东 莞 理 工 学 院

本 科 毕 业 设 计

**毕业设计题目：xxxx设计与实现**

**学生姓名：叶诗健**

**学 号：202041417240**

**学 院：计算机科学与技术学院（软件学院、网络空间安全学院）**

**专业班级：2021级大数据2班**

**指导教师姓名及职称：秦勇教授**

**起止时间： 2024 年 12 月—— 2025 年 5 月**

**摘 要**

本系统是一款面向租赁场景的小程序平台，旨在为租客与管理员提供高效、便捷的租赁管理服务。系统采用前后端分离架构，后端基于 Go 语言与 Gin 框架实现，前端使用微信小程序构建。系统核心功能包括微信与用户名密码两种方式的用户认证与注册、房源浏览与筛选、在线订单管理、电子签约与签名上传、账单自动生成、远程门锁密码管理以及工单提交与处理等。通过模块化设计与清晰的系统结构，本平台实现了租赁流程的全流程数字化管理，显著提升了用户体验与管理效率。系统采用 PostgreSQL 数据库并配合 JWT 鉴权机制，确保数据一致性与访问安全，具备良好的可用性、可扩展性与可维护性。

**关键字 租赁系统、微信小程序、互联网应用**

**Abstract**

This system is a WeChat Mini Program platform designed for rental scenarios, aiming to provide tenants and administrators with efficient and convenient rental management services. It adopts a front-end and back-end separated architecture, with the back-end developed using Go and the Gin framework, and the front-end built as a WeChat Mini Program. The core features include user authentication and registration (via WeChat and username/password), property browsing and filtering, online order management, electronic contract signing and signature uploading, automatic bill generation, smart lock password management, and work order submission and tracking. Through modular design and a clear system structure, the platform enables a fully digitalized rental workflow, greatly improving user experience and administrative efficiency. The system uses a PostgreSQL database and JWT-based authentication to ensure data consistency and access security, with strong usability, scalability, and maintainability.

**Keywords** rental system, WeChat Mini Program, Web Application

目 录

[第1章 绪论 1](#_Toc68681458)

[1.1项目背景与意义 1](#_Toc68681459)

[1.2论文的章节安排 1](#_Toc68681460)

[第2章 相关技术和开发环境 2](#_Toc68681461)

[2.1 相关技术 2](#_Toc68681462)

[2.2 开发环境 2](#_Toc68681463)

[第3章 系统分析 3](#_Toc68681464)

[3.1 系统需求 3](#_Toc68681466)

[3.2 可行性研究 3](#_Toc68681466)

[第4章 概要设计 4](#_Toc68681467)

[4.1 系统设计 4](#_Toc68681468)

[第5章 详细设计 5](#_Toc68681469)

[第6章 系统实现 6](#_Toc68681470)

[第7章 系统测试 7](#_Toc68681471)

[7.1 测试环境 7](#_Toc68681472)

[7.2 测试策略和方法 7](#_Toc68681473)

[7.3 测试内容和步骤 7](#_Toc68681474)

[7.3测试结果及分析 7](#_Toc68681475)

[第8章 总结与展望 8](#_Toc68681476)

[参考文献 9](#_Toc68681477)

[致谢 10](#_Toc68681478)

# 第1章 引言

**1.1项目背景与意义**

随着我国城市化进程的不断加快和人口流动的日益频繁，房屋租赁市场持续扩大，租房需求呈现多样化和个性化的发展趋势。然而，传统的房屋租赁方式仍存在大量信息不对称、流程繁琐、沟通效率低下、合同管理混乱等问题，不仅影响租客的租房体验，也给房东和平台管理方带来较大的管理压力和运营成本。因此，构建一个高效、智能的信息化房屋租赁管理系统，已成为提升租赁服务质量和规范市场秩序的迫切需求。

本课题旨在设计并实现一套基于现代软件工程技术的房屋租赁管理系统，全面覆盖房源管理、租赁合同签订与变更、租金支付、退租申请与审核等核心功能模块。系统将采用结构清晰、可维护性强的单体架构，并在设计中充分考虑未来功能扩展与系统分布式部署的可能性，为平台的长期运营提供技术保障。

与此同时，系统在数据库设计方面注重数据存储的规范性与查询效率，通过优化数据结构与索引机制，有效支持高并发环境下的读写操作。在支付模块实现上，结合实际业务场景，设计完善的账单生成、支付、退款机制，确保资金流转的安全性与合规性。

本项目的实施不仅可以提升房屋租赁过程的数字化管理水平，增强用户对平台的信任感与满意度，还具有显著的现实应用价值和社会经济效益。从技术角度来看，该系统为高并发场景下的服务设计、在线支付集成与复杂业务流程建模提供了实践平台；从行业角度而言，它有助于推动租赁平台服务向标准化、智能化方向发展，促进房屋租赁行业的数字化转型。因此，本课题的研究与实现具有重要的现实意义和实践价值。

**1.2论文的章节安排**

本文的章节安排如下：

引言：本章主要介绍本项目的研究背景与现实意义，阐述房屋租赁市场中存在的主要问题以及信息化管理系统在该领域的重要作用。同时，概述本系统的研究目标、主要功能及技术路线，并简要介绍论文的整体结构。

相关技术与开发环境：本章介绍系统实现过程中所采用的关键技术，包括 Golang 语言、Gin Web 框架、GORM 数据库操作库、微信小程序开发技术、PostgreSQL 数据库等。同时，说明开发所使用的操作系统、工具链与依赖环境，为后续系统设计与实现章节提供技术基础。

系统设计与实现：本章详细阐述本系统的需求分析、系统结构设计、数据库设计及各核心功能模块的具体实现过程。包括用户认证、房源浏览、电子签约、账单生成、远程门锁控制、工单处理等模块。通过模块化设计思路实现系统的高内聚、低耦合，确保系统的稳定性与可维护性。

系统测试与评估：本章主要对系统的各项功能进行测试，涵盖功能测试、接口测试、性能测试与异常处理测试等方面，评估系统在真实运行环境中的稳定性、响应速度与用户体验。同时结合测试结果进行分析，提出优化建议。

总结与展望：本章总结系统的开发过程与实现成果，回顾项目的关键技术点与创新之处。同时指出当前系统的不足与待优化方向，展望未来在功能拓展、用户体验优化、智能化服务等方面的可能发展路径。

# 

# 第2章 相关技术和开发环境

**2.1 相关技术**

本系统采用 Golang 语言作为后端开发语言，结合 Gin 框架构建 RESTful 接口，微信小程序作为主要用户端实现，形成完整的租赁业务系统架构。系统涵盖了房源管理、合同管理、租金账单生成与退租处理等功能。在技术选型方面，系统强调高并发处理能力、开发效率与良好的用户交互体验，具体涉及以下关键技术：

（1）Golang 编程语言

Go 语言是一种静态类型、并发友好的编程语言，具有优秀的执行效率和简洁的语法。在本系统中，Golang 不仅用于构建后端服务接口，还承担了账单自动生成等定时任务的实现。通过 goroutine 和 channel 等原生并发机制，系统能够高效地处理大量并发请求与异步任务。

（2）Gin Web 框架

Gin 是一个高性能的 Web 框架，提供了路由管理、中间件支持和高并发处理能力，适用于构建 RESTful 接口。在本系统中，Gin 框架用于开发面向微信小程序的后端接口，涵盖房源信息查询、合同操作、账单查看等功能，确保接口响应速度快、结构清晰，便于前后端协作开发。

（3）Gorm 框架

早期的后端开发往往需要手动编写 SQL 语句进行数据查询与更新操作，代码中大量的 SQL 语句导致开发繁琐并且不好维护的难题，同时对于数据迁移往往依靠人工维护。GORM 框架能根据后端代码自动在数据库创建相应的实体模型，并能使用高层的编程语句而不是 SQL 语句进行数据库操作，同时具备数据库框架（Schema）自动迁移的功能。上述特性大大地简化了后端开发流程，提升了后端开发

1. 中间件（Middleware）

中间件是一类处于操作系统与应用程序之间的软件组件，常用于实现通用功能的抽象与复用，例如日志记录、请求认证、错误处理、跨域配置等。在本系统中，结合 Gin 框架的中间件机制，开发了用户身份验证、权限控制、请求日志记录等功能模块。这种方式既提高了代码的可维护性，又增强了系统整体的安全性与可扩展性。

（5）协程与定时任务机制

系统使用 goroutine 搭建了后台账单生成协程，在服务启动时自动运行，定期检查合同数据并生成账单。该机制无需额外引入定时任务框架，依赖 Golang 原生的并发能力实现轻量而稳定的后台任务调度，提高了系统的资源利用率和可靠性。

（5）微信小程序技术

微信小程序作为用户端平台，具备使用门槛低、部署快捷、用户基数大等优势。在本系统中，用户通过微信授权登录后访问租赁服务，完成房源浏览、合同签署与账单查看等功能。系统通过获取微信 openid 进行用户唯一标识，从而实现安全、高效的用户认证机制。

（6）PostgreSQL 数据库与接口设计

系统选用 PostgreSQL 作为核心数据库，具备强大的事务支持、丰富的数据类型和扩展性，适合复杂业务的数据建模与查询优化。数据库中设计了用户、房源、合同与账单等多个业务表，结合外键约束与索引优化，确保数据一致性与查询性能。接口设计遵循 RESTful 风格，通过结构化的 JSON 接口与清晰的 URI 路径，支持前端小程序的高效调用与拓展。

**2.2 开发环境**

|  |  |
| --- | --- |
| 操作系统 | Windows11 |
| 开发工具 | Visual Studio Code |
| Golang 版本 | 1.23.4 |
| 数据库 | Postgres 16.2 |
| Python | 3.12 |

**第3章 系统设计与实现**

**3.1 系统需求分析**

本系统主要面向租客与管理员两个角色，涉及如下主要功能：

（1）用户认证与注册：系统支持微信授权登录与用户名密码方式登录注册，满足不同用户使用习惯的同时，提高平台接入便捷性。

（2）房源浏览与筛选：用户可浏览平台上发布的房源信息，并通过位置、价格、户型等条件进行筛选，提升找房效率。

（3）订单管理：租客可以在线创建订单，发起退租流程，并支持订单状态的查询和管理。

（4）电子签约：用户可通过微信小程序在线签署电子合同，并上传电子签名，合同数据与订单信息关联存储。

（5）账单生成与管理：系统具备后台定时生成账单的功能，租客可在前端查看个人账单记录，支持历史账单回溯。

（6）远程门锁控制：租客可在小程序中获取或修改房门密码，提升智能化入住体验。

（7）工单管理：系统支持租客提交并查看维修工单或退租工单，并允许用户实时查看工单处理状态。同时管理员可以使用在管理界面进行工单的管理。

**3.2 可行性研究**

本系统是一款面向房屋租赁场景的小程序应用，主要提供房源浏览、订单管理、电子签署、远程门锁管理以及账单生成等功能。在系统开发前期，对技术实现、经济成本、用户操作性和后期维护等方面进行了初步分析，以确保项目具有较高的可行性。

（1）技术可行性

系统后端采用 Go 语言配合 Gin 框架开发，具备良好的并发处理能力和开发效率。同时，数据库部分选用 PostgreSQL，能够满足复杂查询和数据一致性方面的需求。前端则使用微信小程序技术实现，具备良好的用户覆盖和交互体验。项目已实现功能包括微信授权登录与账号密码登录、房源信息筛选与查看、订单创建与退租、电子签名上传、远程门锁密码更新、工单管理、账单生成等。整体技术路线清晰，所用框架和工具成熟稳定，具备实际部署能力，因此从技术层面是可行的。

（2）经济可行性

项目整体采用开源技术栈，避免了授权费用等额外支出。在初期部署阶段，可以选择配置较低的云服务器完成部署与测试，资源开销较小。微信小程序平台本身的开放性也降低了开发与上线成本。由于系统定位于中小规模的租赁管理，短期内对资源的需求量有限，具备良好的成本控制能力。

（3）操作可行性

系统界面基于微信小程序构建，用户无需下载额外应用即可访问，整体操作流程简洁明了。在房源筛选、订单管理、签署合同、查看账单及提交报修等核心流程中，均结合实际业务需求进行了功能设计，操作逻辑贴近用户日常使用习惯，具有较好的易用性和可接受性。

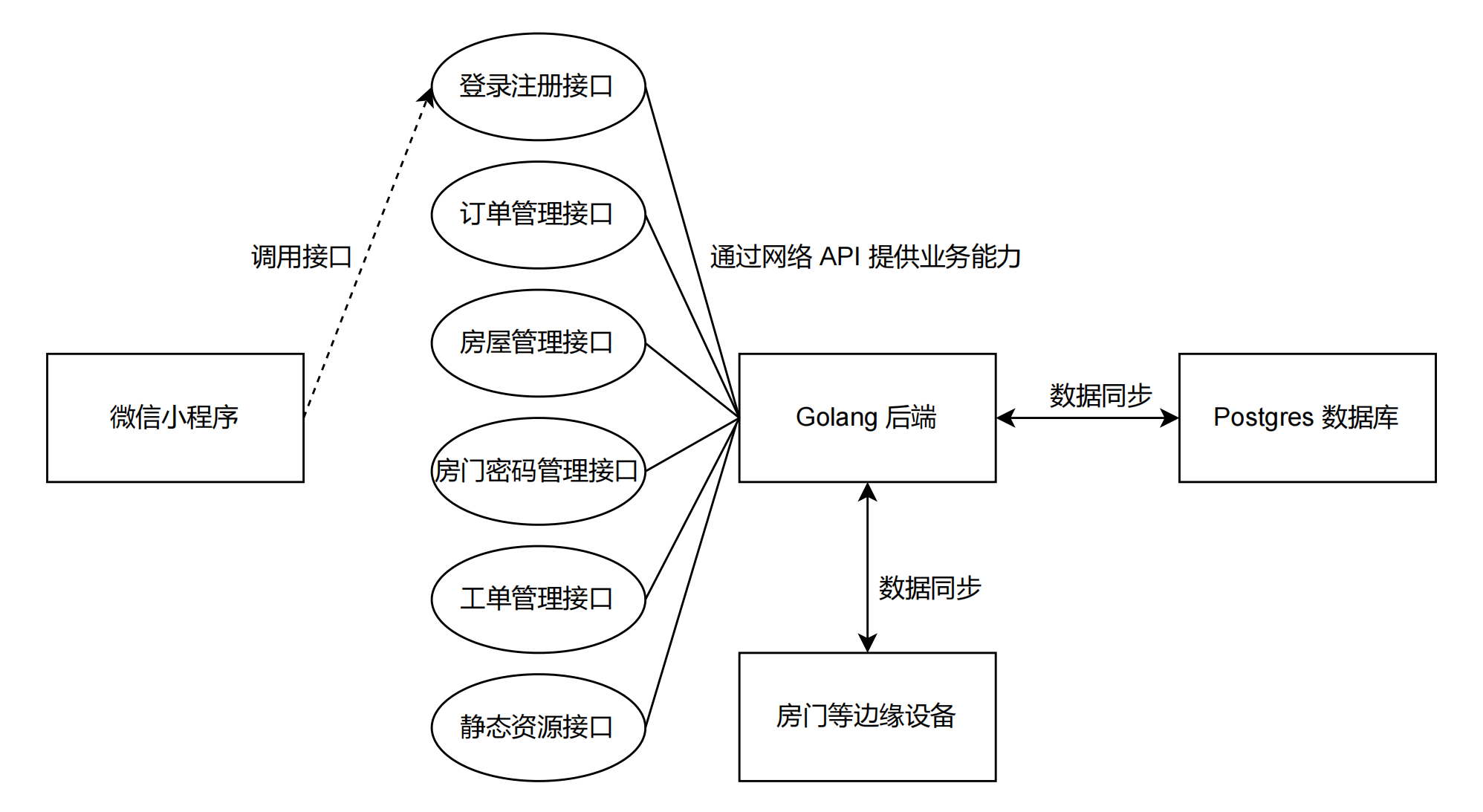
（4）维护可行性

系统采用模块化设计，各功能模块间耦合度较低，便于后期功能扩展和问题定位。后端代码结构清晰，注释规范，数据层使用 PostgreSQL 也便于后续备份、迁移和维护操作。此外，微信小程序的版本管理机制和调试工具也有助于实现快速迭代和功能优化，具备一定的可维护性。

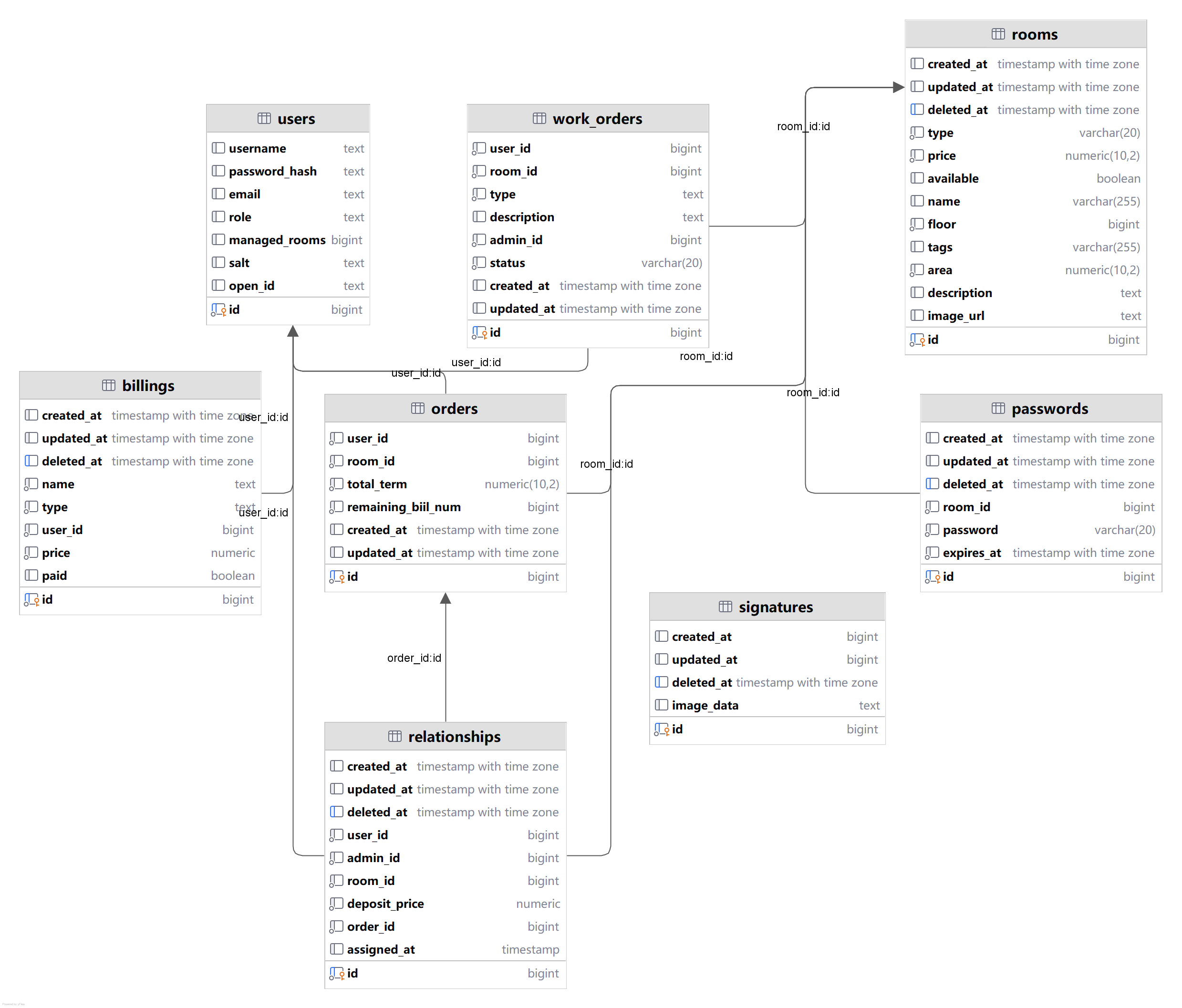
综上所述，本系统在当前开发条件下具有良好的可行性。已实现的功能覆盖了租赁流程中的主要环节，技术方案成熟稳定，具备较强的实际部署能力，同时也预留了后期扩展空间，例如引入支付系统、后台管理界面或对接智能门锁等，具备继续拓展和优化的潜力。为了确保本系统功能齐全、性能稳定且具备良好的用户体验，系统需求从功能性维度进行了详细分析，确保各项模块的实现具有可操作性。

**3.3 系统整体设计**

为了满足房屋租赁场景下的多方需求，本系统采用前后端分离架构，后端基于 Go 语言及 Gin 框架开发，数据库采用 PostgreSQL，前端则使用微信小程序实现。系统主要围绕租客与管理员两个角色进行功能划分，已实现的模块包括：用户登录注册（支持微信与账号密码两种方式）、房源浏览与筛选、订单创建与退租、电子签约上传、账单自动生成与查看、远程门锁密码管理、工单提交与处理等。各模块按照“用户层 - 业务逻辑层 - 数据层”的分层思想独立设计，既保障了系统功能的完整性，又提升了后期维护与扩展的便利性。

其中系统整体设计框架如下图所示：

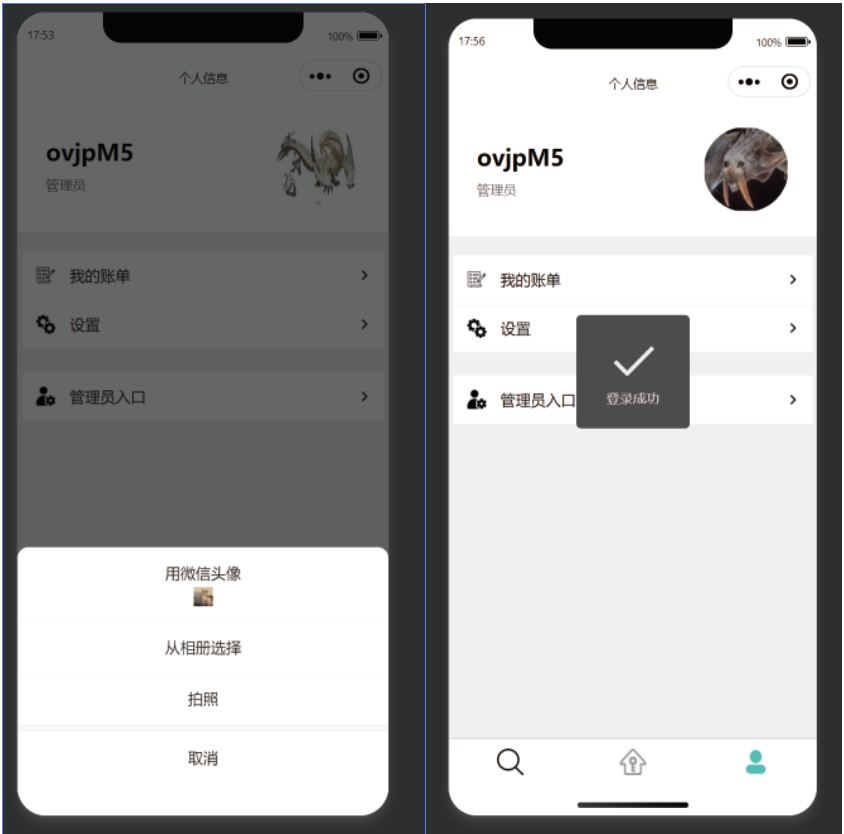
其中数据库实体关系模型（Entity-relationship model）如下图所示



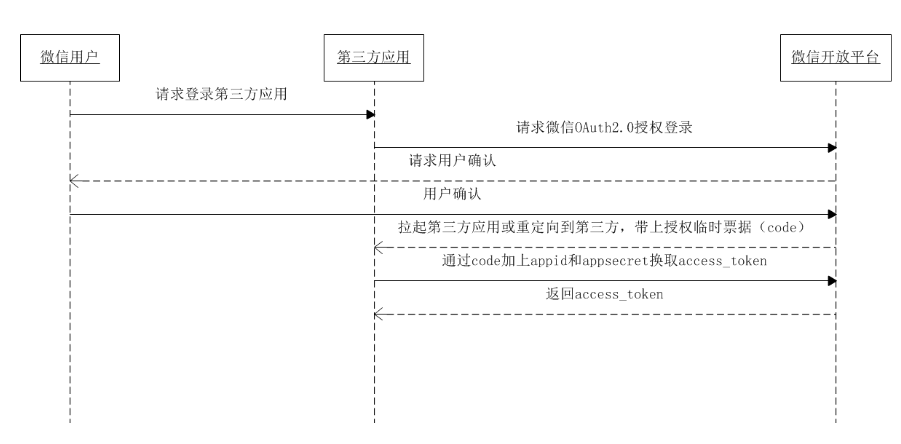
**3.4 各功能模块实现**

**3.4.1 注册与登录模块**

注册与用户登录模块界面如下图所示：



用户可以通过用户名密码或通过微信身份进行登录。其中微信身份登录需要对接微信官方开放接口，在官网注册应用程序获取应用程序 ID 与应用程序密钥之后，微信第三方登录流程如下图所示

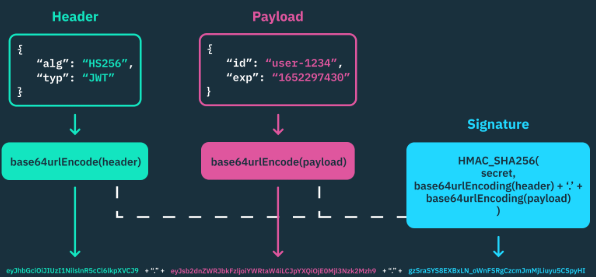


对于微信登陆与用户名密码的注册，本系统有着不同的处理机制：

1. 对于使用微信登录进行注册的用户，后台会自动在数据库中创建用户记录，并记录 OpenID（用户在微信服务数据库中的唯一标识）。之后用户若通过微信登录，将用 OpenID 进行查找与标识。
2. 对于使用用户名密码进行注册的用户，在系统确认密码格式合法之后。系统将生成随机的盐 (Salt) 值与密码明文进行拼接，最后使用 SHA-256 算法将拼接的字符串进行哈希计算，最后将盐值与哈希结果一同保存在数据库中。设 P 为用户输入的密码明文，S为系统生成的随机盐值,H(x) 代表对字符串 x 使用 SHA-256 哈希算法计算的结果，“||”代表对字符串进行拼接，数据库中的密码密文 Text为：

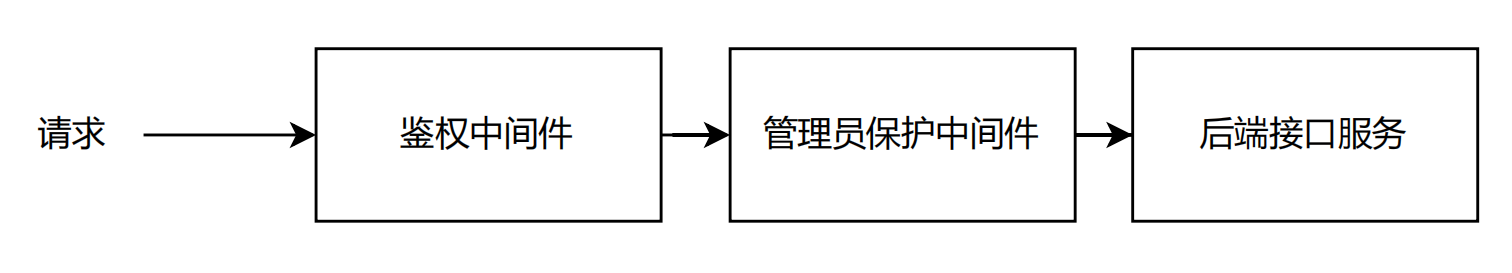
用户登录时，从数据库取出相关用户的盐值，对 HTTP请求携带的密码明文执行相同的操作，将生成的结果与数据库中的密码密文进行对比，若相等则未验证成功，用户成功登录。

用户登陆后，为了能获得对服务的访问权限，需要事先请求访问令牌（Access Token）。在现代互联网应用的鉴权中，访问令牌使用的是 JWT （JSON Web Token）格式，JWT由其头部，载荷与签名组成，其格式如下图所示：



常见的签名实现通常使用 HMAC 进行对称加密并哈希。在本系统的设计中，令牌的签名使用了非对称加密，进一步提升了访问令牌的安全性。

系统生成访问令牌的同时，会把登陆用户的身份信息放入令牌的头部部分，在系统验证签名有效之后，可以从令牌的头部确定用户的身份是普通用户还是管理员。同时使用中间件技术，将验证逻辑以中间件的形式安排在各个接口之前。这样设计避免了在每一个接口中编写验证逻辑，减少了后期维护成本。

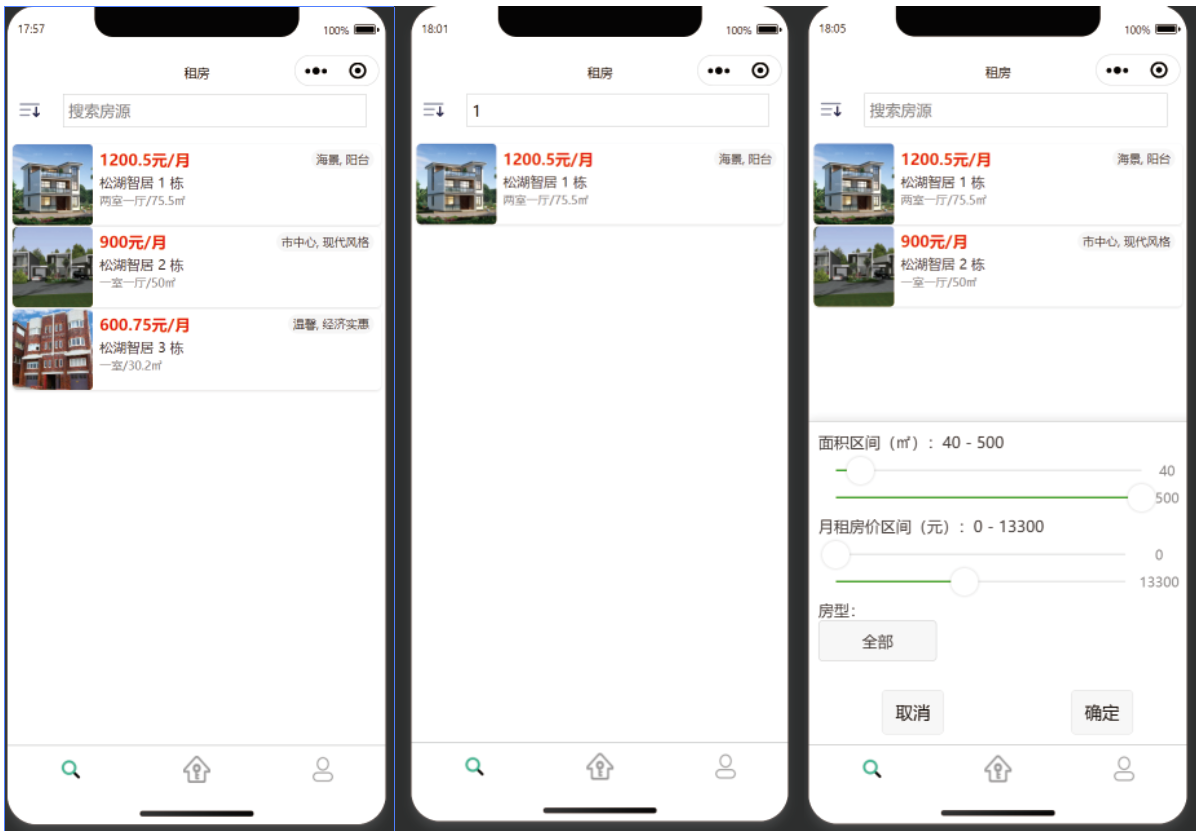
用户请求到达鉴权中间件的时候，中间件会从请求携带的 JWT 中提取用户的身份，之后经过管理员保护中间件的检查，若接口需要管理员权限而 JWT 中携带着普通用户的身份，请求将被后端拒绝。本系统使用中间件与接口的关系如下示意图所示：

生成的 JWT 的签名由SHA-256对头部与载荷拼接形成的字符串进行哈希算法后使用RSA算法加密生成。设 H(x) 代表对字符串 x 使用 SHA-256 哈希算法计算的结果， 表示使用 RSA 私钥对数据 x进行签名操作，base64(x) 代表对数据 x 进行 base64 URL 安全编码。JWT 签名部分 Signature可表示为：

在后端初始化的过程中将生成公私钥对，进行令牌验证的时候，使用私钥对签名进行解密，同时对头部与载荷的 base64 编码字符串进行 SHA-256算法，对比结果是否相等。若相等则令牌验证成功。

**3.4.2 房源浏览与筛选模块**

房源浏览与筛选界面如下图所示：



房源浏览与筛选模块是系统中面向用户的核心功能之一，承担着展示房源信息、提供多维度筛选及搜索能力的任务。微信小程序前端通过列表形式展示所有房源信息，每个房源卡片包含标题、价格、面积、地址、户型、封面图等基本内容，用户点击任意房源可跳转至详情页面查看完整信息。

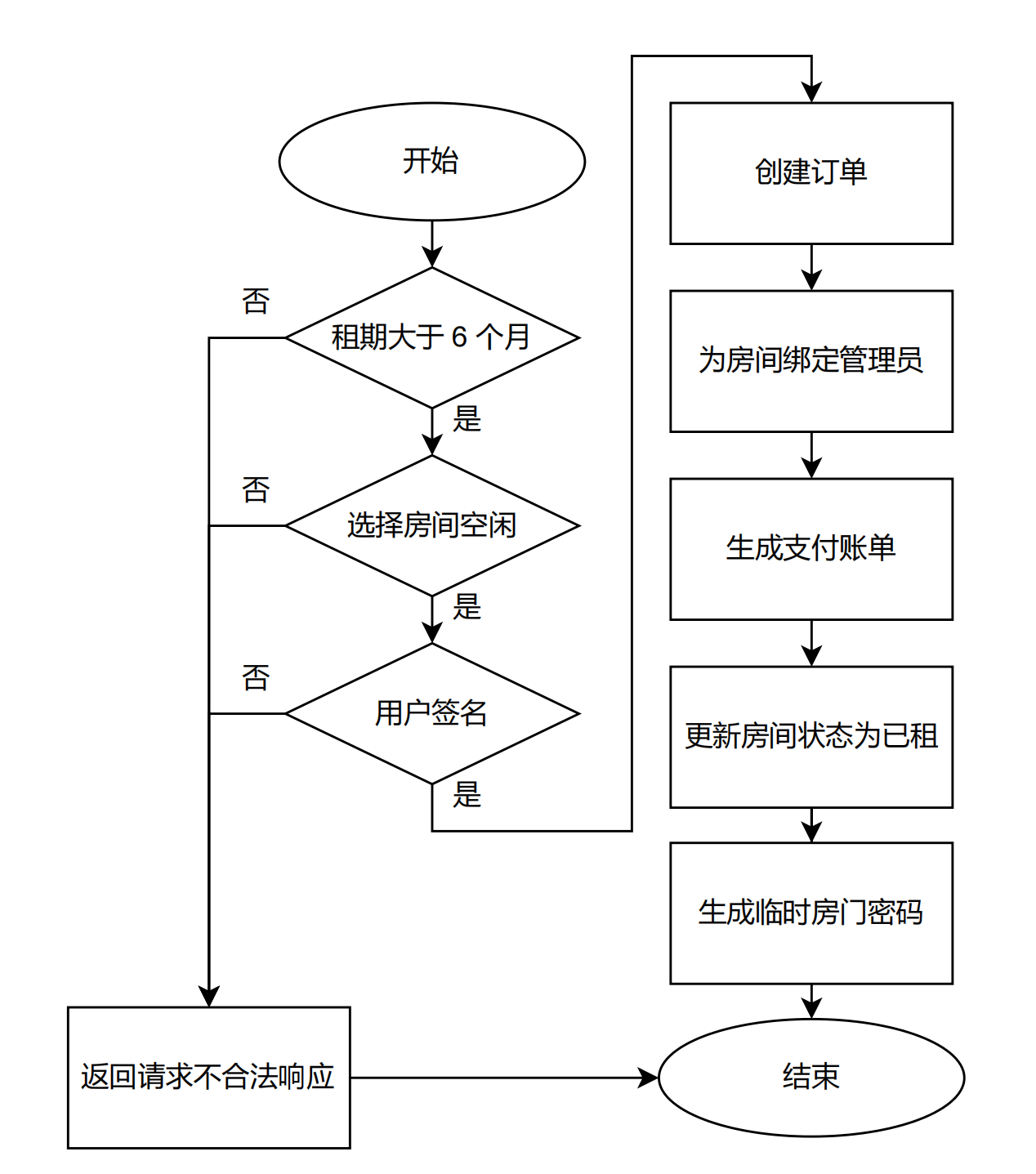
系统支持用户根据多个条件进行筛选，包括价格区间、面积范围、户型结构等，同时也提供关键词搜索功能，用户可以输入房屋名称关键词，后端会根据标题和描述字段使用 “LIKE”SQL语句进行模糊匹配筛选。

**3.4.3 电子签约模块**

签约界面如下图所示：

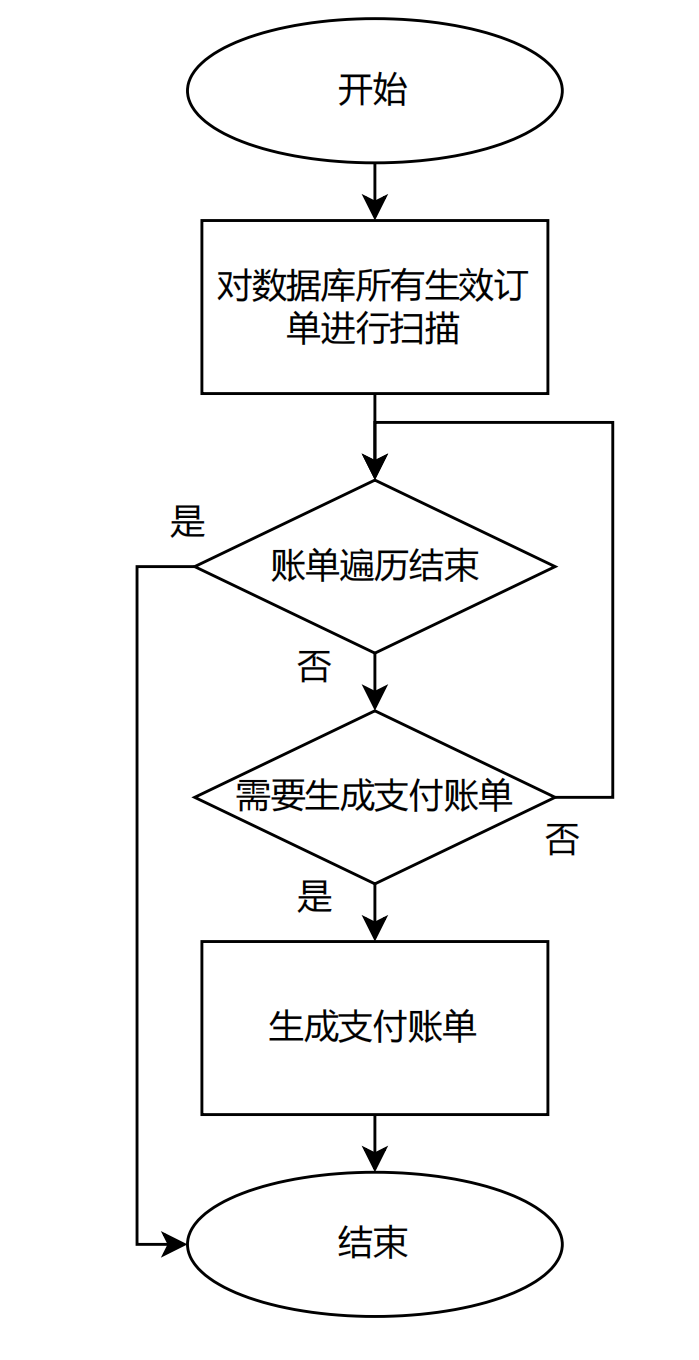
本系统设计中，租期至少为 6 个月。在用户确认房型并进行签约时，系统会将用户信息，日期，房租金额，押金金额自动填入电子合同。用户在阅读完毕点击确认后小程序界面会跳转到签名界面。等待用户签字确认后，生成的签名图片被上传至后端，使用服务器文件系统对电子签名进行管理。同时为了日后能正常索引到签名图片，图片在服务器本地的路径将会被保存至数据库。

当用户成功完成合同签约后，系统后台将自动生成订单数据，并将房间的管理分配给当前管理房间数量最少的管理员，之后房间的工单将自动分发给该房间管理员，并且创建押金支付账单。支付账单金额数为房间月租的两倍，房屋签约流程图展示如下：



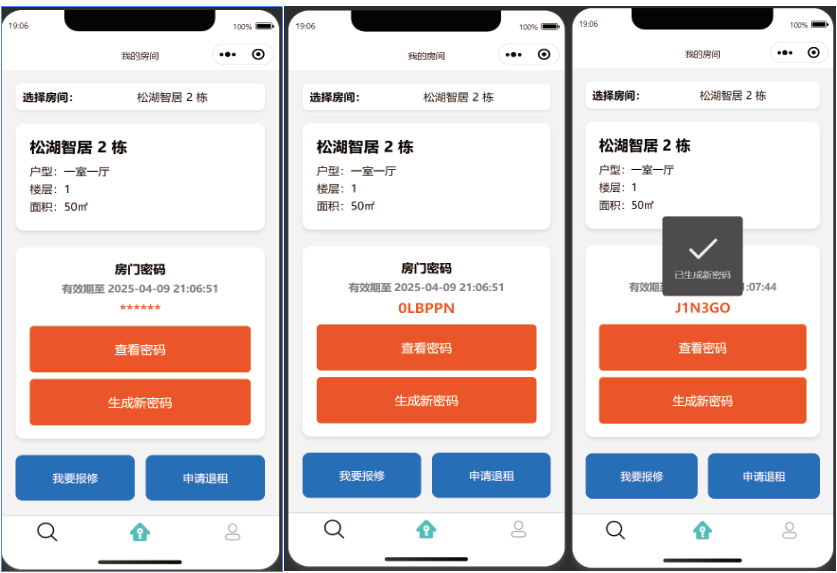
**3.4.4 账单生成模块**

由于租房涉及每月房租的收取，本系统在后台使用一个 Golang 协程进行定时任务。每经过 5 分钟对数据库中正在生效的订单进行扫描，并根据当前时间与订单生成的时间以及已经支付的期数，计算出该订单所需支付的期数。若已经支付的期数等于所需支付期数，则继续进行下一个数据行的检查。否则为该订单生成用户关联的付费账单并保存至数据库。账单生成流程如下图所示：



**3.4.5 门锁密码管理模块**

门锁管理界面如下图所示：



本系统在用户完成房屋签约之后，会自动重新生成随机房间密码，同时与边缘设备同步以下放门锁密码。

同时用户可以选择在小程序界面重新生成密码并进行查看。出于隐私安全的考虑，小程序从后端获取门锁密码之后，将只会保存并显示密码 3 秒。之后会清除本地密码数据并在界面上显示 “\*\*\*\*\*\*”以代表保密的密码。

**3.4.6 工单处理模块**

工单管理模块分为用户端与管理员端，用户报修提交工单后，管理员可在管理界面进入，并对工单进行状态更新。

用户提交工单页面如下图所示



管理员界面如下图所示：

本系统工单管理模块主要用于用户日常生活中的日常维护服务以及用户退租流程的管理。用户可在在小程序界面上描述问题并发起工单，而管理员可进入管理页面进行工单的处理（根据情况选择已完成）。

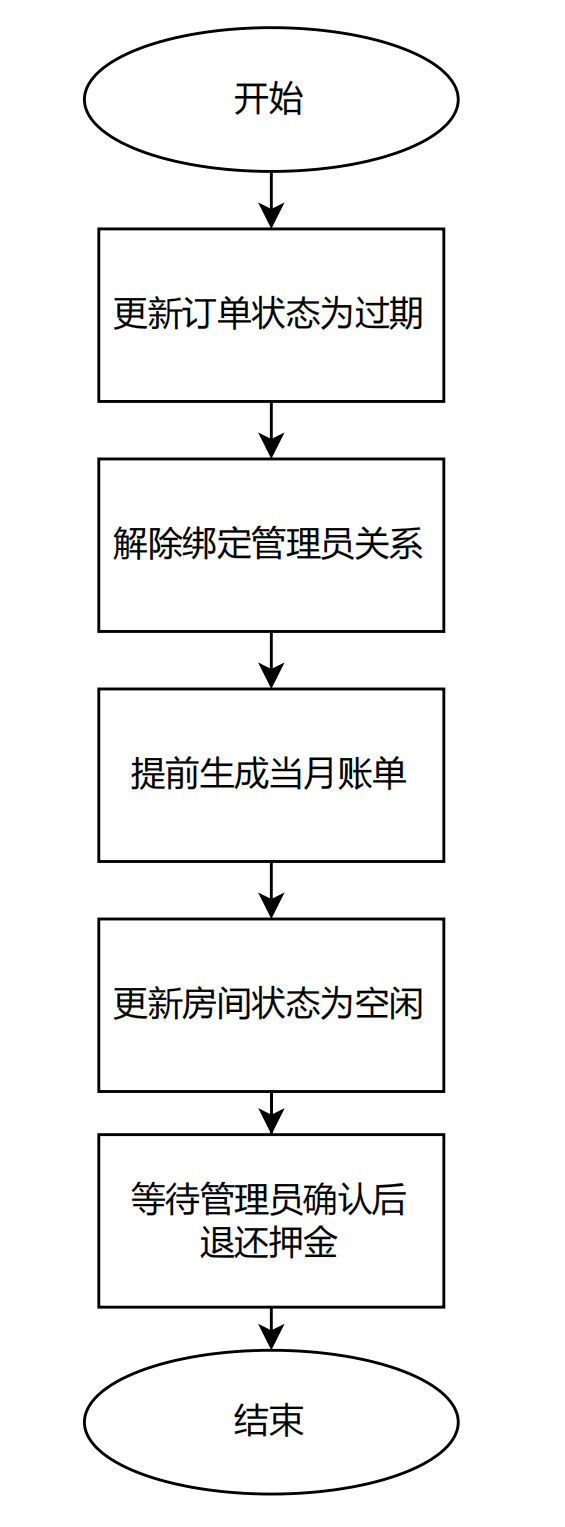
对于退租流程工单，管理员需在检查房屋家具齐全，房屋内无破坏痕迹之后，对退租工单进行确认。用户的退租流程此时才正式启动。退租流程发起后，系统会提前计算本月账单。设整月房租为 R，当前日期为该月的第 d 天，该月的总天数为 D，则应支付的房租金额 A 可通过以下公式计算得出：

该方法根据租客所在的具体日期，将整月房租按天均分，并乘以当日所在天数，从而实现对实际入住天数的精确计费，确保计费的公平性与合理性。

由于系统存在后台协程以每五分钟工作一次的间隔进行账单的生成，我们应该使用互斥锁技术（Mutual Exclusion）保证后台协程与提前计算订单的代码的并发安全。

互斥锁是一种常见的并发控制技术，用于防止多个线程或进程同时访问共享资源（如内存、文件、变量等）导致数据混乱的问题。它的核心思想是：同一时间只允许一个线程持有锁并访问资源，其他线程必须等待当前线程释放锁之后才能继续执行。使用互斥锁的设计保证了提前生成最后一期账单的同时，后台账单生成协程无法对数据库的账单进行更新操作。

退租流程示意图如下所示：



**第4章 系统测试与评估**

## 4.1 测试环境

|  |  |
| --- | --- |
| 操作系统 | Windows11 |
| CPU | Intel Core i7-13400 |
| 接口测试工具 | Postman |
| 数据库 | Postgres 16.2 |
| Python | 3.12 |
| 接口性能测试工具 | Locust |

## 4.2 测试内容和步骤

系统测试主要验证系统是否满足用户需求说明中的功能项，包括用户注册登录、房屋信息浏览、租赁流程、订单管理与后台管理等模块的正常运行。网页应用通常包括接口测试与性能测试，本测试将使用 Postman 逐个对接口进行请求进行功能测试。并使用 Locust 框架与 Python 脚本对接口进行逐步的压力测试以估计程序支持的最大用户在线数。

## 4.2.1 功能测试

功能测试过程中将对各个接口进行调用，并检查输入输出是否符合预期。本系统总共包括 24 个接口。每个接口所需身份权限不同，权限控制由 2.1 节所述中间件实现。接口详细信息如下表所示：

|  |  |  |
| --- | --- | --- |
| 接口名称 | 需要的身份权限 | 说明 |
| /login | 无 | 登录 |
| /register | 无 | 注册 |
| /list-rooms | 无 | 获取所有房间信息 |
| /get-rooms | 无 | 根据 ID 获取房间信息 |
| /list-filtered-rooms | 无 | 根据过滤条件获取房间信息 |
| /create-order | 普通用户 | 创建订单 |
| /pay-bill | 普通用户 | 支付账单 |
| /list-relationships | 普通用户 | 查看房屋，管理员与用户的关系 |
| /list-orders | 普通用户 | 查看历史订单 |
| /list-billings | 普通用户 | 查看全部账单 |
| /terminate-lease | 普通用户 | 退租 |
| /change-room-password | 普通用户 | 更新房间密码 |
| /get-password | 普通用户 | 获取房间密码 |
| /list-user-workorders | 普通用户 | 展示用户历史工单 |
| /create-work-order | 普通用户 | 创建工单 |
| /list-owned-rooms | 普通用户 | 查看用户签约房间 |
| /upload-signature | 普通用户 | 上传电子签名 |
| /register-admin | 管理员 | 创建管理员账号 |
| /update-room | 管理员 | 更新房间信息 |
| /list-admin-workorders | 管理员 | 查看管理员有关工单 |
| /update-workorder | 管理员 | 更新工单状态 |
| /static | 无 | 代理静态图片资源 |

## 4.2.1 性能测试

性能测试中使用 Python 脚本以及 Locust 框架，分别模拟500,1000,1500个用户数目下，同时对后端发起请求，重复执行三次，每次时长 5 分钟。由于业务服务器中 CPU 配置最少为 4 核8线程，本测试将限制后端程序只使用 8 个系统线程。其中进行用户登录测试的脚本如下所示：

from locust import HttpUser, task, between, SequentialTaskSet, task

import random

class UserBehavior(SequentialTaskSet):

# 设置模拟用户的行为流程

@task(1)

def login(self):

"""模拟用户登录"""

login\_data = {

"username": "test\_user",

"password": "password123"

}

response = self.client.post("/login", json=login\_data)

if response.status\_code == 200:

self.token = response.json().get("token") # 假设登录返回 token

print(f"登录成功，Token: {self.token}")

else:

print("登录失败")

@task(2)

def get\_rooms(self):

"""获取房间信息"""

headers = {"Authorization": f"Bearer {self.token}"}

response = self.client.get("/get-rooms", headers=headers)

if response.status\_code == 200:

print("成功获取房间信息")

else:

print("获取房间信息失败")

@task(3)

def create\_order(self):

"""模拟创建订单"""

order\_data = {

"room\_id": random.choice([1, 2, 3, 4, 5]), # 随机选择一个房间

"user\_id": "test\_user\_id",

"start\_date": "2025-01-01",

"end\_date": "2025-12-31"

}

headers = {"Authorization": f"Bearer {self.token}"}

response = self.client.post("/create-order", json=order\_data, headers=headers)

if response.status\_code == 201:

print("订单创建成功")

else:

print("订单创建失败")

@task(1)

def get\_user\_orders(self):

"""获取用户的所有订单"""

headers = {"Authorization": f"Bearer {self.token}"}

response = self.client.get("/list-orders", headers=headers)

if response.status\_code == 200:

print("成功获取用户订单信息")

else:

print("获取用户订单信息失败")

class WebsiteUser(HttpUser):

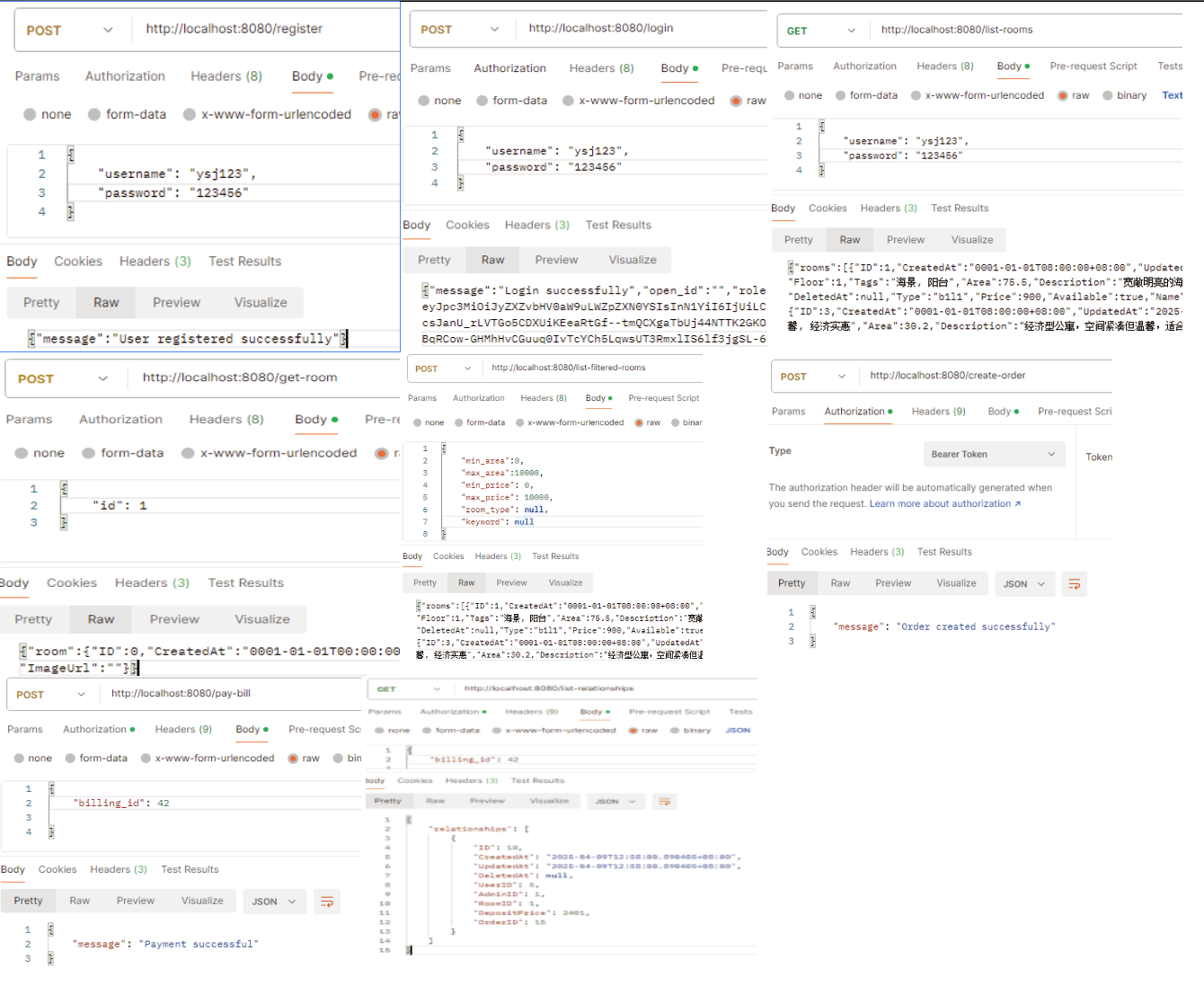
wait\_time = between(2, 5) # 设置请求间隔时间（2到5秒之间）

tasks = [UserBehavior] # 让 WebsiteUser 执行 UserBehavior 中的任务

## 4.3测试结果及分析

## 4.3.1 功能测试结果

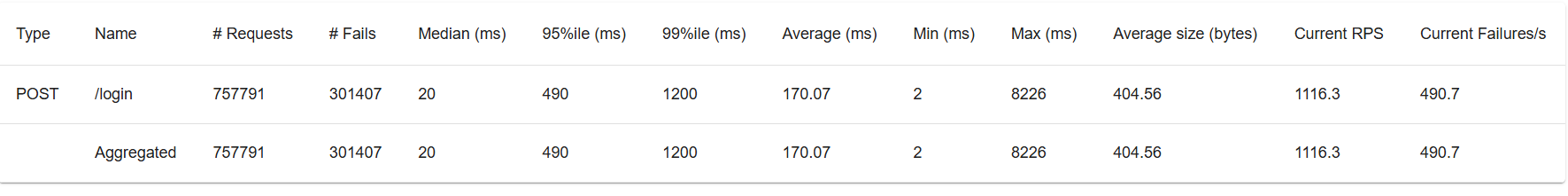
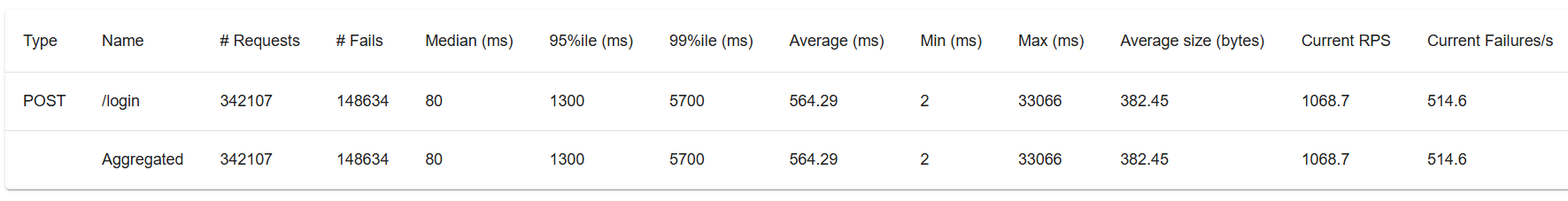
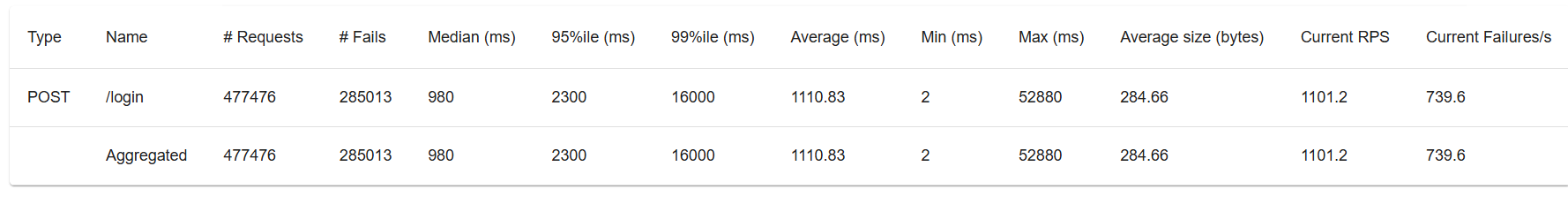
对 7.2.1 中的接口使用 Postman 测试工具进行单独测试，测试部分过程如下图所示：



所有接口都按照预期正常工作，根据输入产生了预期之内的输出。

## 4.3.2 性能测试结果

Locust 框架具有自动生成测试图表的能力，模拟不同用户数下结果如下所示:



整理之后可以得到如下表格：

|  |  |
| --- | --- |
| 模拟用户数（个） | 平均响应时间（ms） |
| 500 | 170.07 |
| 1000 | 564.29 |
| 1500 | 1110.83 |

当一个网页应用程序延迟超过 1000 毫秒时，用户将明显感觉到延迟，即用户体验受到影响。观察表格我们可以发现在 1500用户的并发下，平均用户的响应时间为 1110.83毫秒，这已经会使用户感受到明显的延迟。而在模拟用户数为 500 个的时候，平均响应时间为 170.07 毫秒，这属于良好的网络延迟。综上可以得知在 Intel Core-i7 13400 中 4 核 8 线程的配置下，本系统最大支持同时在线的用户为 1000 至 1500 个，属于良好的结果。

由于本系统未做缓存优化以及良好的索引结构，高负载的工作模式下数据库返回了慢查询 (Slow Query) 时间大于 200 毫秒的警告。这通常说明数据库查询效率可以优化，或者操作系统面临巨大负载。在本测试环节中，操作系统 CPU 占用量保持在 80% 以下，因此推断数据库索引设计可以进一步优化，或者可以通过添加 Redis 缓存层减少对数据库频繁的访问。

**第5章 总结与展望**

本论文围绕本租赁管理系统的设计与实现，详细介绍了系统的需求分析、技术选型、整体架构设计以及各主要功能模块的具体实现过程。系统基于微信小程序前端和 Go + Gin 后端架构，搭配 PostgreSQL 数据库，涵盖用户认证、房源筛选、订单处理、电子签约、账单生成、门锁管理与工单系统等多个核心业务流程。各模块之间通过分层架构设计保持较低耦合度，保障了系统的可维护性和可扩展性。

在实现过程中，系统充分考虑了用户体验、安全性与实际业务需求之间的平衡。例如，微信授权与用户名密码双通道登录机制，既提升了接入效率，也保障了数据安全；远程门锁密码控制与工单系统的集成，增强了租赁过程的智能化与服务响应能力。

尽管本系统已基本实现核心功能，但仍存在进一步完善与优化的空间。未来可从以下几个方面展开拓展与提升：

（1）引入在线支付系统：目前账单生成后仍依赖线下支付，后续可通过接入微信支付等方式实现线上缴费，进一步简化流程。

（2）完善后台管理平台：当前管理员功能仍主要通过数据库操作或简单管理端口完成，未来可开发更友好的后台界面，以提升管理效率与可视化程度。

（3）增强数据安全与备份机制：引入数据加密存储、访问日志审计以及数据库定期备份等功能，提升整体系统的可靠性和数据安全性。

（4）拓展智能设备接口：除门锁外，未来可对接更多智能设备（如智能水电表、摄像头等），进一步提升房屋管理的自动化程度。

在本系统未来的设计中，将更加严谨地检查高并发环境下程序内部的并发控制流的安全性，并通过 Redis 缓存数据库进行常用数据的缓存优化，以提升系统运行效率。系统将在未来与边缘设备集成，实现智能化的水电数据收集与管理，同时优化页面样式以以进一步提升用户体验舒适度并减轻管理人员维护压力与成本。

**参考文献**

**基本要求：（1）参考文献应该严格按照指定格式书写；（2）参考文献在正文中一定要有引用；（3）建议10-15条；（4）参考文献条目格式：固定行距18磅；宋体，五号，数字英文Times New Roman字体。**

**参考文献规范：（请严格执行）**

参考文献的著录均应符合国家有关标准（按GB7714—87《文后参考文献著录格式》执行）。参考文献的序号左顶格[1]，[2]，[3]。按照你的文献的类型（期刊论文、会议论文、专著书籍、标准等类别），论文引用格式也不一样。不同类型格式如下。

[序号] 期刊作者．题名[J]．刊名．出版年，卷(期):起止页码．

[序号] 专著作者．书名[M]．版次(第一版可略)．出版社，出版年:起止页码．

[序号] 会议作者．题名[C]．会议论文集名．出版社，会议地址，出版年:起止页码．

[序号] 学位论文作者．题名[D]．保存地点：保存单位，年份．

[序号] 专利所有者．专利文献题名[P]．国别：专利号．发布日期．

[序号] 标准编号，标准名称[S]．出版者，出版年．

[序号] 报纸作者．题名[N]．报纸名，出版日期(版次)．

[序号] 报告作者．题名[R]．报告地：报告会主办单位，年份．

[序号] 电子文献作者．题名〔电子文献及载体类型标识〕．文献出处，日期．

以上，部分出版社、地址等，若是在难以查实可忽略。但其他大部分应该可以查到。

**注意标点使用要统一：**

**要么全用中文句点和中文逗点（通常是全角）**

**要么全用英文句点和英文逗点（通常是半角，接着后面要加一个空格）**

**参考样例：**

[1] 李丹. 社交网站用户的行为和动机[J]. 传媒观察, 2009(4):44-45.

[2] 孙丽芳. QQ空间对当代大学生价值观的导向[J]. 北京社会科学, 2010(6):78-81.

[3] Adhi Susilo, 王申. 利用"Facebook"发展网络学习社区[J]. 天津电大学报, 2009, 13(1):29-34.

[4] 王玉英. 基于JSP的MySQL数据库访问技术[J]. 现代计算机, 2010. 19(14):63-66.

[5] 李刚. Struts权威指南[M]. 电子工业出版社,2007.9.

[6] 美)Elisabeth Freeman著林旺, 张晓坤译. Head First HTML与CSS、XHTML[M]. 北京:中国电力出版社, 2008.4.

[7] 单东林, 张晓菲, 魏然. 锋利的JQuery[M]. 北京: 清华大学出版社, 2007.

[8] 美)巴萨姆, 贝茨著, 苏钰函, 林剑译. Head First Servlets & JSP[M]. 中国电力出版社, 2006.

[9] 美)Y.Daniel Liang著万波, 郑海红, 潘蓉等译. Java语言程序设计（进阶篇）[M]. 北京:机械出版社.

**致谢**

在回顾本科四年的计算机科学与技术学习时，我深刻感受到了这段经历对我的成长与发展的重要性。四年的学习让我从最基础的编程语言、数据结构与算法，到逐渐深入的计算机网络、操作系统、人工智能等专业领域，都获得了扎实的理论知识和实践能力。同时，课外的项目实践、团队合作和学术研究，使我更好地理解了所学的知识，并学会了如何在实际问题中应用它们。

在此，我要特别感谢我的母校东莞理工学院，感谢她为我提供了一个良好的学习平台和学术环境。学校的教学资源丰富，学术氛围浓厚，培养了我严谨的学术态度和求知的热情。感谢我的各位老师，他们不仅传授了我专业知识，还教会了我如何独立思考，如何面对计算机技术学习过程中的挑战和困难。在他们的悉心指导下，我不仅在学术上取得了进步，也在做人做事的原则上获得了许多宝贵的启示。

尤其要感谢我的指导老师秦勇教授，感谢他在论文选题、研究方法、技术细节等方面给予的细心指导。教授的严谨治学和对细节的关注，给了我深刻的启发，使我能够在科研中做到精益求精。无论是在学术上的引导，还是在思想上的启发，都是我顺利完成本科阶段学习的重要动力。

总之，四年的学习让我不仅收获了知识和技能，也培养了我面对未知问题时的解决能力和创新意识。这些收获将成为我未来不断追求卓越、勇于探索的动力源泉。再次感谢学校、老师以及所有帮助过我的人，是你们的支持和鼓励让我有信心迈向更加广阔的未来。