象 ID 精准删除

记录，返回影响行数供业务判断。我的收藏列表查询采用分页机制，默认按收藏 ID 降序排列，并优化了返回字段只包含必要信息，避免不必要的数据传输。

模块通过继承 BaseMyCustService 获得基础数据访问能力，所有数据库操作都通过 FavMapper 实现。状态变更操作不返回复杂结果，而是通过布尔值直接反映操作后的状态，简化了接口理解成本。

4.3 数据库设计

4.3.1 数据库概要设计

本部分使用实体关系图（E-R 图）反应数据库中实体之间的关系。但系统各表数据繁多，很难直接、完整地抽象 E-R 图的实体关系。所以系统的核心实体关系如下：

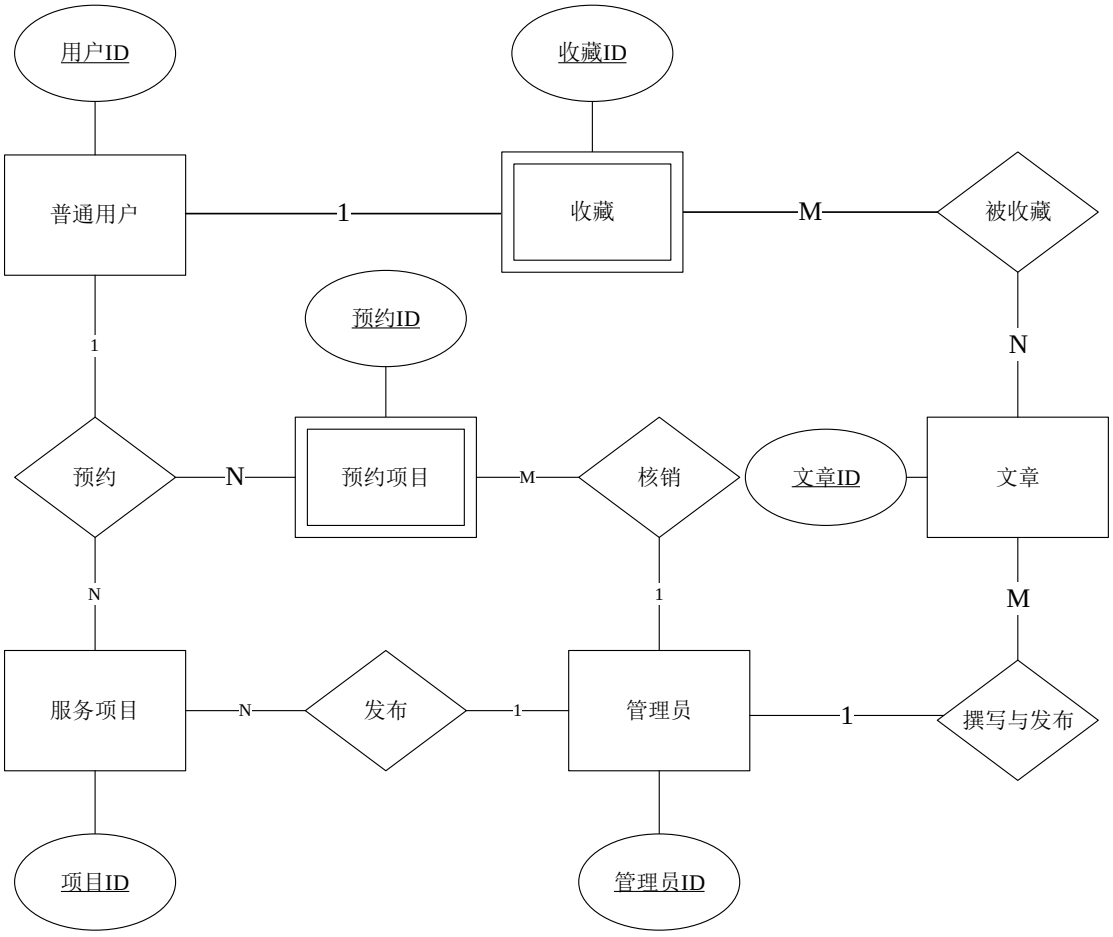

图 4-14 系统实体关系图

4.3.2 数据库逻辑设计

根据上节 E-R 图，对数据库逻辑设计如下：

用户（用户 ID, 用户名, 账号, 状态, 密码, 地址,登录时间, 登录次数, OBJ, FORM,

添加时间, 编辑时间）

管理员（管理员 ID, 用户名, 密码, 描述, 电话, 登录次数, 类型, 状态, 登录时间, 添加时间, 编辑时间）

文章（文章 ID, 标题, 分类 ID, 分类名称, 状态, 排序值, 是否推荐, 内容, 浏览次数, 图片, OBJ, FORM, 添加时间, 编辑时间）

收藏（收藏 ID, 用户 ID, 标题, 类型, 文章 ID, 路径, 添加时间, 编辑时间）

项目（项目 ID, 标题, 分类 ID, 分类名称, 状态, 排序值, 是否推荐, 活动天数, 浏览次数, 最大人数, FORM, OBJ, 添加时间, 编辑时间）

预约（报名 ID, 用户 ID, 项目 ID, 核销码, 是否核销, 核销时间, FORM, OBJ, 状态, 预约时间, 预约日期, 项目标题, 添加时间, 编辑时间）

4.3.3 数据库物理设计

系统使用 MySQL8 数据库进行构建，数据库结构设计创建了系统所需的数据表。

表 4-1 用户表

字段名	类型	长度	主键/外键	允许为 NULL	说明
USER_ID	int		主键	否	用户 ID
USER_NAME	varchar	255		否	用户名
USER_ACCOUNT	varchar	255		否	账号
USER_STATUS	int			否	状态
USER_PASSWORD	varchar	32		否	密码
USER_LOGIN_TIME	bigint			否	最后登录时间
USER_LOGIN_CNT	int			否	登录次数
USER_OBJ	longtext			是	OBJ
USER_FORMS	longtext			是	FORMS
ADD_TIME	bigint			否	添加时间
EDIT_TIME	bigint			否	编辑时间

表 4-2 管理员表

字段名	类型	长度	主键/外键	允许为 NULL	说明
ADMIN_ID	int		主键	否	管理员 ID
ADMIN_NAME	varchar	200		否	管理员用户名
ADMIN_PASSWORD	varchar	255		否	管理员密码
ADMIN_DESC	varchar	255		是	管理员描述
ADMIN_PHONE	varchar	255		是	管理员电话
ADMIN_LOGIN_CNT	int			否	管理员登录次数
ADMIN_TYPE	int			否	管理员类型
ADMIN_STATUS	int			否	管理员状态
ADMIN_LOGIN_TIME	bigint			否	最后登录时间
ADD_TIME	bigint			否	添加时间
EDIT_TIME	bigint			否	编辑时间

表 4-3 文章表

字段名	类型	长度	主键/外键	允许为 NULL	说明
NEWS_ID	int		主键	否	文章 ID
NEWS_TITLE	varchar	255		否	标题
NEWS_CATE_ID	int			否	分类 ID
NEWS_CATE_NAME	varchar	255		是	分类名称
NEWS_STATUS	int			否	状态
NEWS_ORDER	int			否	排序
NEWS_VOUCH	int			否	推荐
NEWS_CONTENT	longtext			是	内容
NEWS_VIEW_CNT	int			否	浏览次数
NEWS_PIC	varchar	255		是	图片
NEWS_OBJ	longtext			是	OBJ
NEWS_FORMS	longtext			是	FORM
ADD_TIME	bigint			否	添加时间
EDIT_TIME	bigint			否	编辑时间

表 4-4 收藏表

字段名	类型	长度	主键/外键	允许为 NULL	说明
FAV_ID	int		主键	否	收藏 ID
FAV_USER_ID	int			否	用户 ID
FAV_TITLE	varchar	255		否	收藏标题
FAV_TYPE	varchar	100		否	收藏类型
FAV_OID	int			否	文章 ID
FAV_PATH	varchar	255		否	路径
ADD_TIME	bigint			否	添加时间
EDIT_TIME	bigint			否	编辑时间

表 4-5 项目表

字段名	类型	长度	主键/外键	允许为 NULL	说明
MEET_ID	int		主键	否	活动 ID
MEET_TITLE	varchar	255		否	活动标题
MEET_CATE_ID	int			否	分类 ID
MEET_CATE_NAME	varchar	255		是	分类名称
MEET_STATUS	int			否	状态
MEET_ORDER	int			否	排序
MEET_VOUCH	int			否	推荐
MEET_DAYS	longtext			否	预约天数
MEET_VIEW_CNT	int			否	浏览次数
MEET_MAX_CNT	int			否	最大人数
MEET_FORMS	longtext			否	FORM
MEET_OBJ	longtext			否	OBJ
ADD_TIME	bigint			否	添加时间
EDIT_TIME	bigint			否	编辑时间

表 4-6 预约表

字段名	类型	长度	主键/外键	允许为 NULL	说明
MEET_JOIN_ID	int		主键	否	预约 ID
MEET_JOIN_USER_ID	int			否	用户 ID
MEET_JOIN_MEET_ID	int			否	项目 ID
MEET_JOIN_CODE	varchar	20		否	核销码
MEET_JOIN_IS_CHECK	int			否	是否核销
MEET_JOIN_CHECK_TIME	bigint			否	核销时间
MEET_JOIN_FORMS	longtext			否	FORM
MEET_JOIN_OBJ	longtext			否	OBJ
MEET_JOIN_STATUS	int			否	状态
MEET_JOIN_TIME	varchar	255		否	预约时间
MEET_JOIN_DAY	varchar	255		否	预约日期
MEET_JOIN_MEET_TITLE	varchar	255		否	项目标题
ADD_TIME	bigint			否	添加时间
EDIT_TIME	bigint			否	编辑时间

4.4 本章小结

本章主要对系统各个关键技术，前后端的设计进行介绍。其中重点设计了系统的注册与单点登录模块、预约模块。之后介绍了数据库的设计。

5 系统实现

上一章已完成对预约系统的关键模块的设计，接下来对核心模块的设计部分进行编码。实现的效果图以微信小程序页面直观表达。通过该方式，用户可以更好的了解整体架构和核心功能。此外，本章将引入部分代码进行讲解，以便用户能理解模块的实现方法。为保证系统质量，本章将对功能进行测试，包括可用性及安全性。将保证在实际使用时表现出良好的性能。

5.1 系统环境

为了保证预约系统的可靠开发，该系统采用了相对稳定的开发环境。具体如下表：

表 5-1 系统开发环境

操作系统	macOS Monterey 12.7.6
后端开发工具	IntelliJ IDEA 2025.1
前端开发工具	微信开发者工具 (Stable Build)
运行环境	Amazon Corretto 1.8.0452
数据库	MySQL 8.4.3

由于本系统后续需要上线至微信小程序，本地运行不可满足其异地、随时访问之要求，因此采用了如下的云端运行环境：

表 5-2 系统运行环境

后端运行环境	Microsoft Azure B1s VM
数据库服务器	Microsoft Azure SQL Database

5.2 系统关键模块的实现

系统总分为两大模块。分别为 framework 框架模块以及 project 项目模块。

Framework 包是一个结构化的基础框架模块，提供了注解用于标记特定行为，AOP 切面实现日志记录等功能，并通过配置类集成 Spring 和 MyBatis-Plus 等组件。此外，还包含异常处理机制、各类实用工具，如 JWTHelper 处理令牌、FileHelper 管理文件、TimeHelper 处理时间）、登录拦截器以及数据校验工具。

Project 模块基于 framework 模块，采用经典的分层架构设计。

在控制层方面，模块设计了面向管理员和普通用户的两套接口体系。管理员接口位于 admin 子包，普通用户接口集中在 cust 子包，所有客户端的控制器都继承自统一的 BaseMyCustController 基类。其中包含了处理认证授权的 PassportController、管理预

约的 MeetController、处理收藏功能的 FavController 等多个业务控制器，每个控制器都专注于特定的业务领域。

数据持久层采用了 MyBatis 作为 ORM 框架，采用增强的 MyBatis-Plus，通过 MeetMapper、UserMapper 等接口与数据库进行交互。这些接口与 MeetModel、UserModel 等实体模型一一对应，构成了完整的数据访问体系，涵盖了用户管理、服务预约、新闻资讯等核心业务数据操作。

业务服务层同样按照管理端和用户端进行划分，分别位于 admin 和 cust 子包中。这一层负责处理核心业务逻辑，协调多个数据访问操作，并实现必要的业务规则校验，是整个系统的业务逻辑核心。

在项目配置方面，ProjectConfig 类提供了项目特有的配置项，而 SmartWorkApplication 作为 SpringBoot 的启动类，负责整个应用的初始化工作。这种配置方式既保持了与基础框架的兼容性，又允许项目进行必要的自定义配置。

整个模块通过规范的命名约定（XxxController、XxxMapper、XxxModel）和清晰的分层设计，实现了系统的核心功能。它与 framework 框架形成了良好的协作关系，framework 框架提供通用能力支撑，业务模块专注于具体业务实现，共同构成了完整的应用系统。以下阐述几个关键功能的实现：

5.2.1 单点登录模块

单点登录模块要完成用户登录和 Token 校验两大主要功能，另外提供登出功能主动使 Token 失效。

登录功能的实现包括前端提供一个登录相关的表单，后端提供登录接口；在登录过程中通过用户 account 查找用户信息，若找到，则对其密码进行 MD5 计算，与数据库中的密码 MD5 校验。登录通过后计算一个全局唯一的 token 值作为会话凭证返回给小程序前端，由前端进行保存，并在之后的请求中携带 token。

单点登录模块为其他模块提供用户身份校验服务，前端暂存的 token 将在使用功能时提交给相应的模块进行校验，能获取到 token 即可使用服务，否则不提供服务，并抛出异常。

以下文件展示 token 校验逻辑：

```
    public Map<String, Object> login(String account, String password) {
    //    此处省略部分代码
    // 判断密码正误
            if (ObjectUtil.isNull(user))
                throw new AppException("用户名或者密码错误");
    // 判断账户状态
            if (user.getUserStatus() == UserModel.STATUS.FORBID) throw new AppException("用户已禁用");
    //  生成 TOKEN
            String token = JWTHelper.genToken(user.getUserId(), user.getUserName(), "cust", AppConfig.JWT_CUST_EXPIRE);
            Map<String, Object> ret = new HashMap<>();
            ret.put("account", user.getUserAccount());
            ret.put("id", user.getUserId());
            ret.put("name", user.getUserName());
            ret.put("token", token);
            ret.put("expire", AppConfig.JWT_CUST_EXPIRE);
            if (user.getUserLoginTime() == 0)
                ret.put("last", "尚未登录");
            else
                ret.put("last", TimeHelper.timestamp2Time(user.getUserLoginTime()));
            user.setUserLoginTime(user.getUserLoginTime() + 1);
            user.setUserLoginTime(TimeHelper.timestamp());
            userMapper.update(user, where);
            return ret;
        }
```

以下为生成 token 的核心代码：

```
    public static String genToken(long id, String name, String sub, Long expire) {
            JWT jwt = JWT.create();
            // 设置携带数据
            jwt.setPayload("id", id);
            jwt.setPayload("name", name);
            jwt.setSubject(sub);
            // 设置密钥
            jwt.setKey(AppConfig.JWT_SECERT.getBytes());
            // 设置过期时间
            jwt.setExpiresAt(new Date(System.currentTimeMillis() + expire * 1000));
            return jwt.sign();
        }
```

图 5-1 访问系统拦截提示

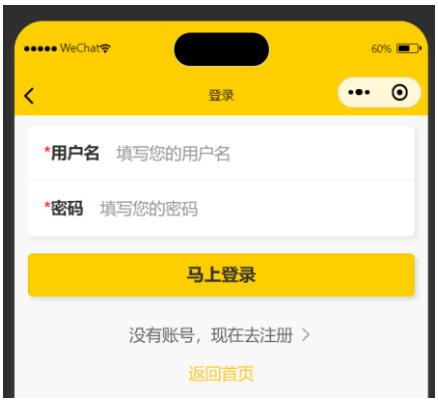

图 5-2 用户登录界面

5.2.2 服务预约模块

宠物预约活动管理类包含控制类、服务组合类、数据持久化类和实体类，用于协调和管理宠物预约活动系统中的各个功能模块。活动管理控制类主要包括 AdminMeetController 和 MeetController 两个核心控制器。AdminMeetController 类专门接收系统管理员对宠物预约活动的管理请求并作出响应，MeetController 类则负责处理普通用户参与预约活动的相关请求，通过与数据持久层和实体类的交互完成预约活动的全流程管理。

在服务层设计上，系统采用 AdminMeetService 和 MeetService 双服务模式。AdminMeetService 类专注于处理管理员侧的活动管理业务逻辑，包括活动创建、修改和状态维护等核心功能；MeetService 类则主要实现用户参与预约活动的业务逻辑，如预约提交、取消和查询等功能。这两个服务类共同构成了宠物预约活动的完整业务处理体系。

数据持久化层由 MeetMapper 和 MeetJoinMapper 两个主要组件组成。MeetMapper 负责提供宠物预约活动基础信息的数据持久化能力，包括活动信息的增删改查等操作；MeetJoinMapper 则专注于处理用户预约记录的数据持久化，确保预约数据的准确存储和高效访问。这两个数据访问组件为上层业务逻辑提供了可靠的数据支撑。

系统实体类设计包含 MeetModel 和 MeetJoinModel 两个核心实体。MeetModel 作为宠物预约活动的主实体类，定义了活动 ID、活动标题、活动分类、最大参与人数等关键属性；MeetJoinModel 则是用户预约记录实体类，包含预约 ID、关联活动 ID、用户 ID、预约时间、签到状态等字段。这两个实体类完整描述了宠物预约活动领域的业务对象，为系统各层提供了统一的数据模型。

这种分层架构设计使得宠物预约活动管理系统能够清晰划分职责边界，AdminMeetController 和 MeetController 分别处理不同角色的请求，AdminMeetService 和 MeetService 实现差异化的业务逻辑，MeetMapper 和 MeetJoinMapper 专注各自的数据领域，而 MeetModel 和 MeetJoinModel 则确保了数据模型的一致性。各组件协同工作，共同实现了宠物预约活动从创建发布到用户参与的全生命周期管理。

下图为管理员添加预约项目：

图 5-3 管理员添加项目

添加项目后，会跳转回上一页，展示各项服务：

图 5-4 管理端项目展示

其中，编辑按钮，可对该项目进行修改操作，即跳转回添加项目页面，此时所有表单均有填写，管理员仅需修改对应项目即可。名单与核销按钮，可查阅所有已预约的用

户，管理员可在此进行后台核销或点击相应按钮进行扫码核销。状态管理，管理员可针对该项目进行启禁操作，更多操作中，管理员可针对该项目是否上推荐页置顶进行操作。

设置项目后，系统默认可预约日期为 0 天，管理员需点击上方“去设置更多日期”文字进行日期编辑，其页面使用日历的方式直观表示，便于操作，如下图所示：

图 5-5 服务日期设置

此处以日历显示的核心实现代码如下：

```
<view wx:for="{{dayArr}}" wx:if="{{mode=='multi'}}" wx:key="key" data-fullday ="{{item.full}}" class="cube {{glow}} {{isLunar?'lunar':''}} {{item.full<fullToday? 'timeout':''}} {{tools.includes(multiDoDay,item.full) ? 'calendar-bg text-white data-checked' : ''}} {{tools.includes(hasDays,item.full) ?'data-has':''}} {{ tools.includes (hasJoinDays, item.full) ?'join-has':''}}" bindtap="bindDayMultiTap">
    <view class="num-grid {{fullToday==item.full? 'now-day-cur' : ''}} ">
        <view lass="num {{!item.curMonth? 'text-no-month' : ''}} ">
            <text class="dd">{{item.show}}</text>
            <text wx:if="{{isLunar}}" class="lunar {{tools.includes(multiDoDay,item.full)? 'text-white' : ''}} {{item.holiday?'text-red':''}}" >{{item.lunar}}</text>
        </view>
    </view>
</view>
```

选定日期后，管理端即完成一次预约发布。

用户端在登录系统后，点击主页对应模块即可查阅到预约项目：

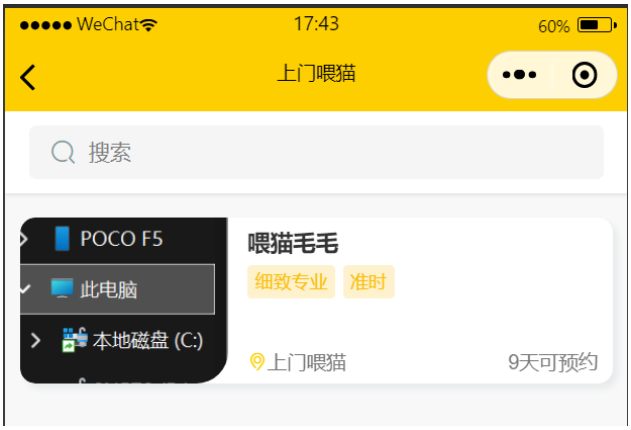

图 5-6 用户查阅服务项目

点击项目后。会跳转至预约时间页面，用户需选择合法的日期进行选择：

图 5-7 预约日期及时间

选择后跳转至确认订单页面，用户需确认手机号以及姓名是否输入正确：

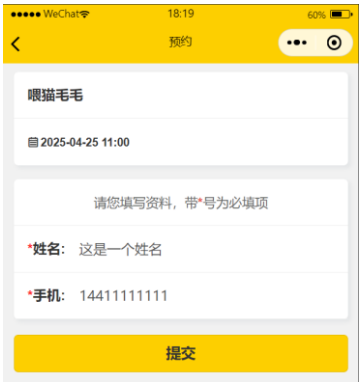

图 5-8 确认页面

点击提交后即预约成功。用户返回至主页，点击我的-我的所有预约可查看预约记录：

图 5-9 预约记录

此时宠物师未上门服务，因此显示未签到。用户待宠物师上门后，可点击查看详情，出示 QR Code：

图 5-10 预约详情

宠物师上门后，在管理端点击扫码核销即可，该操作会调用系统相机服务，因此会向系统申请摄像机权限：

图 5-11 管理员核销

核销后，即为完成一次服务。下方为扫码核销核心代码：

```
public void scanMeetJoin(long meetId, String code) {
        Where<MeetJoinModel> where = new Where<>();
        where.eq("MEET_JOIN_MEET_ID", meetId);
        where.eq("MEET_JOIN_CODE", code);
        MeetJoinModel meetJoin = meetJoinMapper.getOne(where);
        if (ObjectUtil.isEmpty(meetJoin)) throw new AppException("没有该用户的预约记录，核销失
败");
        if (meetJoin.getMeetJoinStatus() != MeetJoinModel.STATUS.NORMAL)
            throw new AppException("该用户未预约成功，核销失败");
        if (meetJoin.getMeetJoinIsCheck() == 1) throw new AppException("该用户已签到/核销，无须
重复核销");
    UpdateWhere<MeetJoinModel> uw = new UpdateWhere<>();
    uw.eq("MEET_JOIN_MEET_ID", meetId);
    uw.eq("MEET_JOIN_CODE", code);
    uw.set("MEET_JOIN_IS_CHECK", 1);
    uw.set("MEET_JOIN_CHECK_TIME", TimeHelper.timestamp());
    meetJoinMapper.edit(uw);
}
```

5.2.3 数据导出模块

系统首先会接收管理员指定的活动 ID 和时间范围参数，然后创建一个结构化的 Excel 文件，包含序号、姓名、手机号、预约日期、具体时段、填写时间和签到状态等 7 个核心字段。在数据处理过程中，采用分页查询机制，每次从数据库获取 100 条记录，避免内存溢出问题。

在技术实现上，利用 Hutool 工具包的 Excel 处理能力，通过 ExcelWriter 实现数据的流式写入。对于每条预约记录，系统会从 JSON 格式的 meetJoinObj 字段中提取用户的姓名和手机号等关键信息，并将数据库中的数字状态标识转换为更易读的"已签到/未

签到"文字描述。以下为写入到 Excel 文件的核心代码：

```
for (int i = 1; i <= pageCount; i++) {
    Page page = new Page(i, size, false);
    PageResult ret = userMapper.getPageList(page, where);
    List<Map<String, Object>> retList = ret.getList();
    for (Map<String, Object> node : ret.getList()) {
        no++;
        ArrayList<String> arr = new ArrayList<>();
        arr.add(Convert.toStr(no));
        arr.add(MapUtil.getStr(node, "userAccount"));
        arr.add(MapUtil.getStr(node, "userName"));
        int status = MapUtil.getInt(node, "userStatus");
        arr.add(status == 1 ? "正常" : "禁用");
        arr.add(MapUtil.getStr(node, "addTime"));
        JSONArray forms = JSONUtil.parseArray(MapUtil.getStr(node, "userForms"));
        for (int j = 0; j < forms.size(); j++) {
            JSONObject jsonObject = forms.getJSONObject(j);
            arr.add(Convert.toStr(jsonObject.get("val")));
        }
        writer.writeRow(arr);
    }
```

以下为导出数据界面展示：

图 5-12 数据导出页面

	A	B	C	D	E	F	G
	序号	姓名	手机	日期	时段	填写时间	是否签到
	1	这是一个姓名	14411111111	2025-04-25	11:00	2025-04-25 18:21:47	未签到
	2	渡边胜	14411111111	2025-04-08	1	2025-04-08 10:17:29	已签到

图 5-13 导出结果

5.2.4 数据合法性校验模块

在上文中，每次调用对应功能时，均需要进行数据合法性校验，否则不予通过。该模块主要用于对输入参数进行类型和格式的严格校验，实现高度可定制的校验逻辑，并且防止 SQL 注入。此处展示三类最基本的校验代码：

```
public class DataCheck {
    static public void checkMust(Object value, String desc) {
        if (ObjectUtil.isEmpty(value))
            throw new AppException(desc + "不能为空", AppCode.ERROR_DATA);
    }
    static public void checkBool(Object value, String desc) {
        if (value.getClass() != Boolean.class)
            throw new AppException(desc + "类型错误(bool)", AppCode.ERROR_DATA);
    }
    static public void checkEmail(Object value, String desc) {
        if (ObjectUtil.isNull(value))
            return;
        String val = Convert.toStr(value);
        if (StrUtil.isEmpty(val))
            return;
        if (!Validator.isEmail(val))
            throw new AppException(desc + "必须为邮箱格式", AppCode.ERROR_DATA);
    }
}
```

该部分实现了前端传来空值、布尔类型错误的常见错误，以及校验电子邮箱输入是否合法。并抛出相应异常以阻止程序继续运行。对于邮箱检验时用到的 isEmail()方法，源自 Hutool 工具集，此处直接调用，减少开发工作量。本模块是统一的校验逻辑，减少在各个具体业务中重复编写校验代码，即避免相同的校验逻辑散落在各处，易于维护。

5.3 系统测试

在预约系统开发过程中，难免遇到各种问题及漏洞，从而影响系统的安全和稳定性。因此，需要进行各种测试以保证系统的正确运行。系统测试是开发过程中重要部分，伴随整个开发过程[12]。本节主要测试系统的功能模块，如登录模块、用户预约模块、数据导出模块，简要进行性能测试。

5.3.1 系统功能测试

本文主要针对预约系统功能模块的部分主要功能进行测试，使用的测试用例包括属性、目的、角色、场景、输入、预期结果、实际结果。每个模块的具体测试如下：

(1) 登录功能测试

用户登录时，需输入已注册的账户名以及对应的密码登录。登录模块的测试用例如

下表所示：

表 5-3 登录模块功能测试

测试属性		功能测试	
测试目的		检验系统登录功能是否正常	
角色		普通用户	
场景	输入数据	预期效果	实际结果
用户点击登录按钮	已注册账号、对应密码	登录成功、跳转页面	登录成功、跳转页面
用户点击登录按钮	已注册账号、错误密码	登录失败、抛出异常	登录失败、抛出异常
用户点击登录按钮	未注册账号及密码	登录失败、抛出异常	登录失败、抛出异常

测试用户登录功能时，根据测试用例操作，注册用户使用正确的账号和密码登录，对于未注册的用户，系统拒绝登录，并在登录时提示用户适当的操作。

(2) 预约功能测试

用户在预约项目时，会选择适应的时间进行预约，以及填写合法的、中国大陆的手机号码，以便管理员进行联系。在预约功能的测试中，需要测试预约时间是否合法，以及手机号是否填写正确。因此，所需的测试用例如下表所示：

表 5-4 预约功能测试

测试属性		功能测试	
测试目的		检验预约功能是否正常	
角色		普通用户	
场景	输入数据	预期效果	实际结果
用户选择时间日期	当前日期在预约范围内	日期选择成功、跳转	日期选择成功、跳转
用户选择时间日期	当前日期出界	无法显示预约项目	无法显示预约项目
用户点击预约按钮	合法中国大陆手机号	预约成功	预约成功
用户点击预约按钮	非法中国大陆手机号	预约失败、报错	预约失败、报错

用户可以正常预约，且各项功能均正常，未出现时间出界、手机号异常注入情况。

管理员收到预约请求后，将对其进行核销，核销的一个输入数据是用户端生成的 QR Code，因此该部分的测试，需要验证扫描错误的 QR Code 的报错以及正确扫描后，是否显示已核销。以下是测试用例：

表 5-5 预约核销功能测试

测试属性		功能测试	
测试目的		检验核销功能是否正常	
角色		管理员	
场景	输入数据	预期效果	实际结果
管理员扫描	用户端二维码	核销成功	核销成功
管理员扫描	非用户端二维码	错误信息，拒绝核销	错误信息，拒绝核销

根据该测试用例，对导出模块进行了测试。通过以上测试结果分析，核销功能正常，未出现扫描其他二维码而导致核销成功的错误。

(3) 数据导出模块测试

数据导出模块，需要管理员输入日期范围，且要求开始时间不得大于结束时间。通过后即可导出文件。以下是测试用例：

表 5-6 数据导出功能测试

测试属性		功能测试	
测试目的		检验导出功能是否正常	
角色		管理员	
场景	输入数据	预期效果	实际结果
管理员点击导出按钮	开始时间 < 结束时间	正确导出 Xlsx 文件	正确导出 Xlsx 文件
管理员点击导出按钮	开始时间 > 结束时间	错误信息，拒绝导出	错误信息，拒绝导出

根据该测试用例，对导出模块进行了测试。通过以上测试结果分析，可以在正确的时间范围内导出所需要的数据。数据导出模块功能正常，基本实现。

5.3.2 系统性能测试

主要测试了系统的运行时间、相应速度、处理速度和系统压力情况。根据此要求进行系统性能测试的测试用例如下表所示：

表 5-7 性能测试

测试属性		性能测试	
测试目的		测试系统运行性能	
角色		系统开发人员	
场景	输入数据	预期效果	实际结果
长期运行系统	无	运行正常	运行正常
用户首次进入系统	无	2 秒响应	5 秒响应

系统可以长时间正常工作，但在用户首次进入系统时，出现了响应时间过长的问题。由于系统模块多，以登录为例进行以下压力测试，结果如下：

表 5-8 压力测试

测试项目			占用情况	
用例	响应速度(TPS)	响应时间(MS)	CPU	DB
5	28	46	14%	3%
10	30	50	16%	5%
50	55	91	22%	26%
100	104	173	26%	44%

TPS 表示每秒发送请求数量，如果每秒大于 40，用户将有明显感知。超过 100 时，

系统将出现一定压力。总体上，系统响应时间较为稳定，压力测试基本合格。

5.3.3 系统测试分析

由上文的测试可知，本系统功能较为完善，未出现致命漏洞，且前端采用微信小程序开发，运行在微信客户端，可有效防止使用地址栏进行 SQL 注入，因此安全性较高。但在性能测试时，用户首次进入系统出现了响应时间较长的问题，分析得知，后端服务器使用了微软 Azure 云服务，其服务器位于日本东京，延迟较高，因此导致响应时间较长。压力测试环节，由于数据库服务器与后端服务器分离，导致在对数据库进行增删改查操作时，也会因远程访问而导致延迟过高，导致卡顿出现。

5.4 本章小结

本章先对预约系统的开发以及运行环境进行了阐述，然后详细介绍了系统关键模块的是实现过程。详细阐述了单点登录模块、服务预约模块、数据导出模块。对系统功能测试以及性能测试进行了介绍与详细说明。在功能测试环节，针对用户登录、用户预约及核销等基本功能进行了测试，得出本系统可以满足上述功能的结论，系统功能完善。在性能测试环节，发现本系统在常规性能测试环节基本合格，但在压力测试环节出现了响应时间过长的问题，有待进一步优化。综上所述，本系统在实现了核心功能的基础上其性能也基本达到预期，满足普通用户低负载运行。

结论

本论文围绕宠物预约系统的设计与实现展开研究，针对现代都市养宠人群在工作繁忙、出差或旅行等场景下的宠物照看需求，开发了一套智能化的同城上门喂宠预约系统。研究首先分析了当前宠物家政服务行业存在的服务不足等痛点，指出传统电话预约模式在效率和服务质量上的局限性，提出了数字化解决方案的必要性。

在技术架构方面，系统采用 Spring Boot+ MyBatis-Plus 的后端技术栈，结合微信小程序前端，构建了前后端分离的现代化 Web 应用。关键技术实现包括：基于 RESTful 规范的 API 设计、JWT 令牌的身份认证机制、MyBatis-Plus 的动态条件查询构建器、以及 Hutool 工具包的 Excel 导出功能。系统创新性地设计了双重核销机制（扫码核销+后台核销）和智能分页导出方案，有效解决了服务过程监管和大数据导出的技术难题。

开发过程中遵循软件工程规范，实现了包含用户端、服务端和管理端的完整系统。用户端提供预约全流程服务，管理端配备数据看板和导出功能。通过严格的测试验证，系统在功能完整性达到预期目标，性能指标和用户体验方面基本达标，为宠物服务行业的数字化转型提供了可落地的技术方案。

本系统在开发过程中仍存在若干需要改进的技术环节。最显著的是尚未实现与微信生态的深度整合，用户目前只能通过传统用户名注册登录，支付环节也暂未接入微信支付功能，这在一定程度上降低了用户使用的便捷性。服务调度也仍然通过管理员进行人工按照地址调度，降低了服务效率。

针对现有不足，后续将重点完善微信生态的整合。计划通过微信开放平台接口，实现用户一键微信登录，消除注册环节的繁琐步骤；同时接入微信支付体系，构建完整的线上支付闭环。在服务调度方面，拟引入基于机器学习的智能匹配算法，综合考虑服务人员位置因素，实现更精准的服务资源分配。

参考文献

[1] 北京派读科技有限公司. 2025 年中国宠物行业白皮书[R]. 派读宠物行业大数据平台,2025.

[2] Sachidanand V. Begur;;David M. Miller;;Jerry R. Weaver.An Integrated Spatial DSS for Scheduling and Routing Home-Health-Care Nurses[J].Interfaces,1997(4).

[3] 沈昊杰.家政服务的互联网应用优化研究[D].南昌大学,2017.

[4] 刘艺溥.基于人脸识别的家政服务平台设计与实现[D].北京交通大学,2020.

[5] 刘一澎.基于微信小程序的专家预约系统的设计与实现[D].南京大学,2021.

[6] Biørn-Hansen, A., Majchrzak, T.A., Grønli, TM. Progressive Web Apps for the Unified Development of Mobile Applications[J]. Web Information Systems and Technologies, 2017.

[7] Zhang, FangCAa; Sun, GuilingCAb; Zheng, BowenCAc; Dong, LiangCAd. Design and implementation of energy management system based on spring boot framework[J]. Information (Switzerland),2021.

[8] 黑马程序员. Spring Boot 企业级开发教程[M]. 人民邮电出版社, 2024.

[9] 腾讯公司. 微信开放文档[EB/OL]. https://developers.weixin.qq.com, 2025.

[10] 王珊, 杜小勇. 数据库系统概论(第 6 版)[M]. 高等教育出版社, 2023.

[11] 张海藩,牟永敏.软件工程导论(第 6 版)[M]. 清华大学出版社,2013.

[12] （美）罗恩·佩腾（Ron Patton）. 软件测试[M]. 机械工业出版社, 2019.

致　谢

光阴荏苒，四年的大学生活很快就要过去了，回顾往昔，有太多的不舍，这四年的时光让我成长了很多，要感谢的人也太多。

首先，感谢我的指导老师闫琛老师。本论文是在闫老师悉心指导下完成。从论文开题到最终完成，闫老师一直给予我不同的意见和建议，解答了我在撰写过程中遇到的各种难题。

感谢实训基地北京百知教育科技有限公司的魏振楠工程师，在我遇到项目上的困难时，愿意细心解答，帮助我快速的了解业务，为依据项目撰写论文打下了良好的基础。

感谢太原理工大学的四年本科生培养，以及软件学院的各位老师。正是因为老师们的悉心教导让我一步步蜕变为更好的自己。

感谢本科就读期间结实的每一位挚友，你们的包容和帮助让我倍感珍惜，在未来的日子里，愿我们各自奔赴山海，不忘初心，砥砺前行。

感谢含辛茹苦养育我的父母，多年漂泊在外，对他们的照顾与关怀甚是不足，心中愧疚万分。愿我的家人身体健康。

最后感谢从未轻言放弃的自己。