即时排队系统

需求分析文档

2021

**Abstract**: In recent years, our people's living standard has been greatly improved, people's production and life style has changed a lot, and traveling has become the first choice of many people's holidays, and the hotel industry closely related to tourism also ushered in a wave of dividends, has been rapidly developed. The competition among hotels is also increasingly fierce, the traditional manual operation has been unable to meet the development of the modern hotel industry, not only will bring customers bad experience, and lead to the low level of staff management. Therefore, how to provide customers with a better experience and improve the level of management has become particularly important. Under the background of the information age, this paper USES SpringBoot and Mybatis framework to design the chain hotel application system, which realizes the hotel information management, realize the main functions of hotel room reservation, user information management, hotel information management, etc. Users can search in the system to meet their needs of the hotel room type, to solve the problem of difficult to find a room, booking difficult, to achieve the new model of people at home, first booking, to eliminate confusion in the mind. System administrator can manage all kinds of information in the background, provide convenience for hotel management booking, improve work efficiency and reduce the workload of hotel staff.

**Key words:** Hotel chain system;SpringBoot;Mybatis

目录

[一、绪论 1](#_Toc78556093)

[1.1 选题背景和意义 1](#_Toc78556094)

[1.2 国内外研究和应用现状 1](#_Toc78556095)

[1.3 论文组织结构 1](#_Toc78556096)

[二、 相关技术与工具 2](#_Toc78556097)

[2.1 关键技术 2](#_Toc78556098)

[2.2 开发工具 2](#_Toc78556099)

[2.3 开发环境 2](#_Toc78556100)

[三、 可行性研究与需求分析 3](#_Toc78556101)

[3.1 可行性研究 3](#_Toc78556102)

[3.2 需求分析---（用例图） 3](#_Toc78556103)

[3.3 非功能需求 4](#_Toc78556104)

[三、 概要设计 4](#_Toc78556105)

[4.1系统架构设计---（架构图） 4](#_Toc78556106)

[4.2系统功能设计---（功能模块图） 4](#_Toc78556107)

[4.3数据库设计---（E-R图） 5](#_Toc78556108)

[四、 详细设计和实现 6](#_Toc78556109)

[5.1 用户个人模块的详细设计与实现 ---（流程图/项目截图/核心代码截图） 6](#_Toc78556110)

[5.2 商家管理模块的详细设计与实现 ---（流程图/项目截图/核心代码截图） 13](#_Toc78556111)

[5.3 管理员模块的详细设计与实现 ---（流程图/项目截图/核心代码截图） 17](#_Toc78556112)

[5.4 登录模块的详细设计与实现 23](#_Toc78556113)

[五、 总结和展望 23](#_Toc78556114)

# 一、绪论

## 选题背景和意义

随着科技和信息的发展，社会秩序的延伸，更多的公共场合、商家、客户越发存在对于秩序化的、顺序化的、可预见性的、可预定性的交互，例如在店面中点餐，往往会存在许多排队异常、顺序异常的问题，引起大家的不愉快，并且排队往往要求顾客时刻身在店面中，取决于队列前列的人的个人习惯，往往导致最终排队时间难以预见，从而增加顾客的排队顾虑和担忧，以及无谓的时间浪费。

可见市面上对于排队的优化、计划存在普遍性需求。

本项目是为了解决排队效率问题，增加顾客、商家的时间利用率和就餐环境等问题，与此同时为用户提供更加舒适的服务。可预见的是本项目会是链接顾客、商家的重要途经和排队选择。

## 国内外研究和应用现状

目前国内外市场的排队管理主流的仍然依托于低效率的当堂叫号的形式，市面上缺乏这样一种解决即时排队问题的技术解决手段，可以预见本项目的竞争空间是非常之大。

只要我们能够给出完整的项目和管理方案，那么可以快速地推进应用地域，乃至于全球推广。

## 论文组织结构

第一章，绪论。介绍论文的选题背景和意义，国内外研究与应用现状。

第二章，相关技术与工具。介绍开发此系统的相关技术与开发工具，以及开发环境。

第三章，可行性研究与需求分析。介绍系统开发可行性，以及系统所具有的功能需求和性能需求。

第四章，概要设计。介绍本系统的架构设计，系统功能设计，数据库设计，以及物理结构设计。

第五章，详细设计和实现。详细介绍本系统各个功能的类的实现和核心实现，以及各大功能模块的实现效果。

第六章，系统测试。进行系统测试工作，发现系统中潜在的漏洞，并加以说明。

第七章，总结和展望。对毕业设计的制作过程进行总结和思考，从中认识到该系统做的好的地方和做的不好的地方，并对系统做的不好的地方提出优化方案。

# 二、 相关技术与工具

## 2.1 关键技术

2.1.1 Spring Boot

2.1.2 Mybatis

2.1.3 Mybatis-plus

2.1.4 B/S架构

## 2.2 开发工具

集成开发环境：Idea

前端页面开发工具：vscode + Idea

## 2.3 开发环境

硬件：Intel Core i7-6700HQ CPU @2.60GHZ 2.59GHZ、内存容量：8GB，硬盘容量500GB

软件：操作系统:windows10 64位

开发平台:IDEA 2021

web Springboot

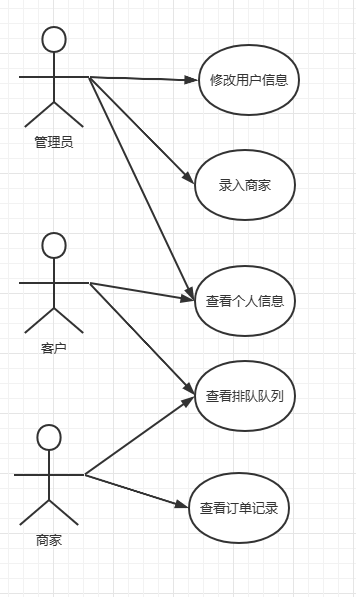
jdk:JDK 1.8

数据库:MariaDB 10.8

# 三、 可行性研究与需求分析

## 3.1 可行性研究

## 3.2 需求分析---（用例图）

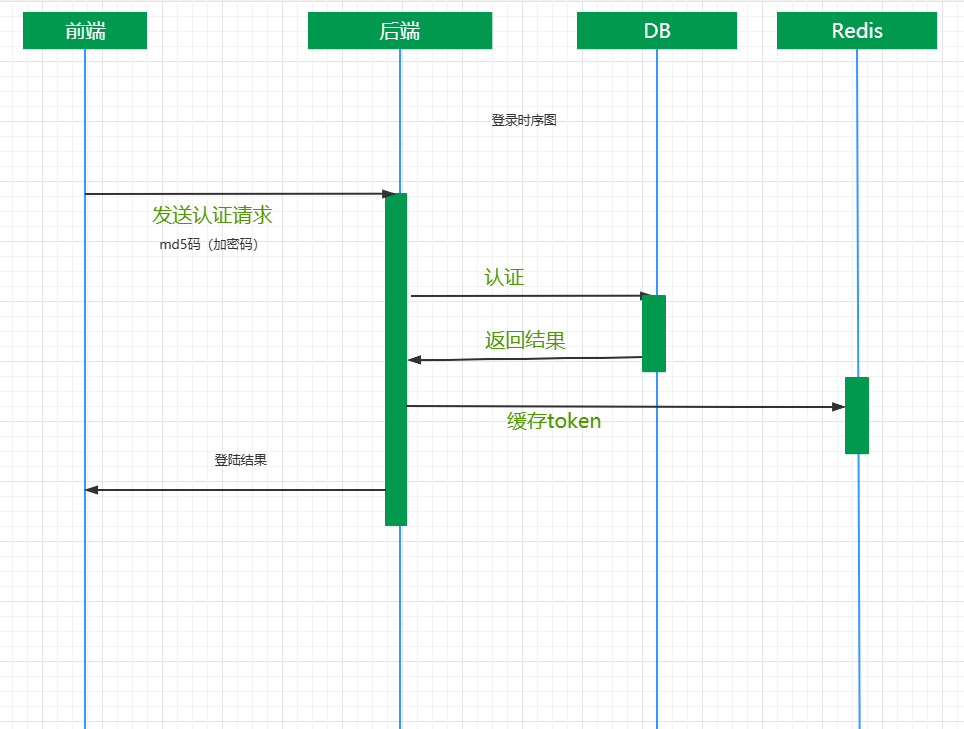


## 3.3 非功能需求

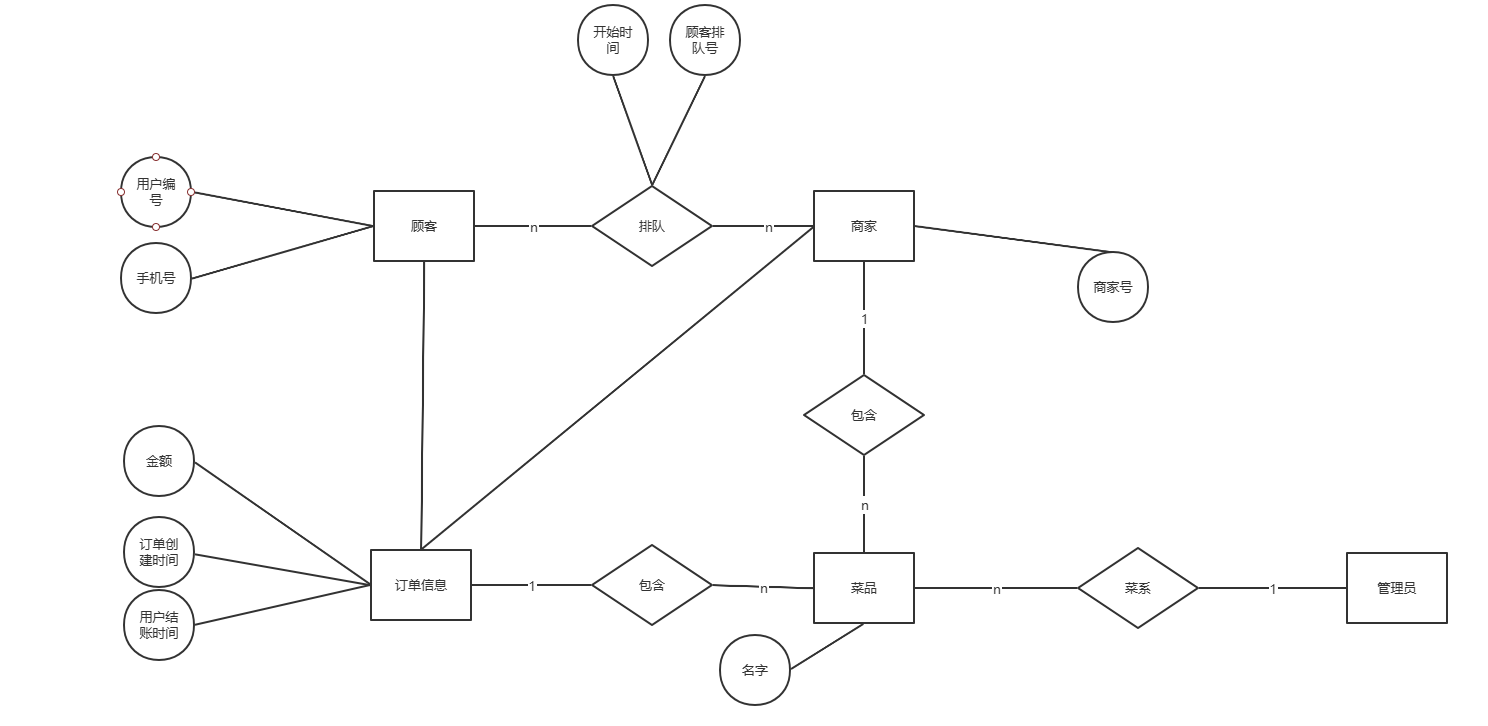
# 三、 概要设计

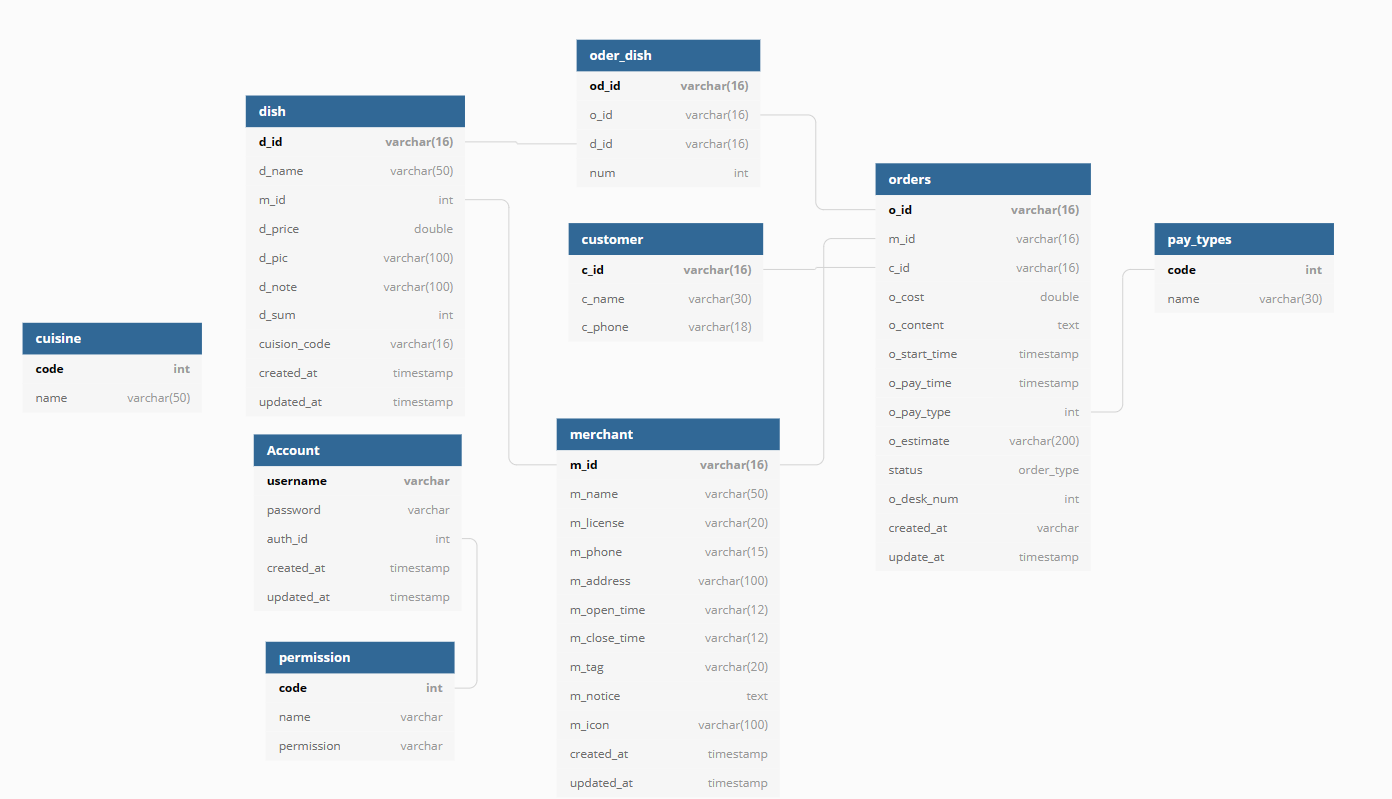
## 4.1系统架构设计---（架构图）

## 4.2系统功能设计---（功能模块图）



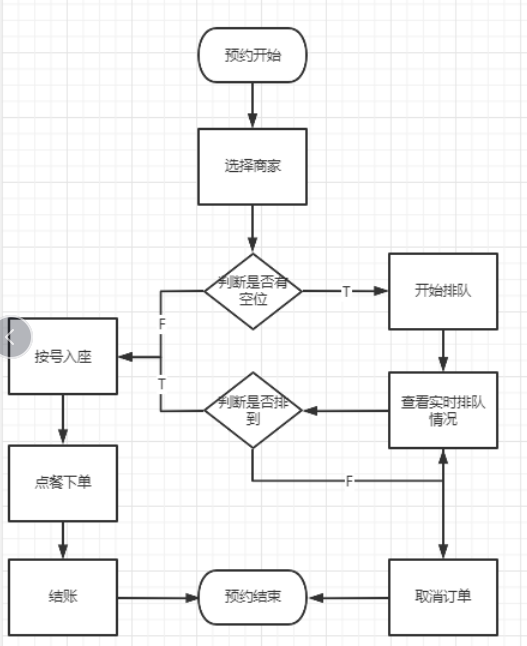
## 4.3数据库设计---（E-R图）

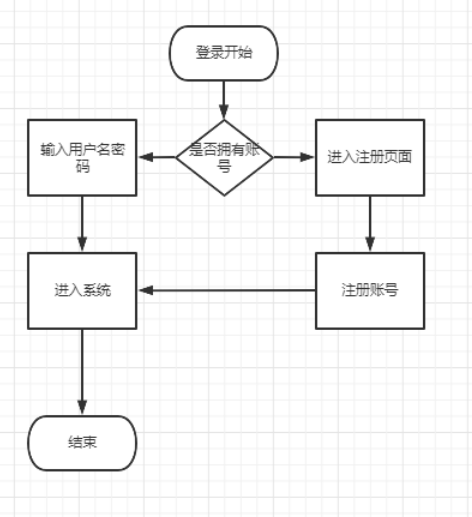


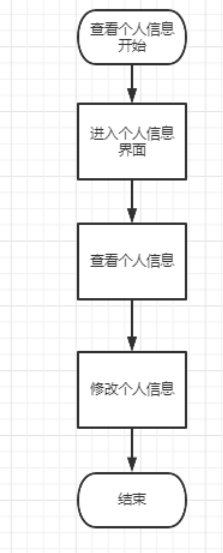


# 四、 详细设计和实现

## 5.1 用户个人模块的详细设计与实现 ---（流程图/项目截图/核心代码截图）

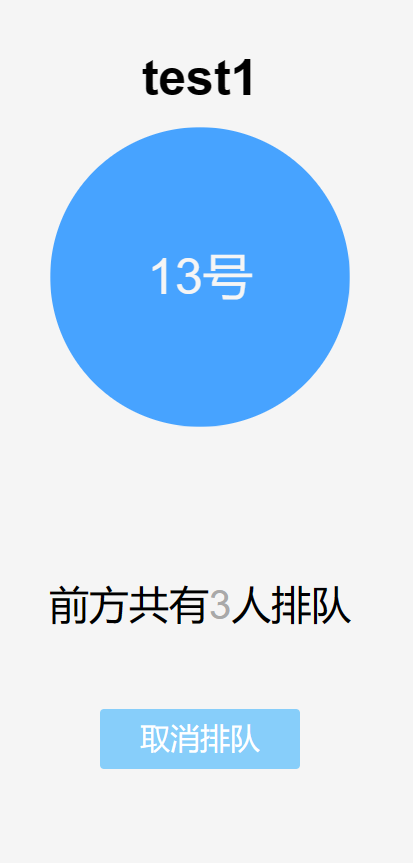


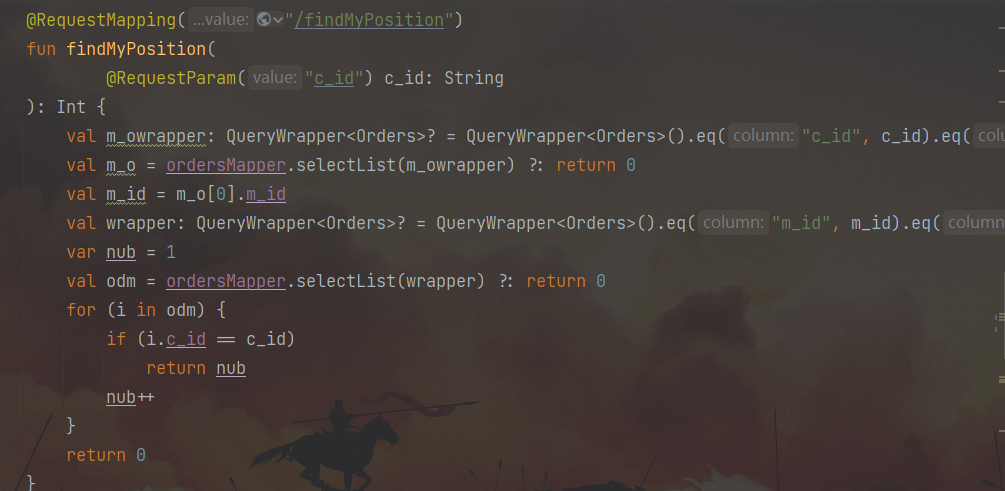






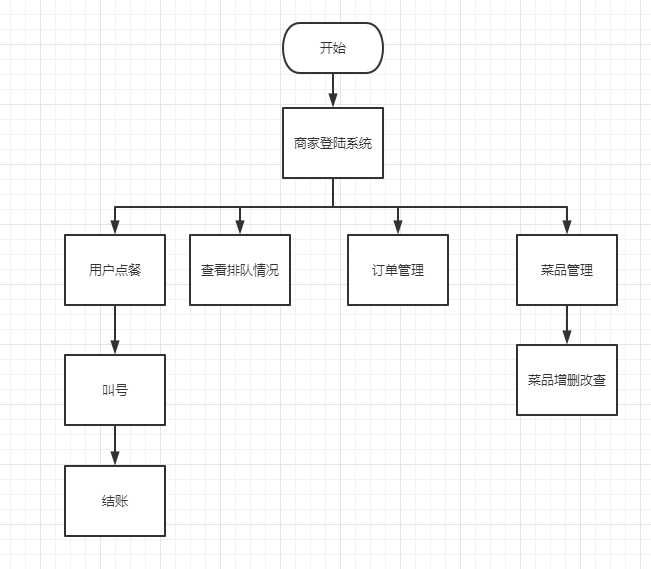






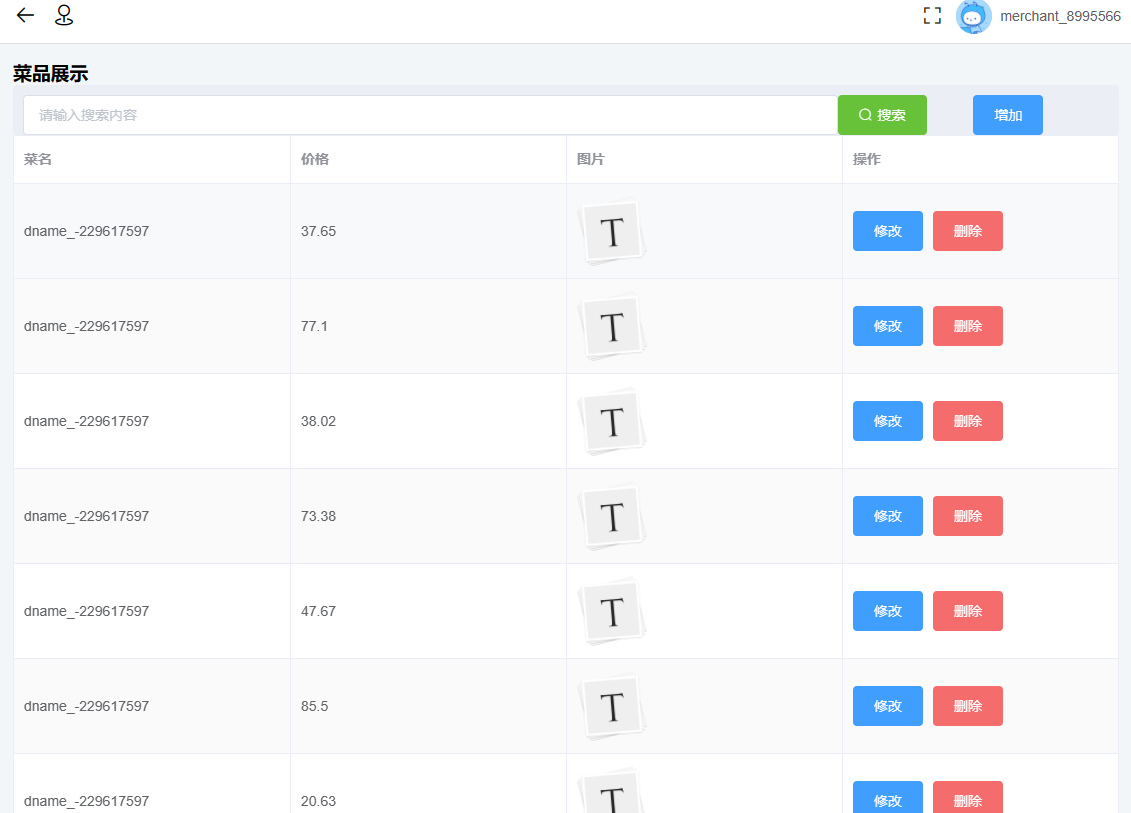


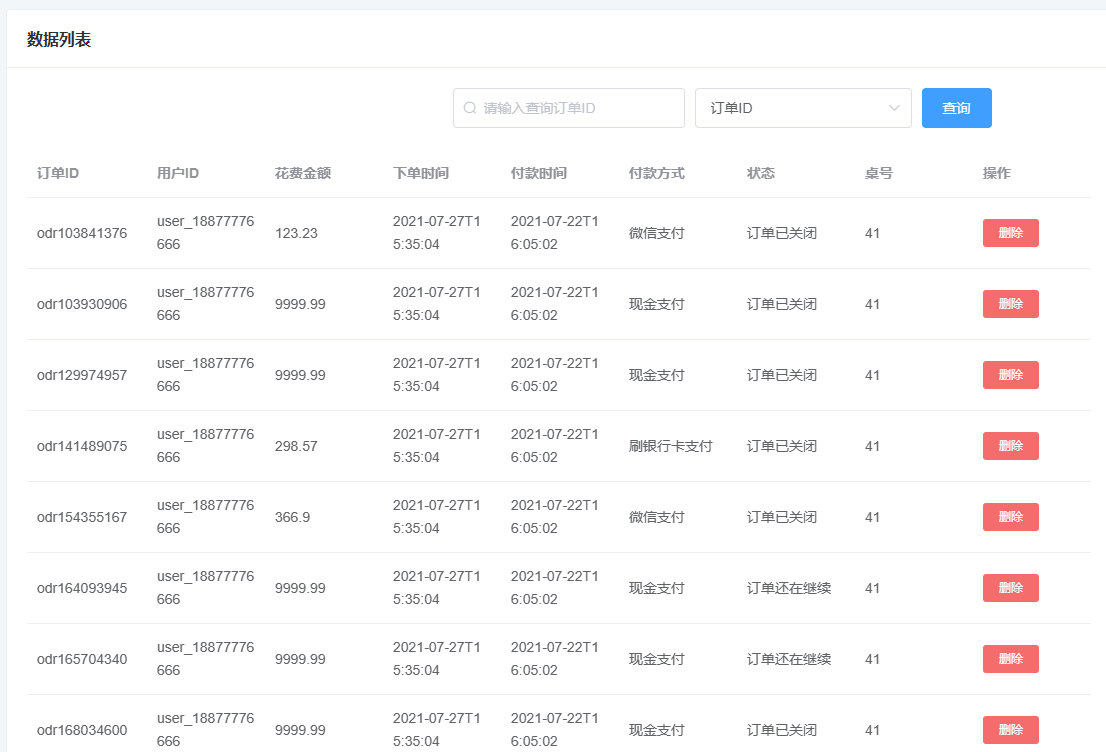
## 5.2 商家管理模块的详细设计与实现 ---（流程图/项目截图/核心代码截图）

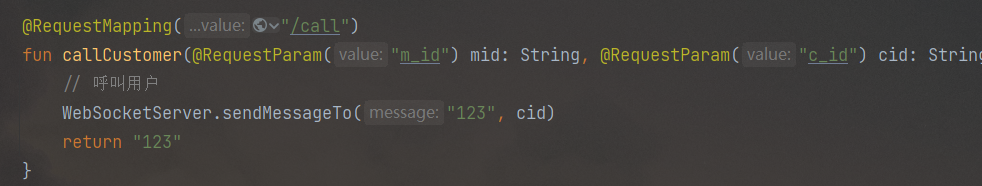


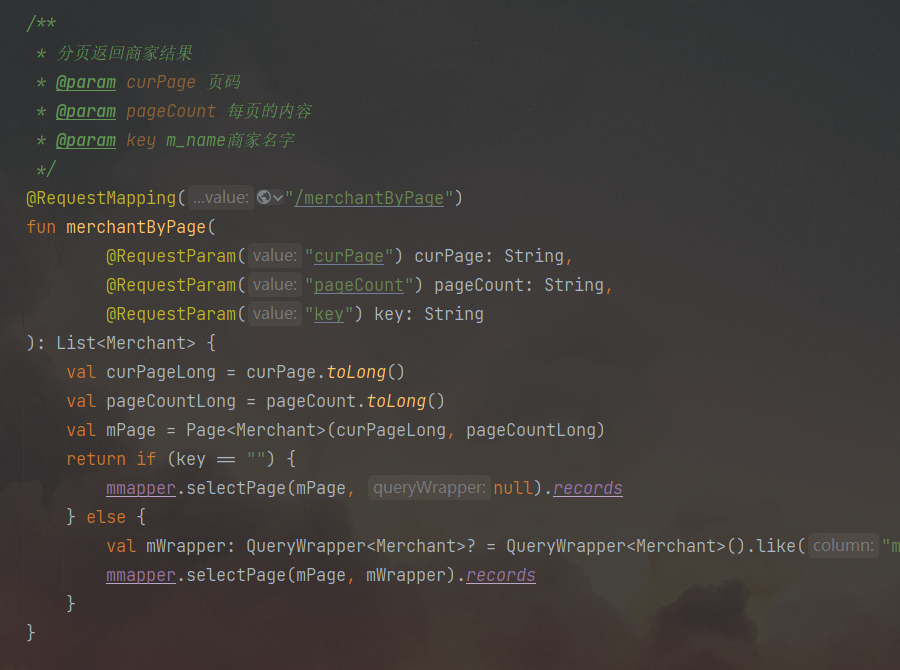






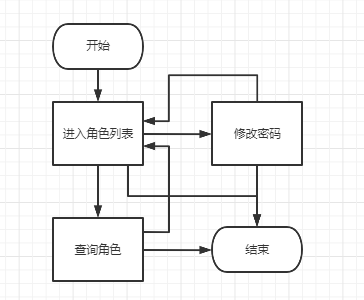


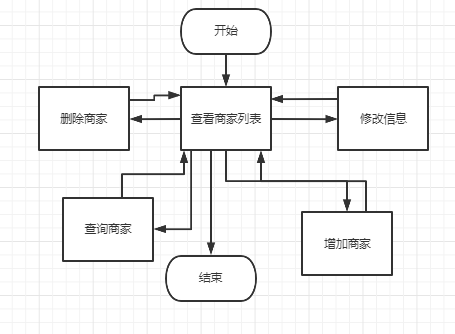


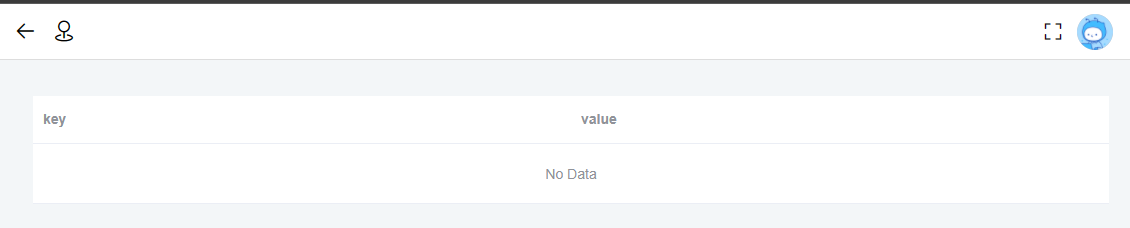


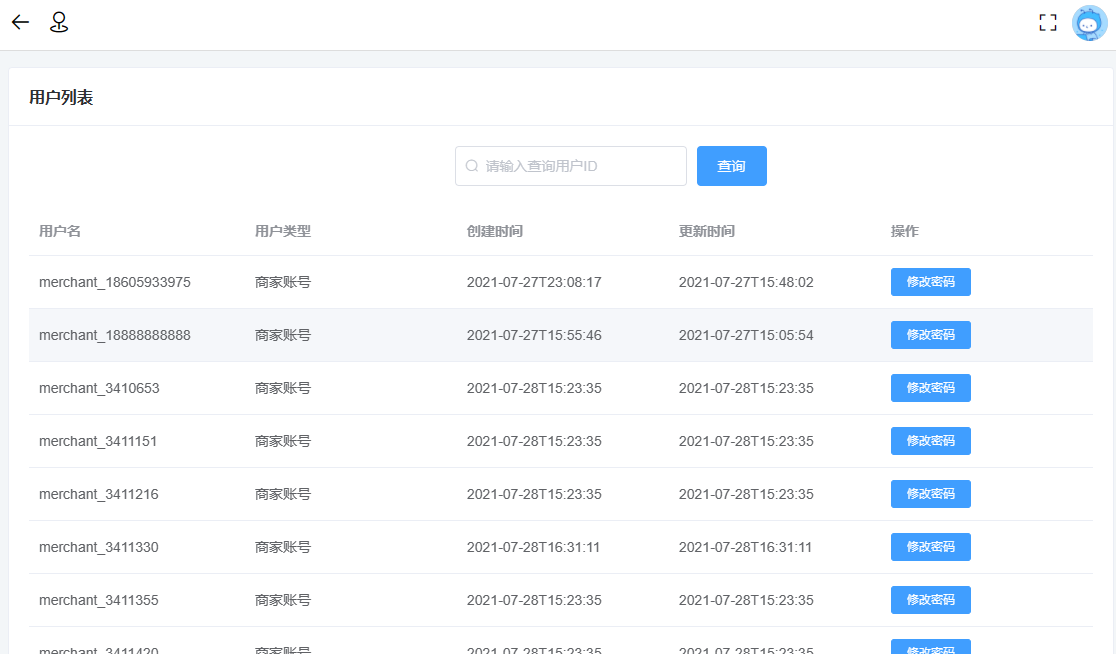


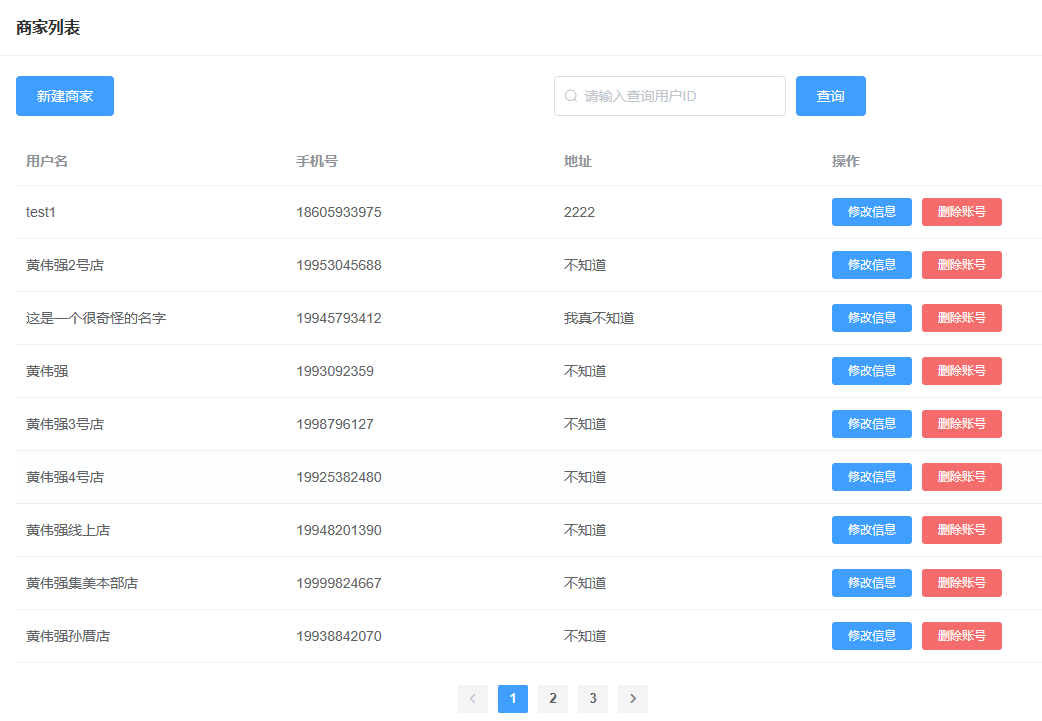
## 5.3 管理员模块的详细设计与实现 ---（流程图/项目截图/核心代码截图）













|  |
| --- |
| @Service  class RedisServiceImpl : RedisService {  companion object {  const val SEVEN\_DAY\_SEC: Long = 7 \* 24 \* 60 \* 60  }  @Resource  private val redisTemplate: RedisTemplate<String, Any>? = null  override fun set(key: String, value: Any) {  val rs = StringRedisSerializer()  redisTemplate?.keySerializer = rs  val vo = redisTemplate?.opsForValue()  vo?.set(key, value.toString(), SEVEN\_DAY\_SEC)  }  override fun delete(key: String): Boolean {  return redisTemplate?.delete(key) ?: false  }  override fun get(key: String): Any? {  val ops = redisTemplate?.opsForValue()  return ops?.get(key)  }  } |

## 5.4 登录模块的详细设计与实现

|  |
| --- |
| /\*\*  \* 执行登录操作（调用顺序应该先调用check）  \* 插入token数据并返回token-username-auth三个数据（包装为实体类）  \* @param act Account 可以只带有username和password  \*/  @RequestMapping("/login")  fun login(@RequestBody act: Account) : String {  // 查询数据库  val account = accountMapper.selectByName(act.username)  // 验证密码  if(account != null && BCryptPasswordEncoder().matches(act.password, account.password)) {  // 验证成功  // 生成token  val token = LoginUtil.generateToken(account.username)  // 生成权限字符串  val auth: String = LoginUtil.generateAuth(account.auth)  // 此处应该插入数据库，保存token时间  // 生成成功信息  val res = success(ResponseAccount(token, act.username, auth))  return JSON.toJSONString(res)  }  // 密码错误，返回错误信息  return JSON.toJSONString(fail(ResultCode.USER\_CREDENTIALS\_ERROR))  } |

## 5.5 大数据模块的详细设计与实现

|  |
| --- |
| /\*\*  \* 写入新的数据  \*/  fun writeInFile(uid: String, search: String, index: Int, mid: String ) {  // 获得class运行路径  val path: String = getPath()  // 打开存储路径  val fileUploadDir = File("$path/log")  // println(fileUploadDir.path)  if( !fileUploadDir.exists()) {  fileUploadDir.mkdirs()  }  // 查看昨天的文件是否还在，还在的话就上传并删除掉  var fn = getFileName()  var file = File(fileUploadDir, fn)  // println(file.exists())  if( file.exists() && file.isFile ) {  uploadFile(file)  file.delete()  }  // 写入今天的文件  fn = getTodayFilename()  file = File(fileUploadDir, fn)  if( !file.exists()) {  file.createNewFile()  }  val bw = BufferedWriter(FileWriter(file))  val arr = "$uid $search $index $mid"  bw.write(arr+"\n")  println("\n\npath: ${file.path}\n\n")  println("\n\nwrite: $arr\n\n")  // 关闭  bw.close()  } |

|  |
| --- |
| /\*\*  \* 次数序号RDD执行业务类的执行函数  \* @param filename 文件名，hdfs文件请使用hdfs链接形式  \* @return 序号-次数键值对  \*/  fun run(filename: String) : Map<String, Int> {  // val map = HashMap<String, Int>()  // 读取文件  val lines : JavaRDD<String> = javaSparkContext.textFile(filename).cache()  // 根据第三位字符串（序号字符串），进行map  val lineWords : JavaPairRDD<String, Int> = lines.mapToPair { Tuple2(pattern.split(it)[3], 1) }  // 合并  val numCount = lineWords.reduceByKey { v1, v2 -> v1 + v2 }  // 返回结果  return numCount.collectAsMap()  } |

|  |
| --- |
| @RestController  @RequestMapping("/api/spark")  class SparkController {  /\*\*  \* 显示大数据内容  \*/  @RequestMapping("/bigdata")  fun getLastDayBigData(@RequestParam("index") i: Int ) : Map<String, Int> {  return when(i) {  1 -> MySpark.getNumberWithCount()  2 -> MySpark.getMerchantVisited()  else -> MySpark.getSearchWordCount()  }  }  @RequestMapping("/writer")  fun writeRecords(@RequestParam("c\_id") cid: String,  @RequestParam("search") search: String,  @RequestParam("index") idx: Int,  @RequestParam("m\_id") mid: String  ) : String {  MySpark.writeInFile(cid, search, idx, mid)  return "OK"  }  } |

# 六、 系统测试

## 6.1 测试用例

表6.1 房型预定的测试用例及结果

|  |  |  |  |
| --- | --- | --- | --- |
| 用例 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 点击预计到店日期按钮 | 日期选择框弹出 | 结果符合预期 |
| 2 | 点击预计离店日期按钮 | 日期选择框弹出 | 结果符合预期 |
| 3 | 点击搜索框输入条件进行搜索 | 页面局部刷新显示酒店列表 | 结果符合预期 |
| 4 | 点击已被订满房型按钮 | 无法点击 | 结果符合预期 |
| 5 | 在未登录情况下进入房型详情页面 | 提示用户并跳转至登录界面 | 结果符合预期 |
| 6 | 在支付界面停留长时间不进行支付 | 交易失败 | 结果符合预期 |

表6.2 个人中心模块的测试用例及结果

|  |  |  |  |
| --- | --- | --- | --- |
| 用例 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 点击刷新按钮 | 订单列表刷新 | 结果符合预期 |
| 2 | 选择订单点击取消订单按钮 | 订单状态改变 | 结果符合预期 |
| 3 | 选择已退订状态的订单进行退订 | 提示用户选择有效信息 | 结果符合预期 |
| 4 | 修改用户密码 | 修改成功 | 结果符合预期 |
| 5 | 点击下一页内容 | 显示下一页内容 | 结果符合预期 |
| 6 | 在修改密码时将密码改为空 | 提示输入有效信息 | 结果符合预期 |

表6.3 会员模块的测试用例及结果

|  |  |  |  |
| --- | --- | --- | --- |
| 用例 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 选择会员等级进行升级 | 进入支付界面 | 结果符合预期 |
| 2 | 完成支付 | 会员等级升级成功 | 结果符合预期 |
| 3 | 点击当前会员等级进行升级 | 提示先前已完成升级 | 结果符合预期 |
| 4 | 点击更低会员等级进行升级 | 提示先前已完成升级 | 结果符合预期 |

表6.4 会员管理模块的测试用例及结果

|  |  |  |  |
| --- | --- | --- | --- |
| 用例 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 点击刷新按钮 | 会员列表刷新 | 结果符合预期 |
| 2 | 在搜索框输入手机号进行搜索 | 显示符合手机号会员列表 | 结果符合预期 |
| 3 | 点击新增按钮 | 弹出新增会员弹窗 | 结果符合预期 |
| 4 | 未选择会员信息点击修改按钮 | 提示请选择有效数据 | 结果符合预期 |
| 5 | 未选择会员信息点击删除按钮 | 提示请选择有效数据 | 结果符合预期 |
| 6 | 在修改会员信息时将条目改为空 | 提示输入有效信息 | 结果符合预期 |

表6.5 房型管理的测试用例及结果

|  |  |  |  |
| --- | --- | --- | --- |
| 用例 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 选择房型图片上传 | 图片预览成功 | 结果符合预期 |
| 2 | 修改房型图片 | 图片修改成功 | 结果符合预期 |

表6.6 订单管理的测试用例及结果

|  |  |  |  |
| --- | --- | --- | --- |
| 用例 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 选择订单进行入住办理 | 订单状态改为已入住 | 结果符合预期 |
| 2 | 选择订单进行退房办理 | 订单状态改为已退房 | 结果符合预期 |
| 3 | 选择订单进行换房 | 换房成功 | 结果符合预期 |
| 4 | 选择订单进行退款办理 | 退款办理成功 | 结果符合预期 |
| 5 | 进行订单搜索 | 显示符合条件订单列表 | 结果符合预期 |
| 6 | 选择不符合状态的订单进行操作 | 提示用户选择有效信息 | 结果符合预期 |

表6.7 权限校验的测试用例及结果

|  |  |  |  |
| --- | --- | --- | --- |
| 用例 | 输入/动作 | 预期结果 | 测试结果 |
| 1 | 非超级管理员进入酒店管理界面 | 提示并跳转至会员管理界面 | 结果符合预期 |
| 2 | 非超级管理员进入会员机制界面 | 提示并跳转至会员管理界面 | 结果符合预期 |

## 6.2 测试方法

#### 6.2.1集成测试

集成测试是将各个模块集成为系统进行测试。系统中某些模块单独运行时没有问题，但把他们集成起来一起运行就很有可能有问题暴露。从系统的最小模块进行测试，然后逐层往上测试，以保证系统可以正常运行。

#### 6.2.2黑盒测试

黑盒测试又称为功能测试。测试通过输入的数据看输出的结果，从而了解系统的工作。

## 6.3 测试结果

通过测试的测试用例，测试的结果与预期基本吻合，说明系统的基本功能都满足预期的设定。

# 五、 总结和展望

**总结：**

本文所设计并实现的即使排队系统是参考了国内外的即使排队系统，和结合自身生活经历的实际需求后进行的。在整个设计和开发的过程自己感觉受益良多。以下是本系统在设计与开发过程中的成果：

1. 透过前期大量的文献资料查询和对即使排队系统的参考，理清了系统所应具备的功能和各个功能之间的衔接工作，前期的准备为后来的开发打下了坚实的基础。

2. 谨记软件工程的思想，秉承“高内聚，低耦合”的设计原则，明确了系统开发的所要达到的效果。对系统各个功能的操作流程图进行了详细设计，明确了系统开发的步骤。

3. Ajax、SpringBoot、MyBatis、Redis技术和MySql数据库，阿里云短信、支付宝沙箱以及三层架构等，为系统开发提供了坚实的技术保障。

4. 系统设计完成后，透过大量的测试，不断改进，提高了酒店管理的效率，完善了用户的体验。

**展望：**

由于自身开发经验的不足，系统还有许多需要改进的地方，如下：

1. 项目分析阶段应尽可能预先想到系统可能存在的问题和不足，这样才能避免开发进行到一半时又回过头来改进之前的开发，造成无效开发和时间的浪费。

2. 整个系统的开发过程中以实现功能为紧要目标，并没有考虑代码冗余的情况，以及代码执行效率。

3. 当今互联网世界，网络安全问题越来越得到人们的重视，本系统有存在安全隐患的风险，因在系统安全性方面没有深入研究，需要日后不断升级改进。