

**毕业设计**

题 目 基于微信小程序的酒店管理系统的设计与实现

二级学院 数据科学与计算机学院

专 业 计算机科学与技术

学 号 201851701140

姓 名 李东福

指导教师 李贞辉 讲师

2021年4月9日

广东培正学院教务处制

毕业设计书原创性声明

本人郑重声明：所呈交的设计书是本人在指导教师的指导下独立进行研究所取得的研究成果。除了设计书中特别加以注明引用的内容外，本设计书不包含任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。



毕业设计书作者签名（手写）

2022年4月9日

毕业设计版权使用授权书

本毕业设计书作者完全了解学校有关保障、使用毕业设计书的规定，同意学校保留并向有关学位设计书管理部门或机构送交设计书的复印件和电子版，允许设计书被查阅和借阅。本人授权省级优秀学士学位设计书评选机构将本设计书全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位设计书。

本设计书属于：

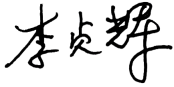
保密□，在 年解密后适用本授权书。

不保密☑。

（请在以上相应的方框内打“√”）



作者签名（手写） 2022年4月9日



指导教师签名（手写） 2022年4月9日

目录

[1 绪论 1](#_Toc100443394)

[1.1 课题背景及意义 1](#_Toc100443395)

[1.1.1 课题背景 1](#_Toc100443396)

[1.2 研究现状 1](#_Toc100443397)

[2 开发工具和开发平台 2](#_Toc100443398)

[2.1关键性知识及技术简介 2](#_Toc100443399)

[2.1.1 云开发技术 2](#_Toc100443400)

[2.1.2 Go语言概述 2](#_Toc100443401)

[2.2 数据库平台 3](#_Toc100443402)

[2.3 开发环境及工具 3](#_Toc100443403)

[2.4 系统部署 3](#_Toc100443404)

[3 总体分析 4](#_Toc100443405)

[3.1 可行性分析 4](#_Toc100443406)

[3.2 功能需求分析 5](#_Toc100443407)

[3.3 数据库需求分析 5](#_Toc100443408)

[3.4 性能需求分析 6](#_Toc100443409)

[4 总体设计 7](#_Toc100443410)

[4.1 实现功能简介 7](#_Toc100443411)

[4.1.1 总体功能简介 7](#_Toc100443412)

[4.1.2 我的界面设计 7](#_Toc100443413)

[4.1.3 其他界面功能设计 8](#_Toc100443414)

[4.2 数据库结构分析 8](#_Toc100443415)

[4.2.1 数据库设计E-R图 8](#_Toc100443416)

[4.2.2 数据库表设计 9](#_Toc100443417)

[5 系统功能实现 13](#_Toc100443418)

[5.1 登录模块 13](#_Toc100443419)

[5.2 展示模块 16](#_Toc100443420)

[5.3 订单模块 18](#_Toc100443421)

[5.4 后台管理模块 27](#_Toc100443422)

[5.5 人脸识别入住模块 44](#_Toc100443423)

[6 系统测试 47](#_Toc100443424)

[6.1 测试目的 47](#_Toc100443425)

[6.2 功能模块测试用例 47](#_Toc100443426)

[6.2 测试总结 50](#_Toc100443427)

[7 总结与展望 51](#_Toc100443428)

基于微信小程序的酒店系统的设计与开发

李东福

摘要： 对于传统的使用人工的方式经营酒店带来的诸如查找、统计等问题，结合实际工作中酒店管理人员的实际需要，设计一个基于小程序的酒店管理系统。该系统的开发利用Go、JavaScript、go-zero、gorm以及MYSQL数据库，开发了给用户使用的小程序端和管理人员使用的Vue后台管理端，实现了用户在小程序端预览、预约、退房、收费、用户注册功能，管理人员查看房间状态，修改信息等功能，为用户提供了便捷的预览以及订退房环境，为酒店管理员提供了高效的工作平台。

关键词： go；mysql；javascript；微信小程序；酒店管理系统

Design and development of hotel management system based on WeChat applet

By

Li Dongfu

April, 2022

Abstract: For the traditional use of manual way to operate the hotel brought about by such problems as search, statistics, etc., combined with the actual needs of hotel managers in actual work, design a hotel management system based on small programs.The system uses Go, JavaScript, go-zero, gorm, and MYSQL database for development, provides a small program for users and a Vue background management terminal for administrators, and enables users to preview, reserve, Check-out, charging, user registration functions, administrators check room status, modify information and other functions, provide users with a convenient preview and check-out environment, and provide an efficient working platform for hotel administrators.

Keywords: go; mysql; javascript; wechat applet; hotel management system

# 1 绪论

## 1.1 课题背景及意义

### 1.1.1 课题背景

自第三次科技革命以来，从计算机诞生到计算机的普及，信息化和互联网已成为每个人生活中的必不可少的一部分，在很大程度上冲击着人们的生活。在效率为王的现代社会背景下，人们追求一种方便、快捷的生活方式。酒店作为人们旅游和外出的消费对象，随着人民从事劳动获得劳动报酬的能力与日俱增，在旅游和外出的消费也逐年提高，所以酒店需要有一种更加简单、高效的方式使得酒店的人力、物力上每个环节的充分投入与高效产出。[1] 为了占据市场，酒店需要严控质量关卡，不断在人员、物品上做到精益求精，这就离不开人工智能、数据分析等新兴的现代互联网技术。[2] 高度信息化的酒店管理系统能够明显减少人力成本以及运营成本，同时如果酒店管理系统具备完备的数据接口，就可以满足酒店的日常管理需求。[3] 要完成这一应用，需要深刻理解酒店相关业务并解决业内痛点，能够对数据进行大规模存储与整合，对系统的可扩展性也有一定的要求，从而达到提高酒店经济收益的目的。[4]

## 1.2 研究现状

从古至今，毫无疑问，信息技术能够推动人类生活不断进步，可见它在人类生产活动中扮演着非常重要的角色。信息技术起源于 19世纪上半叶，管理技术起源于 20 世纪 10 年代，但是两者的结合是起源于 20 世纪的 50 年代，自此信息管理系统出现在人们的日常生活中，信息管理系统要具备能够利用计算机软件对机构数据进行 收集、管理和存储等进行操作的能力，又由于互联网那时候没有发展起来，而这就构成了早期的单机型信息管理系统。

在国外，酒店的繁荣昌盛与计算机互联网的不断普及的共同影响，使得开发许多优秀酒店管理系统成为一种可能。[5] ECI 公司是众多优秀的酒店管理系统供应商之一，他们公司擅长的业务是对酒店数据进行收集、统计和分析的能力，这些信息处理的能力构成现代酒店管理系统的锥形。[6]

在我国，随着计算机行业的不断发展与信息产业的不断深化改革，一些国内酒店从实际问题出发并借鉴国外丰富优秀酒店管理经验，慢慢摸索出符合国内行情的酒店管理系统，这些开发酒店管理系统的专业技术公司一开始建立就蓬勃发展起来。但是迄今为止我国很多酒店依旧没有采用管理信息系统体系，建立组织的网络化、科学化、规范化的酒店管理系统。国内的酒店管理系统通常会由于对酒店业务理解不完整以及没有严格遵守软件工程生命周期而导致以下问题：

（1）开发周期长。

（2）系统功能不够全面。由于互联网的快速发展，人们对于网络的要求越来越高，某些功能不齐全的酒店管理系统已经遭到人们淘汰 。

（3）维护难。在单机系统下，服务器发生宕机，没有做容灾处理，就会产生整个系统的不可用，一旦系统不可用就会产生数据丢失问题，在当下数据即金钱的时代，这样是极其不可取的。

总的来说，本次论文设计将设计一种基于微信小程序的酒店管理系统，实现无纸化、自动化、高效化的酒店数据管理。在互联网科技高速发展的新时代，对酒店管理工作水平的提升具有积极的意义。

# 2 开发工具和开发平台

## 2.1关键性知识及技术简介

### 2.1.1 云开发技术

云开发就是使用各大厂商SDK调用相关函数，例如：文件存储、数据存储等基本服务，但是无须关注底层，即无须关注基础资源支持，可以提供各种API支持，这种模式本质上可以看成是一个Paas平台。云原生的概念开始爆火起来，就经历了Iaas到Paas再到Saas的一个过程。国内有条件的厂商都会自己去搭建自己的Paas或者Saas，这需要强大的基建能力，也需要对云服务有深度的认识。现在小型的网站上云更加划算，国内上云的厂商有华为云，阿里云，腾讯云，这样就无须关注底层基础资源细节，这个就是Iaas。所以微信小程序可以看成是一个Pass平台。而微信小程序的服务端，可以申请一个阿里云进行接口调用，数据处理与存储。[7] 这里的阿里云，本质上就是Iaas。

### 2.1.2 Go语言概述

Go语言诞生于2007年，2009年正式对外发布，经历了14年的时间，由Google公司发明。Go语言在云原生时代下火起来，著名的Docker以及Kubernetes就是使用Go语言进行编码的。[8]

Go语言特点：

1）开发效率高。体现在语法简单，编译速度快，部署运维也简单。编译完成之后的文件就是一个可执行文件。[9]

2）Go有比线程更加轻量级的协程机制，这是Go语言能够有高并发的基础。

3）Go在性能上，无论是运算性能还是运行空间上都会比各大面向对象的语言效果更好，唯一的无力感就是在go协程的时候应付不了计算密集型的函数。

## 2.2 数据库平台

利用MYSQL进行数据的增删改查操作，目前的体量不适宜进行分库分表操作，对分布式下的分库分表要考虑事务的二阶段提交；现阶段只需要考虑适当添加索引，对索引进行优化，如索引下推，最佳左前缀，这是因为Innodb的底层数据结构是B+树，但是不应该在经常增删改的字段增加索引，因为这时会触发底层操作系统层面的页分裂

利用Redis进行数据缓存，但是应该注意缓存穿透和缓存雪崩。警惕在分布式环境下注意主从模式引发脑裂问题进而发生数据不一致问题。

利用ELK挖掘出有价值的信息。

## 2.3 开发环境及工具

操作系统：Windows 10 Professional for Education

编程语言：JavaScript、Go

开发平台：VSCode、GoLand开发环境、微信开发者工具

开发环境：阿里云

数据库：MYSQL、Redis

## 2.4 系统部署

（1）申请一个阿里云公网IP。

（2）为公网IP配置域名。

（3）在微信开放社区中注册AppID。

（4）初始化云环境。

（5）利用Dockerfile文件在阿里云服务器上配置MYSQL等服务。

# 3 总体分析

## 3.1 可行性分析

（1）技术可行性

开发一款微信小程序，小程序界面以及交互方面，使用的技术栈与前端基本相同，不同点在于微信有自己独有的方法与名称。JS的知识更大多程序设计语言知识一样，有定义变量、变量赋值、循环分支等基础知识，以及前后端交互过程中需要的知识。而WXSS和CSS可以认为基本一致，所以针对有基础的前端开发工作者就不用花费太多的学习成本去学习WXSS。

酒店管理系统服务端采用Go语言作为后端语言，go-zero官网也有命令语句使用脚手架搭建整个项目结构，若不使用微服务（SOA的一种具体实现）的架构，不考虑领域驱动设计，对数据库进行垂直领域，竖直领域进行分层，只需要一开始使用api文件定义HTTP以及使用grpc定义rpc的Restful风格接口后，对数据进行组装，然后在api文件和proto文件生成的go文件的服务上进行数据层方法调用，这就是go-zero的API的创建过程，可以简单的创建出来。在一些相对较难的领域，直接使用小程序平台自带函数，这就为后台提供了支撑条件。

（2）经济可行性

当下的互联网环境，小规模的运营不会轻易的被整个微信生态卡脖子，不会被强制收取过高的手续费用，当盈利达到一定规模，就要考虑使用APP的方式或者其他营销推广的方式来宣传并使用，这是非常重要的一点。因为小程序天然对不同机型不同系统的手机适配，而安卓的APP或者苹果的APP并不互通下产生取舍问题，所以小程序在对比其他平台APP上更具优势；

小程序的创建本身并不需要成本，但是如果需要调用的API服务或数据过多，就会产生服务费用，按需扩容机制对于中小企业开发者来说都是一件好事。但是因为我事先已申请了阿里云的ECS和域名，所以将采用阿里云的方式搭建服务器，以及相关配置，这在数据体量不大的情况下，无疑是锻炼自己，提升自己的方式。本身小容量，小带宽，低核CPU的ECS并不需要花费太多成本，所以在经济上也是可以接受的。

## 3.2 功能需求分析

系统需求分析是整个软件生命周期中非常重要的一部分，拥有一份清晰的需求规格说明书将会对编码起到非常重要的作用。业内也有程序员只有20%的时间进行编码这一说法，而80%的时间是在弄清楚需求的说法。本系统主要实现酒店经营活动中一些常用的功能，最终为用户提供查看客房，预定客房等功能且后续持续可交付的简单易上手的酒店管理平台。[10]

系统使用前后端分离架构以及C/S混合B/S的模式进行功能的开发，包括查看房间、预定客房、查看订单、注册登录等功能。概括总结包括以下需求：

（1）首页通过地图API查询相关酒店位置。

（2）用户注册登录功能。

（3）在房型页可以查看相关房型。

（4）注册过的用户可以预约酒店的房间并选择入住时间。

（5）注册过的用户允许拨打电话。

（6）在我的页面能够查看订单，看到待支付，已支付，退款订单，点击后能对待支付订单进行取消订单或退款订单操作，能对已支付订单但是没有去住的酒店进行退款。

（7）查询优惠券。

（8）对应的后台查询。

## 3.3 数据库需求分析

本系统所使用的数据库是一个高版本关系型数据库MYSQL，不仅仅提供了对数据的增、删、改、查和事务回滚的能力，还可以使用Canal发送增量数据到消息队列再通过消息队列发送到异构数据库服务器以及搭建MYSQL的集群高可用架构MHA或MGR的高阶能力。使用MYSQL时，它需要开发者创建数据库、定义表的结构，然后才能添加数据。在本系统中，游客只能浏览酒店基本信息以及查看房源信息，而注册用户才能真正预定客房。根据酒店管理的需求，可将数据库划分为8个表：

（1）用户表（user）需要有微信生成的openid、手机号信息。

（2）管理员表（admin）需要有管理权限、用户名、密码。

（3）身份表（guest）需要有用户的id、身份证信息号码、名字信息。

（4）房间表（room）需要有房间号、房间类型、房间状态、额外备注、卫生间判断、电视判断等信息。

（5）价格表（price）需要有对应的房间类型以及价格信息。

（6）房间介绍表（room\_intro）需要有房间类型，卫生间判断、电视判断，窗户判断、房间描述等信息。

（7）订单表（room\_order）需要有用户id、房间id、预定时间、入住时间、退房时间、是否空房、房间类型等信息。

（8）旅店介绍表（hotel\_intro）需要有酒店名称、酒店介绍、酒店地址、酒店联系方式、酒店图片等信息。

## 3.4 性能需求分析

本系统主要是为了预定客房的用户以及查看酒店信息的管理员而进行设计的，所以需要满足UI设计美观以及前后端达到基本性能的要求，要求如下：

（1）页面直观易懂且美观。小程序的设计之初，应该尽可能站在用户的角度考虑问题，将小程序业务进行规整，理清每页小程序应该具备哪些功能，同时也需要尽可能提升小程序的美感，做到合理的排版布局，简约而不简单。在内容进行加载时能够区分当前页与其他页的不同，重要的用户操作信息能够得到反馈。底部Tab栏如图3.1所示。



图3.1 底部Tab栏图

（2）性能流畅，响应请求的时间控制在500ms以内。达到此要求，需要在开发的过程中提高小程序的响应速度以及后端处理数据及时且各个服务之间链路调用不会过长。从小程序上看，可以使用用户的本地缓存来减少用户请求，控制数据包的大小，使用异步请求来完成数据的及时响应。后端方面，优化数据库，在数据库表中加入索引，只传给前端需要使用的数据，而不是全量数据。同时也需要对业务进行清晰的梳理，了解整个系统的技术难点，在分布式系统下，尽量避免使用分布式事务等。

# 4 总体设计

## 4.1 实现功能简介

### 4.1.1 总体功能简介

本系统需要完成管理财务信息、注册登录、查看酒店基本信息、查看房源信息、管理房间、管理酒店信息等功能。本系统的功能结构如图4.1所示：

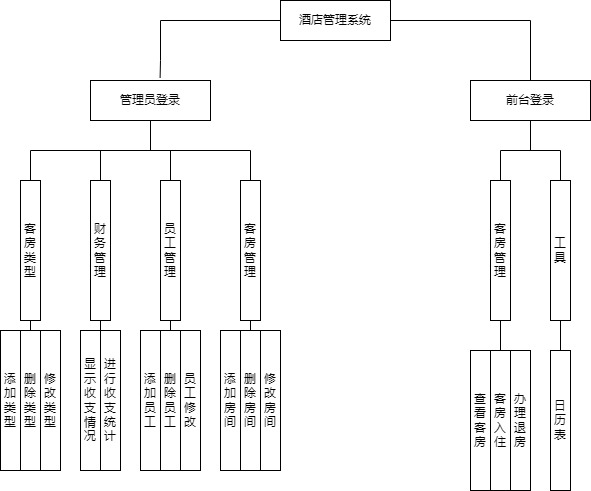


图4.1功能结构图

### 4.1.2 我的界面设计

考虑到数据的安全性，本系统在用户模块直接调用微信的认证系统，用户需要先去微信服务器上获得code，然后再通过网络传输发送code到后端，后端通过appid和secret以及code向微信服务器申请获得openid ，这时用户就相当于登录了，不在小程序直接生成openid的原因是openid不可变，直接用openid在http协议下明文传输有风险，这时候就需要使用https协议，进而需要进行加解密操作，从而影响服务器的一些性能。用户在登录之后就能够进行酒店预定，办理入住服务以及办理退房服务，查看订单情况等核心功能，而没有登录的有空只能够浏览首页的旅店信息以及预订页的房间信息。具体注册流程如图4.2所示。

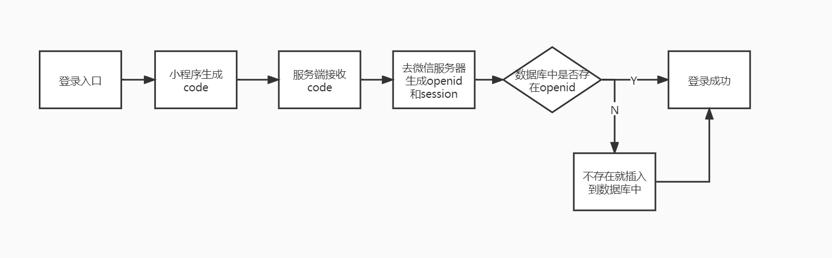


图4.2 注册流程图

### 4.1.3 其他界面功能设计

（1）登录模块：用户可以查看个人信息、查看订单完成情况。

（2）首页展示模块：展示酒店简介、酒店联系方式、酒店地址等信息

（3）房间展示模块：展示房间的各种类型信息

（4）支付模块：显示房型、预定时间、支付时间等交易信息

## 4.2 数据库结构分析

### 4.2.1 数据库设计E-R图

本系统数据库E-R图如图4.3所示。

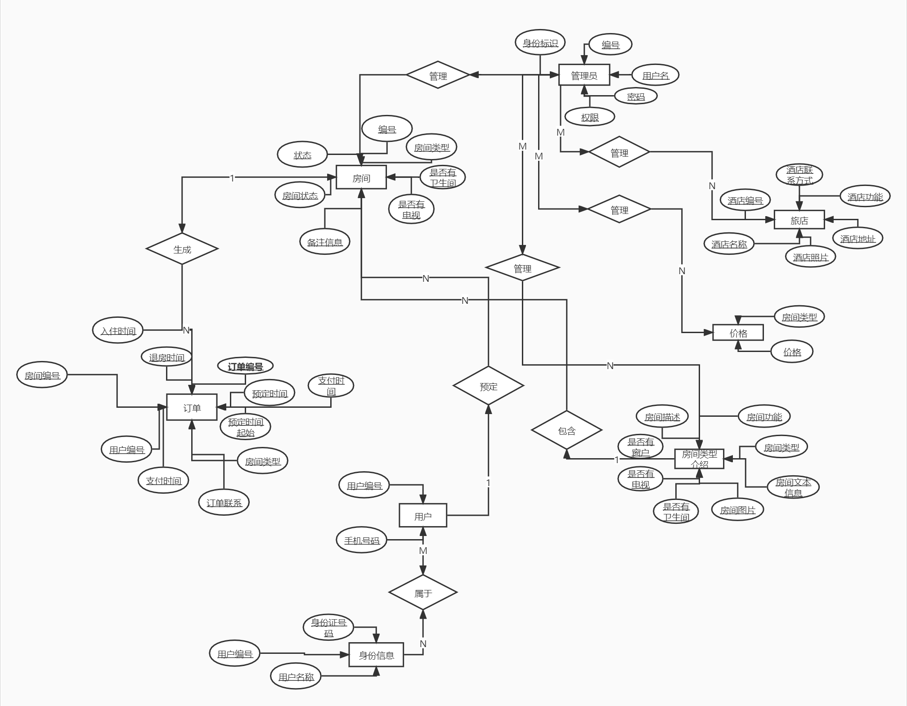


图4.3 数据库E-R图

### 4.2.2 数据库表设计

目前主流的数据库还是通过MYSQL来进行存储，一方面是因为MYSQL开源，另一方面关系型数据库对数据存储也有一定支撑以及符合一般的数据建模，所以本系统采用MYSQL来完成存储操作。

根据各个实体之间的逻辑关系设计了8个表结构，分别为admin(管理员表)、room(房间表)、room\_order(订单表)、guest(用户身份表)、price(价格表)、user(用户表)、hotel\_intro(旅店介绍表)、room\_intro(房间介绍表)。具体如下：

表4-1所示的user表用于记录用户的openid、手机号的个人信息。

表4-1 user表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 空 | 默认值 | 说明 |
| uid | varchar | NO |  | 主键 |
| 续表4-1 user表 | | | | |
| 字段名称 | 数据类型 | 空 | 默认值 | 说明 |
| phone | varchar | NO |  | 手机号 |

表4-2所示的admin表用于记录用户名、密码、权限等相关的活动信息。

表4-2 admin表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 空 | 默认值 | 备注 |
| aid | varchar | NO | auto\_increment | 主键 |
| username | varchar | NO |  | 用户名 |
| password | varchar | NO |  | 密码 |
| permission | int | NO |  | 权限等级 |
| idkey | varchar | NO |  | 身份标识 |

表4-3所示的room表用于记录房间的房间号、有无卫生间及电视、房间类型、房间状态等信息。

表4-3 room表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 空 | 默认值 | 说明 |
| rid | varchar | NO | auto\_increment | 主键 |
| number | varchar | NO |  | 房间号 |
| type | int | NO |  | 房间类型 |
| shower | int | NO |  | 判断卫生间 |
| tv | int | NO |  | 判断电视 |
| extra | varchar | YES |  | 额外说明 |
| img | varchar | YES |  | 房间照片 |
| state | int | NO |  | 状态 |

表4-4所示的room\_order 表用于记录房间类型、订单联系人、支付时间、用户订单的状态、入住时间、入住天数、订房时间等信息。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表4-4 room\_order表 | | | | |
| 字段名称 | 数据类型 | 空 | 默认值 | 说明 |
| oid | varchar | NO | auto\_increment | 主键 |
| uid | varchar | NO |  | 外键 |
| rid | varchar | YES |  | 外键 |
| pay\_time | bigint | YES |  | 支付时间 |
| reservation\_time | bigint | NO |  | 预定时间 |
| reservation\_during | bigint | NO |  | 预定天数 |
| check\_in\_time | bigint | YES |  | 入住时间 |
| check\_out\_time | bigint | YES |  | 退房时间 |
| state | int | NO |  | 订单状态 |
| type | int | NO |  | 房间类型 |
| contact | varchar | YES |  | 订单联系 |
| number | int | NO |  | 订单号码 |

表4-5所示的guest 表用于记录用户的身份信息，有用户身份证号码信息、用户姓名信息以及用户Id信息。

表4-5 guest表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 空 | 默认值 | 备注 |
| oid | varchar | NO |  | 外键 |
| gid | varchar | NO |  | 身份号码 |
| name | varchar | NO |  | 姓名 |

表4-6所示的price 表用于记录房间类型的价格。

表4-6 price表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 空 | 默认值 | 备注 |
| type | int | NO | NO | 房间类型 |
| price | int | NO |  | 价格 |

表4-7所示的hotel\_intro表是对旅店进行介绍，有名称、酒店介绍、酒店联系方式、酒店地址等信息。

表4-7 hotel\_intro表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 空 | 默认值 | 备注 |
| id | int | NO | auto\_increment | 主键 |
| name | varchar | NO |  | 酒店名称 |
| intro | varchar | YES |  | 酒店介绍 |
| options | varchar | YES |  | 酒店操作 |
| address | varchar | NO |  | 酒店地址 |
| contact | varchar | NO |  | 联系方式 |
| img | varchar | YES |  | 酒店图片 |

表4-8所示的room\_intro表用于记录的价格。

表4-8 room\_intro表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 空 | 默认值 | 备注 |
| type | int | NO | auto\_increment | 主键 |
| text | varchar | NO |  | 类型信息 |
| shower | int | NO |  | 判断厕所 |
| tv | int | NO |  | 判断电视 |
| window | int | NO |  | 判断窗户 |
| options | varchar | YES |  | 可选操作 |

续表4-8 room\_intro表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 空 | 默认值 | 备注 |
| img | varchar | YES |  | 房间图片 |
| intro | varchar | YES |  | 房间描述 |

# 5 系统功能实现

## 5.1 登录模块

用户登录流程如图5.1所示。

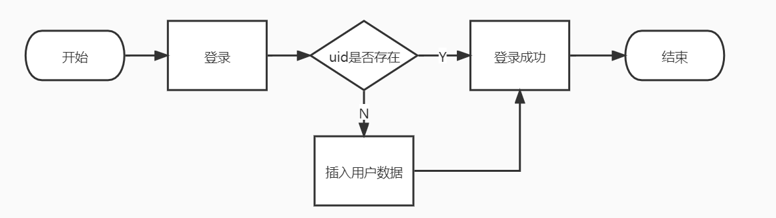


图5.1 登录流程图

界面如图5.2所示，可实现用户注册与登录功能。

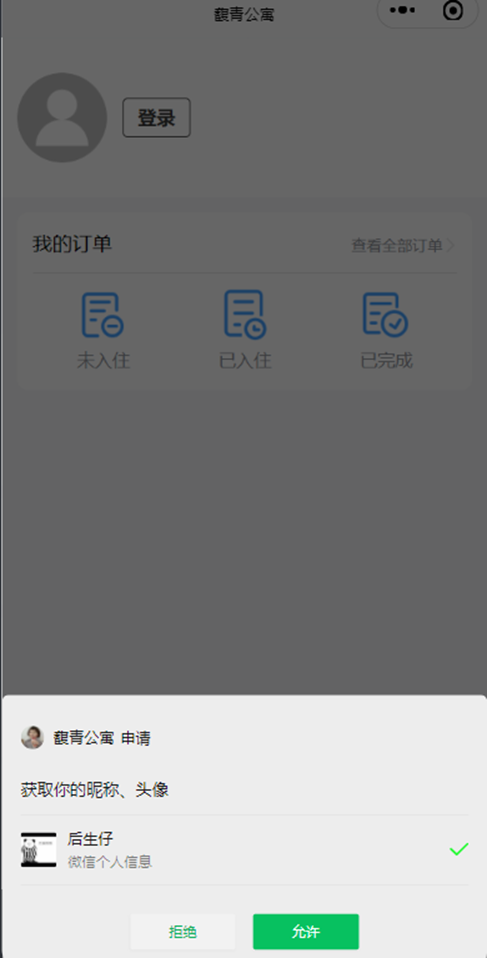


图5.2 登录界面图

部分相关代码如下：

// 登录功能

wx.login({

success: loginRes => {

loginUserRequest(loginRes.code).then(res => {

if (res.state) {

this.globalData.phone = res.data.phone;

console.log(loginRes.code);

wx.setStorage({

data: res.data.sessionId,

key: 'sessionId',

});

}

else {

console.log(res.msg);

}

})

}

});

// 增加缓存等代码

getUserProfile() {

console.log(12);

wx.getUserProfile({

desc: 'login',

success: (res) => {

app.globalData.userInfo = res.userInfo;

this.setData({

userInfo: res.userInfo,

avatarUrl: res.userInfo.avatarUrl,

hasUserInfo: true,

});

wx.setStorageSync('userInfo', res.userInfo);

if (!app.globalData.phone) {

this.open();

}

},

fail: (err) => {

console.log(err)

}

})

}

## 5.2 展示模块

展示流程如图5.3所示。

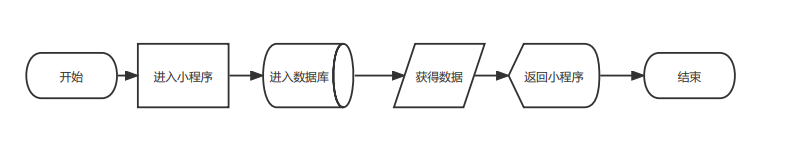


图5.3 展示流程图

界面如图5.4所示，无论是用户还是游客，都可以对小程序内的酒店信息、房间类型信息以及房间具体信息进行查看。

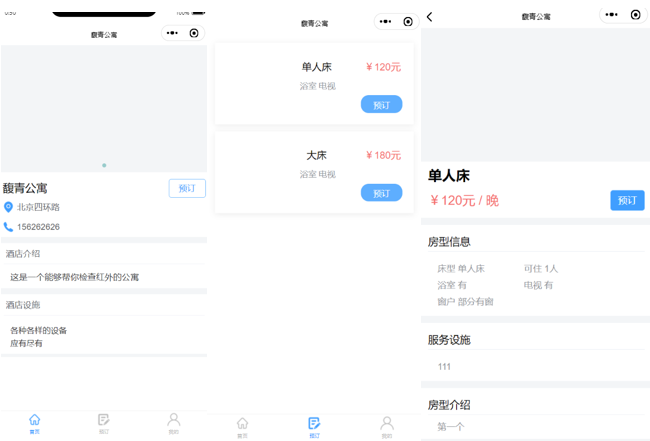


图5.4 展示界面图

部分相关代码：

// 旅店展示代码

getHotelIntroRequest().then(res => {

if (res.state) {

res.hotelIntro.optionsArray = res.hotelIntro.options.split(';');

this.setData({

hotelIntro: res.hotelIntro,

});

}

else {

console.log(res.msg);

}

}

// 房型展示代码

wx.getStorage({

key: 'roomIntro',

success: (res) => {

this.setData({

roomIntro: res.data,

})

}

})

// 房间展示代码

function (options) {

wx.getStorage({

key: 'roomIntro',

success: (res) => {

this.setData({

room: res.data[options.index],

total: res.data[options.index].price,

})

}

});

}

## 5.3 订单模块

界面如图5.5所示，已登录的用户，可以进行预定酒店客房的操作，以及查看未入住、已入住、已完成的订单情况。

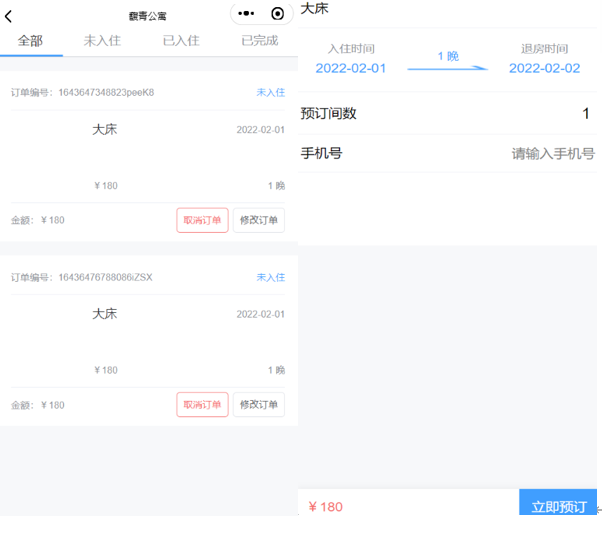


图5.5 订单界面图

部分相关代码：

// 查看订单情况并按实际情况进行修改

onLoad() {

if (app.globalData.userInfo) {

this.setData({

userInfo: app.globalData.userInfo,

avatarUrl: app.globalData.userInfo.avatarUrl,

hasUserInfo: true

});

}

if (app.globalData.phone) {

this.setData({

hasPhone: true

});

}

},

getUserInfo(e) {

if (e.detail.userInfo) {

app.globalData.userInfo = e.detail.userInfo;

this.setData({

userInfo: e.detail.userInfo,

avatarUrl: e.detail.userInfo.avatarUrl,

hasUserInfo: true,

});

if (!app.globalData.phone) {

this.open();

}

}

else {

console.log('pity');

}

},

getUserProfile() {

console.log(12);

wx.getUserProfile({

desc: 'login',

success: (res) => {

app.globalData.userInfo = res.userInfo;

this.setData({

userInfo: res.userInfo,

avatarUrl: res.userInfo.avatarUrl,

hasUserInfo: true,

});

wx.setStorageSync('userInfo', res.userInfo);

if (!app.globalData.phone) {

this.open();

}

},

fail: (err) => {

console.log(err)

}

})

},

getPhoneNumber (e) {

this.setData({

show: false

});

if (e.detail.encryptedData) {

getPhoneRequest(wx.getStorageSync('sessionId'), e.detail.encryptedData, e.detail.iv).then(res => {

if (res.state) {

app.globalData.phone = res.phone;

this.setData({

hasPhone: true,

});

wx.showToast({

title: '绑定成功',

icon: 'success',

duration: 600

})

}

}).catch(err => {

console.log(err);

});

}

else {

console.log('getPhone error');

this.setData({

hasPhone: false

});

}

},

open() {

this.setData({

show: true

});

},

toOrder(e) {

wx.navigateTo({

url: '../order/order?state=' + e.currentTarget.dataset.state,

})

}

});

// 搜索投票

inputTyping: function (e) {

let that = this

that.setData({

inputVal: e.detail.value

});

// 进行预定客房的操作

onLoad: function (options) {

if (options.oid) { // 修改订单

getUserOrderRequest(wx.getStorageSync('sessionId'), options.oid).then(res => {

this.setData({

room: res,

phone: res.contact,

['date.start']: getFormatDate(res.reservation\_time),

['date.end']: getFormatNextDate(res.reservation\_time, res.reservation\_during),

['date.reservation\_time']: res.reservation\_time,

['date.reservation\_during']: res.reservation\_during,

number: res.number,

oid: options.oid,

});

});

}

else { // 新订单

const app = getApp();

this.setData({

phone: app.globalData.phone,

});

wx.getStorage({

key: 'roomIntro',

success: (res) => {

this.setData({

room: res.data[options.type],

total: res.data[options.type].price,

});

}

})

}

},

startDateChange: function(e) {

let startTime = new Date(e.detail.value).getTime();

let endTime = new Date(this.data.date.end).getTime();

let during = (endTime - startTime) / 86400000

this.setData({

['date.start']: e.detail.value,

['date.reservation\_time']: startTime,

['date.reservation\_during']: during,

total: this.data.number \* this.data.room.price \* during,

})

},

endDateChange: function(e) {

let startTime = new Date(this.data.date.start).getTime();

let endTime = new Date(e.detail.value).getTime();

let during = (endTime - startTime) / 86400000

this.setData({

['date.end']: e.detail.value,

['date.reservation\_time']: startTime,

['date.reservation\_during']: during,

total: this.data.number \* this.data.room.price \* during,

});

},

minusNumber() {

(throttle(() => {

this.setData({

number: this.data.number > 1 ? --this.data.number : 1,

total: this.data.number \* this.data.room.price \* this.data.date.reservation\_during,

});

}, 100))();

},

addNumber() {

(throttle(() => {

this.setData({

number: ++this.data.number,

total: this.data.number \* this.data.room.price \* this.data.date.reservation\_during,

});

}, 80))();

},

reserve() {

let data = {

phone: this.data.phone,

number: this.data.number,

sessionId: wx.getStorageSync('sessionId'),

reservation\_time: this.data.date.reservation\_time,

reservation\_during: this.data.date.reservation\_during,

type: this.data.room.type,

oid: this.data.oid,

};

if (data.reservation\_during < 1) {

wx.showToast({

title: '不能小于1晚',

icon: 'error',

duration: 800

});

return;

}

wx.showLoading({title: '加载中'});

reserveRequest(data).then(res => {

wx.hideLoading();

wx.showToast({

title: res.msg,

icon: res.state ? 'success' : 'error',

duration: 600

});

if (res.state) {

wx.switchTab({

url: '/pages/reserve/reserve'

});

}

}).catch(err => {

console.log(err);

wx.showToast({

title: '失败',

icon: 'error',

duration: 600

});

});

},

})

## 5.4 后台管理模块

后台管理模块分为登录模块，以及管理模块。管理模块分为基本的房间管理、酒店信息管理、订单管理和人员账号管理。

界面如图5.6所示，管理员进行登录操作，每个管理员都有对应的权限，登录完成后有Token令牌才能进行相应权限的酒店的管理。

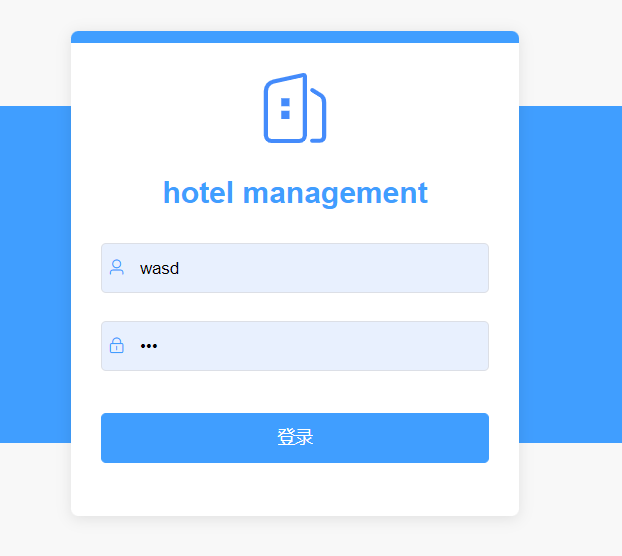


图5.6 后台登录界面图

部分相关代码：

//进行登录操作，生成token并发送

router.post('/login', (req, res) => {

queryAdminByUsername(req.body.username).then(result => {

console.log(result);

if (result.length && encryptPasswordSync(req.body.password) === result[0].password

) {

const token = getToken({ username: result[0].username, permission: result[0].permission });

res.json({ state: true, token });

}

else {

res.json({ state: false, msg: '用户名或密码错误' });

}

}).catch(err => {

console.log(err);

});

});

界面如图5.7所示，管理员在订单管理中，可以查询未入住、已入住、已退房的订单，并可为未入住用户办理入住操作。

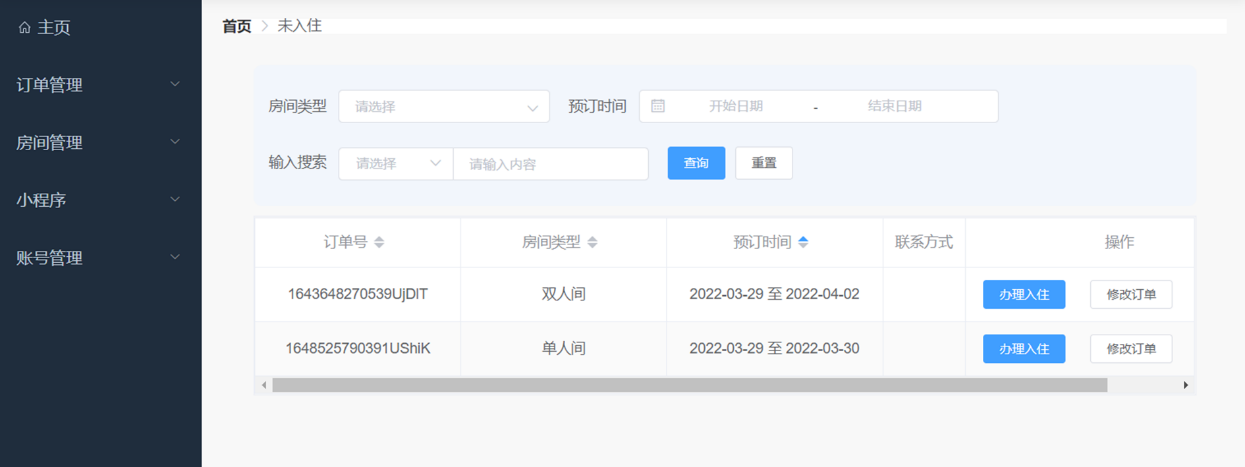


图5.7 后台订单管理界面图

部分相关代码

//获得订单相关代码

router.post('/getOrders', (req, res) => {

if (req.body.state === 0) {

queryAllOrders(req.body.state).then(result => {

res.json(result);

})

}

else {

queryAllOrdersOwnRoom(req.body.state).then(result => {

res.json(result);

}).catch(err => {

console.log(err);

});

}

});

//办理入住相关代码

router.post('/checkIn', async (req, res) => {

let obj = req.body.obj;

let contact = req.body.contact;

let guests = req.body.guests;

for (let guest of guests) {

if (!validateGid(guest.gid)) {

return res.json({ state: false, msg: '身份证格式不对' });

}

}

// 已预定

if (obj.uid) {

try {

let result = await queryOrderByOid(obj.oid);

if (result.length) {

await updateRoomStateByNumber(1, obj.number);

let roomResult = await queryRoomByNumber(obj.number);

await checkInApi({ rid: roomResult[0].rid, check\_in\_time: obj.check\_in\_time, state: 1 }, contact, obj.oid);

let values = [];

for (let guest of guests) {

values.push([...new Guest(obj.oid, guest.gid, guest.name)]);

}

await insertGuest(values);

return res.json({ state: true, msg: '办理成功' });

} else {

return res.json({ state: false, msg: '未预订房间' });

}

} catch (err) {

console.log(err);

}

}

else {

try {

let result = await queryRoomByNumber(obj.number);

if (result.length) {

await updateRoomStateByNumber(1, obj.number);

let order = new Order(createOid(), result[0].rid, null, obj.place\_time, obj.reservation\_time, obj.reservation\_during,

obj.check\_in\_time, null, 1, obj.type, contact, 1);

await insertOrder(order);

let values = [];

for (let guest of guests) {

values.push([...new Guest(order.oid, guest.gid, guest.name)]);

}

await insertGuest(values);

return res.json({ state: true, msg: '办理成功' });

} else {

return res.json({ state: false, msg: '房间不存在' });

}

} catch (err) {

console.log(err);

}

}

});

//删除订单相关代码

router.post('/deleteOrder', (req, res) => {

deleteOrder(req.body.oid).then(result => {

if (result.affectedRows) {

if (req.body.rid) {

updateRoomStateByRid(0, req.body.rid).then(() => {

res.json({ state: true, msg: '删除成功' });

}).catch(err => {

console.log(err);

});

}

res.json({ state: true, msg: '删除成功' });

}

else {

res.json({ state: false, msg: 'error' });

}

}).catch(err => {

console.log(err);

});

});

//得到订单明细信息

router.post('/getOrderInfo', (req, res) => {

if (req.body.oid) {

queryOrderByOid(req.body.oid).then(result => {

if (result.length) {

res.json({ state: true, order: result[0] });

}

else {

res.json({ state: false, msg: '无此订单' });

}

}).catch(err => {

console.log(err);

});

}

else if (req.body.uid) {

queryReservedByUid(req.body.uid).then(result => {

if (result.length) {

res.json({ state: true, order: result[0] });

}

else {

res.json({ state: false, msg: '无此订单' });

}

}).catch(err => {

console.log(err);

});

}

});

界面如图5.8所示，管理员在房间管理中，包括了查看房间、添加新房间以及修改房间价格。对于没有在小程序预定的用户，可在酒店前台直接办理入住操作。



图5.8 后台房间管理界面图

部分相关代码如下：

// 添加房间

router.post('/addRoom', (req, res) => {

queryRoomByNumber(req.body.number).then((result) => {

if (result.length <= 0) {

if (!validateRoomNumber(req.body.number)) {

return res.json({ state: false, msg: '房间号为长度3-8的数字或字母' });

}

const room = new Room(null, req.body.number, req.body.type, req.body.shower, req.body.tv, req.body.extra, '', 0);

insertRoom(room).then(() => {

res.json({ state: true, msg: '添加成功' });

}).catch(err => {

console.log(err);

})

}

else {

res.json({ state: false, msg: '该房间号已存在' });

}

}).catch(err => {

console.log(err);

});

});

//删除房间

router.post('/deleteRoom', async (req, res) => {

try {

let result = await queryRoomByNumber(req.body.number);

if (result.length && result[0].state === 1) {

return res.json({ state: false, msg: '此房间已被预订，暂不可删除' });

}

let deleteResult = await deleteRoomByNumber(req.body.number)

if (deleteResult.affectedRows) {

let dirPath = path.join(\_\_dirname, '../../static/img');

let name = req.body.number + path.extname(req.body.img);

try {

fs.unlinkSync(dirPath + '\\' + name);

} catch {

console.log('image not exists');

}

return res.json({ state: true, msg: '删除成功' });

}

else {

return res.json({ state: false, msg: 'error' });

}

} catch (err) {

console.log(err);

}

});

//更新房间

router.post('/updateRoom', async (req, res) => {

try {

// 状态检测

let result = await queryRoomByNumber(req.body.number);

if (result.length > 0 && result[0].state) {

return res.json({ state: false, msg: '该房间正在使用，暂不可更改' });

}

// 重名检测

if (req.body.oldNumber !== req.body.number) {

let result = await queryRoomByNumber(req.body.number);

if (result.length > 0) {

return res.json({ state: false, msg: '该房间号已存在' });

}

}

if (!validateRoomNumber(req.body.number)) {

return res.json({ state: false, msg: '房间号为长度3-8的数字或字母' });

}

const room = new Room(null, req.body.number, req.body.type, req.body.shower, req.body.tv, req.body.extra, '', 0);

Reflect.deleteProperty(room, 'rid');

Reflect.deleteProperty(room, 'img');

Reflect.deleteProperty(room, 'state');

await updateRoomByNumber(room, room.number);

return res.json({ state: true, msg: '更改成功' });

} catch (err) {

console.log(err);

}

});

//修改房间价格信息

router.post('/updatePrice', async (req, res) => {

try {

let priceArray = [];

priceArray.push([0, req.body.kingRoomPrice]);

priceArray.push([1, req.body.singleRoomPrice]);

priceArray.push([2, req.body.doubleRoomPrice]);

for (let array of priceArray) {

if (!validatePrice(array[1])) {

return res.json({ state: false, msg: '金额只能为数字' });

}

}

let result = await queryAllPrice();

if (result.length) { // 更新

await Promise.all([updatePrice(priceArray[0].reverse()), updatePrice(priceArray[1].reverse()), updatePrice(priceArray[2].reverse())]);

}

else { // 插入

await insertPrice(priceArray);

}

return res.json({ state: true, msg: '修改成功' });

} catch (err) {

console.log(err);

return res.json({ state: false, msg: 'error' });

}

});

界面如图5.9所示，管理员在小程序管理中可以查看和修改酒店简介和房间描述信息。

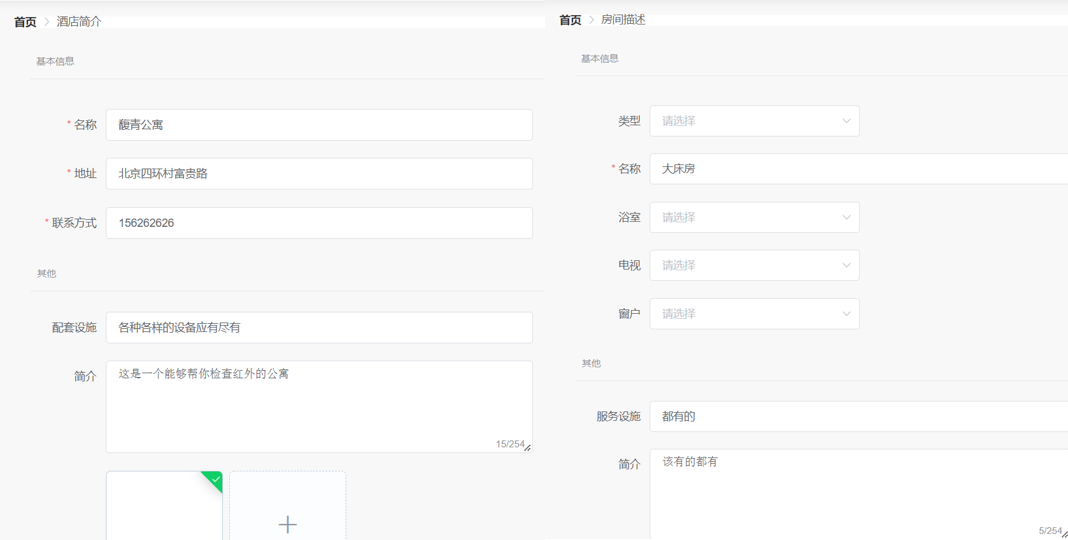


图5.9 小程序管理界面图

部分相关代码：

//更新酒店信息

router.post('/updateHotelIntro', async (req, res) => {

try {

const hotelIntro = new HotelIntro(1, req.body.name, req.body.intro, req.body.options, req.body.address, req.body.contact, '');

let result = await queryHotelIntro();

if (result.length) { // 更新

Reflect.deleteProperty(hotelIntro, 'id');

Reflect.deleteProperty(hotelIntro, 'img');

await updateHotelIntro(hotelIntro, 1);

}

else { // 插入

await insertHotelIntro(hotelIntro);

}

return res.json({ state: true, msg: '成功' });

} catch (err) {

console.log(err);

}

});

//更新酒店房间信息

router.post('/updateRoomIntro', async (req, res) => {

try {

const roomIntro = new RoomIntro(req.body.type, req.body.text, req.body.shower, req.body.tv, req.body.window, req.body.options, '', req.body.intro);

let result = await queryRoomIntroByType(req.body.type);

if (result.length) { // 更新

Reflect.deleteProperty(roomIntro, 'type');

Reflect.deleteProperty(roomIntro, 'img');

await updateRoomIntroByType(roomIntro, req.body.type);

}

else { // 插入

await insertRoomIntro(roomIntro);

}

return res.json({ state: true, msg: '成功' });

} catch (err) {

console.log(err);

}

});

界面如图5.10所示，账号管理中，查看用户列表，高权限用户修改低权限用户，并可以进行删除操作。超级管理员可以对用户进行注册操作。



图5.10 人员管理界面图

部分相关代码：

//查看管理员基本信息

router.post('/getAdmins', (req, res) => {

console.log(req.body)

console.log((req.body.username!="") && (req.body.password!=""))

if ((req.body.username!="") && (req.body.password!="")) {

console.log(1)

queryAllAdmins().then(result => {

console.log(result)

for (let admin of result) {

Reflect.deleteProperty(admin, 'password');

Reflect.deleteProperty(admin, 'idkey');

}

console.log(result);

res.json(result);

}).catch(err => {

console.log(err);

});

}

});

//添加管理人员

router.post('/register', (req, res) => {

queryAdminByUsername(req.body.username).then(result => {

if (result.length <= 0) {

const validation = validateUser(req.body); // 校验

if (!validation.state) {

res.json(validation);

}

const user = new User(null, req.body.username, encryptPasswordSync(req.body.password), req.body.permission, req.body.idkey);

insertAdmin(user).then(() => {

res.json({ state: true, msg: '注册成功' });

}).catch(err => {

console.log(err);

})

}

else {

res.json({ state: false, msg: '该用户已存在' });

}

}).catch(err => {

console.log(err);

});

});

//更新密码

router.post('/updatePassword', (req, res) => {

queryAdminByUsername(req.body.username).then(result => {

if (result.length && result[0].idkey === req.body.idkey) {

if (validatePassword(req.body.password)) {

updateAdminPassword(encryptPasswordSync(req.body.password), req.body.username).then(() => {

res.json({ state: true, msg: '修改成功' });

}).catch(err => {

console.log(err);

});

}

else {

res.json({ state: false, msg: '密码为长度4-20的数字或字母或下划线' })

}

}

else {

res.json({ state: false, msg: '密保错误' });

}

}).catch(err => {

console.log(err);

});

});

//更新权限

router.post('/updatePermission', (req, res) => {

updatePermission(req.body.username, req.body.permission).then(result => {

if (result.affectedRows) {

res.json({ state: true, msg: '修改成功' });

}

else {

res.json({ state: false, msg: 'error' });

}

}).catch(err => {

console.log(err);

});

});

## 5.5 人脸识别入住模块

界面如图5.11所示，因为疫情降低接触，简易模拟人脸识别功能，但是具体是要进行人脸核身功能，用户通过门前人脸核身设备进入酒店房间。



图5.11 人脸识别界面图

部分相关代码：

//前端部分代码

// 人脸面部勘探轮廓识别绘制

async fnRunFaceLandmark() {

console.log("RunFaceLandmark");

if (this.videoEl.paused) return clearTimeout(this.timeout);

//识别绘制人脸信息

const result = await faceapi[this.detectFace](

this.videoEl,

this.options

).withFaceLandmarks();

if (result && !this.videoEl.paused) {

const dims = faceapi.matchDimensions(this.canvasEl, this.videoEl, true);

const resizeResult = faceapi.resizeResults(result, dims);

// 在这里准备开始到了人脸的信息

let ctx = this.canvasEl.getContext("2d");

// 把当前视频帧内容渲染到canvas上

ctx.drawImage(this.videoEl, 0, 0);

// 转base64格式、图片格式转换、图片质量压缩

var imgbase64 = this.canvasEl.toDataURL("image/jpeg", 1);

axios.post("http://localhost:8081/faceSearch", {

imagebase64: imgbase64,

})

.then(function (response) {

console.log(response)

let username = response.data.username

if (response.data.username != null) {

console.log("人脸识别成功");

document.getElementById("showusername").innerHTML = "你好" + username

}

})

.catch(function (error) {

console.log(error);

});

//后端相关代码

func (l \*FaceSearchLogic) FaceSearch(req types.FaceSearchReq) (resp \*types.FaceSearchResp, err error) {

// todo: add your logic here and delete this line

if req.Imagebase64 == "" {

return &types.FaceSearchResp{

},err

}

credential := common.NewCredential(

l.svcCtx.Config.TenXunCloud.SecretId,

l.svcCtx.Config.TenXunCloud.SecretKey,

)

cpf := profile.NewClientProfile()

cpf.HttpProfile.Endpoint = "iai.tencentcloudapi.com"

client, \_ := iai.NewClient(credential,l.svcCtx.Config.TenXunCloud.Region, cpf)

request := iai.NewSearchFacesRequest()

request.GroupIds = common.StringPtrs([]string{ "handsome" })

request.Image = common.StringPtr(req.Imagebase64)

//request.Url = common.StringPtr("1")

request.MaxFaceNum = common.Uint64Ptr(1)

request.MinFaceSize = common.Uint64Ptr(80)

request.MaxPersonNum = common.Uint64Ptr(1)

request.NeedPersonInfo = common.Int64Ptr(1)

request.QualityControl = common.Uint64Ptr(3)

request.FaceMatchThreshold = common.Float64Ptr(80)

request.NeedRotateDetection = common.Uint64Ptr(1)

response, err := client.SearchFaces(request)

if \_, ok := err.(\*errors.TencentCloudSDKError); ok {

fmt.Printf("An API error has returned: %s", err)

return &types.FaceSearchResp{},nil

}

fmt.Printf("%s", response.ToJsonString())

fmt.Println()

name := response.Response.Results[0].Candidates[0].PersonName

fmt.Printf("%s", name)

fmt.Println()

nicket := \*name

fmt.Printf("%s",nicket)

fmt.Println()

return &types.FaceSearchResp{

\*name,

},nil

# 6 系统测试

## 6.1 测试目的

软件测试就是尽可能地检查、发现并代码中存在的问题，从性能和功能上考虑系统的健壮性，从而有效提高系统数据安全、系统运行安全，最终尽最大可能规避软件各大小版本发布后带来的风险等。

## 6.2 功能模块测试用例

对系统主要功能模块测试的用例表如表6-1~表6-12所示。

表6-1 管理员注册

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |
| 1 | 必填项 | 令某项必填数据为空 | 提示请输入xxx | 成功 |
| 2 | 用户名唯一性 | 输入已注册过的用户名 | 提示用户名已存在 | 成功 |

续表6-1 管理员注册

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |
| 3 | 提交注册信息 | 输入未注册过的用户名、填写必填项、输入正确的格式 | 注册成功，跳转到登录页面 | 成功 |

表6-2 管理员登录测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |
| 1 | 必填项 | 令某项必填数据为空 | 提示请输入xxx | 成功 |
| 2 | 密码正确性 | 输入错误的密码 | 提示密码错误 | 成功 |
| 3 | 用户名有效性 | 输入未注册的用户名 | 提示用户名不存在 | 成功 |
| 4 | 提交登录信息 | 输入注册过的用户名、正确的密码 | 登录成功，跳转到首页 | 成功 |

表6-3 管理员修改密码测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |
| 1 | 密码格式验证 | 输入错误格式密码 | 提示密码为长度4-20的数字或字母或下划线 | 成功 |
| 2 | 密保正确性 | 输入错误密保信息 | 提示密保错误 | 成功 |
| 3 | 提交修改信息 | 输入正确密码格式和密保 | 提示修改成功 | 成功 |

表6-4 获取管理员信息测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |

续表6-4 获取管理员信息测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |
| 1 | 用户名是否存在 | 输入错误用户名 | 提示用户名不存在 | 成功 |
| 2 | 查看管理员信息 | 输入正确用户名 | 成功返回管理员的信息 | 成功 |

表6-5 添加新管理员测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |
| 1 | 添加新管理员 | 输入正确密码和idkey | 成功将该用户添加成管理员 | 成功 |

表6-6 删除管理员测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |
| 1 | 删除管理员 | 输入管理员用户名 | 成功将该管理员删除 | 成功 |
| 2 | 二级管理删除一级管理 | 登录二级管理账号并删除一级管理员账号 | 不能删除 | 成功 |

表6-7 获取房间信息测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |
| 1 | 房间号验证 | 输入错误房间号 | 提示无此房间 | 成功 |
| 2 | 提交房间号信息 | 输入正确房间号 | 成功查询该房间号的房间信息 | 成功 |

表6-8 添加新房间测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |
| 1 | 房间号格式验证 | 输入错误格式房间号 | 提示房间号为长度3-8的数字或字母 | 成功 |
| 2 | 房间号唯一性 | 输入已存在的房间号 | 提示该房间号已存在 | 成功 |

续表6-8 添加新房间测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |
| 3 | 提交房间号 | 输入正确格式的房间号 | 提示添加成功 | 成功 |

表6-9 修改房间测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |
| 1 | 房间状态时权限 | 输入被入住的房间号 | 提示该房间正在使用，暂不可修改 | 成功 |
| 2 | 房间号唯一性 | 输入准备修改的房间号 | 提示该房间号已存在 | 成功 |
| 3 | 提交修改的房间信息 | 输入准备修改的房间信息 | 提示修改成功 | 成功 |

表6-10 上传图片测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |
| 1 | 图片格式验证 | 上传错误图片 | 提示上传失败 | 成功 |
| 3 | 提交图片信息 | 上传正确图片 | 提示上传成功 | 成功 |

表6-11 查看房间状态测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |
| 1 | 房间状态 | 输入正在使用的房间号 | 提示此房间已被预定，暂不可删除 | 成功 |
| 3 | 提交房间号 | 输入正确格式的房间号 | 提示删除成功 | 成功 |

表6-12 获取订单信息测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 用例说明 | 操作步骤 | 预期结果 | 测试结果 |
| 1 | 订单是否存在 | 输入错误订单号/客户id | 提示无此订单 | 成功 |
| 2 | 提交订单信息 | 输入正确订单号/客户id | 查询出订单信息 | 成功 |

## 6.2 测试总结

本系统的测试用例都是针对系统的每个功能点对应的函数进行全方位测试，确保基本符合业务功能以及业务需求，考虑了系统安全维度的代码健壮性，最终达到系统稳定高可用的状态。

# 7 总结与展望

在系统设计方面，采用前后端分离技术进行开发；在需求层面，确定划分四大主要功能模块，分别为个人中心模块、预定客房模块、预览房型及旅店模块、后台的酒店管理模块；在数据库设计方面，通过分析每个模块必要字段来构建数据表字段，并通过E-R图清晰地描述出每个实体的逻辑关系以及每个实体对应的具体属性；最后数据流方面，确定主要功能中数据的流动方向。在系统的实现阶段，本系统使用前端技术体系以及Go语言技术进行开发。

在首页中，游客可以浏览基本的房源信息以及酒店信息，但是只有使用了微信登录的方式的用户在后台生成了openid才能进行预定客房，查看订单等核心功能。在界面设计上，尽可能设计出美观、易懂的页面。

由于前端开发经验的不足且自身能力有限，不能做到酒店管理小程序的尽善尽美，后期仍然有待改善，例如添加更多的功能，包括对订单进行智能化的分析、管理员对用户恶意退单行为进行拉黑处理、对一些跨网站脚本攻击进行防护、对单位时间内每个用户流量进行一定的限制来防止恶意破坏网站。还有缺乏设计模式的丰富思想，可能会导致后期代码的维护困难，因此后期需要对代码进行优化，例如采用设计模式对代码进行重构，引入消息队列对数据进行解耦、用缓存技术缓存热点数据、对数据进行分库分表、代码架构从单体架构转变为微服务或云原生方向靠拢。

参考文献：

1. 徐耀荣. 基于Java Web的小型酒店管理系统[D].大连理工大学,2018.
2. 陈雷. 基于B/S中小型酒店客房管理系统的设计与实现[D].江西财经大学,2016.
3. 郝一鸣. 基于B/S模式的酒店管理系统设计及实现[D].西安电子科技大学,2017.
4. 蒋曰钦. 酒店管理系统的设计与实现[D].青岛大学,2019.
5. 吕治江. 中小型酒店管理系统的设计与实现[D].厦门大学,2017.
6. 娄圣军. 酒店管理系统的设计与实现[D].青岛大学,2017.
7. 张利香,甘发旺,薄建国.基于微信小程序与云开发的成绩查询系统实现[J].内蒙古科技与经济,2019(13):59-60.
8. Krill, Paul. Google goes forward with Go language[J]. InfoWorld.com,2009:
9. Paul Krill. Go 1.17 brings language and compiler enhancements[J]. InfoWorld.com,2021:

[10] Frances L. Van Voorhis. Reflecting on the Homework Ritual: Assignments and Designs[J]. Theory into Practice,2004,43(3).

致谢

时间过得真快，四年大学生活转瞬即逝。在这四年里我收获了知识、收获了友情、收获了成长。经过近三个月的学习和努力，在导师的悉心指导和监督下，我终于完成了这篇毕业论文。

在整个毕业设计与论文撰写过程中，让我深刻的体会到自己的专业知识还不够全面，想尝试微服务项目但是又对分布式、微服务理解不够深刻，导致写起来困难重重，以及平时写前端机会太少导致的技术水平欠缺，在开发过程中我遇到了些许的问题，但每解决完一个难点或者优化代码，我就会获得成就感，让我充分体会到了编码不易，报错频繁，探索艰辛和攀登之喜，在这过程中所培养的能力对我未来的学习和工作有很大的帮助。通过毕业设计，我收获了很多，不仅是编码方面，还有整个系统的开发流程，哪种系统应该用哪种开发模型，在浩瀚的书海里浏览以及对自动化测试、部署和运维有了更进一步的理解。这其中不仅仅是我个人的努力，还有一些关心我的人支持我的人的鼓励和帮助。

首先感谢我的指导老师李贞辉老师，感谢他一开始就选择指导我毕业设计和毕业论文，从侧面上认可我的能力，让我相信自己的能力。感谢他即使在凌晨也来帮我审查、修改我的论文。老师的检查总是很细致入微，可以认真地看论文中的每一个细节的格式要求，同时为我的论文撰写提供了很多指导意见。

另外感谢身边的同学、朋友，甚至是未曾谋面但帮助过我解决编程问题、算法问题的网友，感谢他们陪伴我度过了精彩、充实的大学四年。

最后，我要特别真诚地感谢我的父母。只要我有想法，他们都会在精神上和物质上给予他们能够给予的最大帮助，在目前人生的迷茫阶段他们成为我前进的推动力。

在今后的日子里，我要不断地充实自己。只有学无止境、与时俱进、不断提高自己的能力才能坦然地面对时代潮流的巨浪，最后争取在所学领域有所作为。