

山东工商学院

本科毕业论文(设计)



题 目：易生活服务系统

姓 名：姜山

学 院：计算机科学与技术学院

专 业：软件工程专业

班 级：2019 级 1 班

学 号：2019214251

指导教师：

职称：

2023 年 5 月 15 日

易生活服务系统

E-life Ticket Manager

姜山

Jiang Shan

2023 年 5 月 15 日

诚信声明

本人郑重声明：所呈交论文，是在导师指导下独立进行研究所取得的研究成果。论文除文中已经注明引用的内容外，不包含任何其他集体或个人已经发表或在网上发表的内容。

特此声明。

声明人：

年 月 日

指导教师意见

指导教师姓名：		职称：	
指 导 参 考 项 目			
论文选题	1	符合专业培养目标	
	2	具有学术价值	
	3	具有新颖性	
	4	难易程度适中	
文献资料	5	文献收集系统完备	
	6	文献理解准确得当	
	7	文献征引合乎规范	
	8	外文翻译准确通顺	
	9	观点归纳完整清晰	
论文写作	10	文题相符，立论正确	
	11	数据准确，逻辑严谨	
	12	条理清晰，论证充分	
	13	行文流畅，格式规范	
	14	篇幅适中，按期完成	
论文成果	15	文献综述完整，研究基础厚实	
	16	具有理论意义或实际价值	
	17	作者见解独到新颖	
	18	具有拓展、延伸性	
写作态度	19	积极同指导教师沟通	
指导成绩			
是否同意答辩（“□”内划“√”）		是□	否□
指导教师意见	指导教师签字： 年 月 日		

评阅人意见与成绩评定

评阅人姓名：		职称：	
评 价 项 目			
论 文 选 题 (10 分)	1	符合专业培养目标	
	2	具有学术价值	
	3	具有新颖性	
	4	难易程度适中	
文 献 资 料 (25 分)	5	文献收集系统完备	
	6	文献理解准确得当	
	7	文献征引合乎规范	
	8	外文翻译准确通顺	
	9	观点归纳完整清晰	
论 文 写 作 (40 分)	10	文题相符，立论正确	
	11	数据准确，逻辑严谨	
	12	条理清晰，论证充分	
	13	行文流畅，格式规范	
	14	篇幅适中，按期完成	
论 文 成 果 (25 分)	15	文献综述完整，研究基础厚实	
	16	具有理论意义或实际价值	
	17	作者见解独到新颖	
	18	具有拓展、延伸性	
评阅人最终给分			
评 阅 人 评 语	<div>评阅人签字： 年 月 日</div>		

答辩（评审）委员会意见

成绩 _____

鉴定意见：

主任（签章） _____

年 月 日

易生活服务系统

摘 要

在生产实践中，客户服务大多采用一对一的模式。但是对于小微企业以及只有数人的小型工作室而言，人力资源仍然是最重要的资源。

基于此，本文采用了一种基于服务单的客户服务模式，使用基于 MIT 协议开源的若依框架进行二次开发，实现了包括用户端注册、用户端登录、重置密码、服务单转发、搜索客服评价、查看评价详情、查看综合评价、添加问题类型、删除问题类型、搜索问题类型、修改问题类型、搜索投诉、删除投诉、查看投诉、搜索入驻申请、同意入驻申请、驳回入驻申请、搜索客服、添加客服、注销客服、企业管理、申请服务单、关闭服务单、重开服务单、删除服务单、发送消息、服务评价、投诉服务、查看日志、添加员工角色、删除员工角色、修改员工角色、搜索员工角色、赋予员工角色、撤销员工角色、自定义回复模板、搜索回复模板、修改回复模板、删除回复模板在内的功能。

经过一年的实践证明，这种模式可以支持一个不足百人的客户服务小组满足一万余名客户的服务需求，取得了巨大的成功。事实证明，该模式不仅可以提高小微企业的资源利用效率，还能够在一定程度上支持大型企业以及外包性质的客户服务。

关键词： 易生活服务系统；服务单；客户服务；服务模式；小微企业

E-life Ticket Manager

Abstract

In production practice, customer service mostly adopts a one-to-one method. However, for enterprises and studios with only a few employees, human resources are still the most important resource.

To solve this problem, we applied a customer service method based on tickets. Using Ruoyi-Vue, which licensed under MIT license. System has many functions includes registration, login, password reset, service ticket forwarding, search Customer service evaluation, view evaluation details, view comprehensive evaluation, add ticket type, delete ticket type, search for ticket type, edit ticket type, search for complaints, delete complaints, view complaints, search settlement approves, pass settlement, reject settlement, search for employee , add employee, cancel employee, enterprise management, apply ticket, close ticket, reopen ticket, delete ticket, send message, service evaluation, service complaint, view system log, add employee role, delete employee role, modify employee role , search for employee roles, grant employee roles, revoke employee roles, customize reply templates, search for reply templates, modify reply templates, and delete reply templates.

During a yearlong period, a team that less than 100 people have met the service needs of more than 10,000 customers by using this method and has achieved great success. Facts have proved that this method can not only improve the resource utilization efficiency of company that have limited human resource, but also can support large enterprises and outsourced customer service.

Keywords: E-life Ticket Manager; Ticket; micro company; Ruoyi-Vue

目 录

第一章 绪论.....	2
1.1 项目背景和意义.....	2
1.1.1 项目背景.....	2
1.1.2 项目意义.....	2
1.2 服务单模式.....	3
1.2.1 服务单.....	3
1.2.2 服务流程.....	4
1.3 项目中涉及的开源项目和其他依赖.....	5
1.3.1 MyBatis	5
1.3.2 RuoYi-Vue.....	5
第二章 需求分析.....	6
2.1 总体描述.....	6
2.1.1 产品概述.....	6
2.1.2 系统目标.....	6
2.1.3 定义及缩写词.....	7
2.1.4 产品功能.....	8
2.1.5 用户特征.....	9
2.2 具体需求.....	9
2.2.1 企业入驻申请.....	9
2.2.2 申请服务单.....	11
2.3 功能需求.....	12
2.3.1 用例列表.....	15
2.3.2 用例表.....	17
2.4 数据库需求.....	19
2.5 系统的质量属性.....	20
2.5.1 可靠性.....	20
2.5.2 可获得性.....	20
2.5.3 保密性.....	20
2.5.4 可维护性.....	21
2.6 需求模型.....	21
2.6.1 静态模型.....	21
2.6.2 动态模型.....	22
第三章 体系结构设计.....	26
3.1 部署视图.....	26
3.1.1 概述.....	26
3.1.2 部署政策.....	26
3.2 逻辑视图.....	26
3.2.1 概述.....	27
3.2.2 包或子系统.....	27
3.3 进程视图.....	30
3.3.1 概述.....	30
3.3.2 过程间通信机制.....	31

3.4 Use-Case 视图.....	31
3.5 实现视图.....	33
3.5.1 业务流程层.....	33
3.5.2 业务逻辑层.....	40
3.5.3 数据访问层.....	49
3.5.4 领域模型层.....	59
3.6 数据视图.....	59
3.6.1 逻辑设计.....	59
3.6.2 数据访问机制.....	61
第四章 详细设计.....	62
4.1 用户交互层.....	62
4.1.1 企业入驻审批.....	62
4.1.2 申请服务单.....	65
第五章 软件测试.....	68
5.1 测试对象和概要.....	68
5.1.1 测试方法与工具.....	68
5.1.2 测试环境与配置.....	69
5.1.3 单元测试用例设计.....	69
5.1.4 功能测试用例设计.....	71
5.1.5 性能测试用例设计.....	76
5.1.6 整体策略.....	76
5.1.7 测试范围.....	77
5.2 测试内容和执行情况.....	80
5.2.1 功能测试.....	80
5.2.2 性能测试.....	80
5.3 覆盖分析.....	81
5.4 缺陷统计与分析.....	81
5.5 测试结论.....	81
参考文献	82
致谢	83

第一章 绪论

1.1 项目背景和意义

1.1.1 项目背景

在生产实践中，大部分企业为客户提供服务皆通过常规的多对一的模式进行——为每一个客户安排一位客户服务人员，每个客户服务人员对多名客户提供服务。在此模式下，对于单个客户，服务的时间窗口不可避免地会被一定程度压缩，而且转发其他部门时也要考虑其当前是否有人在线。一旦在短时间内大量人员提出请求或用户请求出现突发现象，就可能造成单个客户服务人员工作量剧增，在一定程度上降低客户服务的质量。并且，由于客户的个性化需求越来越多，现有的体系其实并不能迅速响应客户的需求^[8]。

目前，许多企业提高服务质量的解决方案仍是扩大客户服务团队的规模；而对于许多小微企业，尤其是仅由几人组成的小型工作室以及个体工商户而言，人力资源和技术资源都占有举足轻重的地位。由于没有专门的客户服务系统，他们通常采用传统的聊天软件（如 QQ、微信）来进行客户服务。在这种情况下，经常出现由于用户急于获取服务，向每个可以与用户对接的客户服务人员都提出服务请求的情况。这种行为产生了大量的资源浪费并在极大程度上降低了，甚至毁灭了这些小微企业的客户服务质量。此外，常常由于员工难以及时对客户的请求做出回应，使得客户服务部门的员工需要承受更多来自用户的谩骂和侮辱；或在耐心解释后得到用户“已经解决”的回复。这些情况使得员工在工作中难以获得成就感，在极大程度上消磨了员工的精神、降低了员工积极性和自我认同感，在恶性循环中降低了服务质量。因此，一种能够替代传统聊天软件的客户服务系统逐渐成为了一种迫切的需求。

1.1.2 项目意义

由此，开发一种能够解决中小型企业 and 小型工作室的客户服务问题的系统，不仅能够整合大量的小微企业、小型工作室甚至个体工商户的需求，降低整个系统的运营成本；并且能够在一定程度上解决当前小微企业的困境，解放生产力，优化资源安排、促进产业发展。未来能够按照需求整合电商平台、直播平台或其他功能，系统可以进一步统合资源，提高工作效率。近年来，大部分的相关研究都指向了智能化方向^{[6][7][9]}。面对专业知识不足的经营实体，更多采用的对策是透明与封装，使得整个系统对于用户来说难以理解。本文采用了基于服务单的管理模式，希望能够引入一种新的思路，起到抛砖引玉的效果。

除此之外，本项目从服务模式出发，充分提高新员工在工作中的参与感，平衡挑战和成就，带来积极的愉悦和满足，从而使其依靠内在动力去做自己想做的事情，在学习与工作中不断探索、挑战，寻求新的目标而不知疲倦^[5]。

1.2 服务单模式

易生活服务系统（下称：本系统）统筹各项资源的核心方法是一种经过重新架构的模式。这种模式以一种被称为“服务单”的概念为最小单位，我们将其称为基于服务单的客户服务模式，简称服务单模式。该模式不仅可以提高客户服务人员的服务质量，也可以提高用户体验。最重要的是，该模式能够减少占用企业的人力资源。

经过一年的实践，截至 2023 年 5 月 1 日，通过使用这种模式，一个不到 100 人的团队在一个总人数 15827 人的社区中成功提供了总计 20064 次客户服务，基本能够达到及时响应、高效沟通的要求，得到了业务团队和社区成员的极佳评价。

在下一节中，本文将详细描述服务单的概念。

1.2.1 服务单

服务单是本系统的核心，也是我们提出的一种全新的用于衡量客户服务工作的单位；在国际化界面中称为 Ticket。由于这种概念对用户不够友好，在实际的操作界面中往往将其封装成易于用户理解的友好界面。相比现有的工单系统，服务单面向某个部门的全体客户服务人员而不指定某个具体负责的员工。

服务单是由企业产品的用户提出的一个或多个服务申请组成的、由企业客服人员与其进行的一次或数次会话；服务单也是企业产品用户阐明自己的需求，企业客服人员对其进行服务的依据。在本模式中，服务单将作为处理客户请求的单位。

服务单可以是独立的，也可以是一组服务申请的一部分。可以为一张服务单定义子单：如果客户提出的申请过长、过多或过于繁琐，难以在一次会话中系统、简明地阐述，就需要定义一组服务单。一般来说，只要客户提出的需求可以在一张服务单中阐明，并且双方能够达成一致，则不应为用户发放子服务单。由多个服务申请组成的服务单仅在用户的请求过于繁琐、冗长且难以用篇幅较短的文字说明时才应出现。

在线上的服务单管理系统中，由类似聊天群组的形式对服务单进行管理。定义一个只能由指定客服人员和客户可见的聊天频道，我们可以将其视作一张线上的服务单。对于这种服务单来说，可以定义自一个频道创建开始到频道被关闭为止，作为一张服务单的生命周期。

一般来说，服务单具有以下几种状态：

1. 申请并提交

用户为了从企业处获得期望的服务，向企业申请一张服务单。这张服务单可以是单独的，也可以是一张服务单的子服务单。

一般来说，企业应在接到申请后向用户发放服务单。在本系统中，企业可以通过设置单个用户最大服务单申请上限，来避免某些用户恶意申请多张服务单、进行信息轰炸等具

有潜在危害的行为。这个上限一般由企业自定义，若企业未明确定义或尚未定义，则数量会被默认为两张。

用户在申请服务单后应当填写相关条目，包括期望获得的服务，现在的状况等；并在填写完成后提交到企业，以使企业进行有针对性地安排专业人员进行服务。

2. 关闭

当用户提交服务单时意愿进行服务的问题已经被解决，或企业与用户双方达成一致后，用户不再需求企业的服务。此时可以关闭服务单，减少企业客户服务人员在工作时的检索压力，释放一定的资源，并将服务单进行归档整理。

关闭服务单的操作可以由用户或企业员工执行，在关闭服务单后用户可以对本次服务进行评价。服务单关闭后，用户将不再能够检索到服务单；但是企业员工可以在对应的界面中检索已经关闭的服务单。

如果企业发现问题尚未被解决或需要后续通知用户，服务单可以被再次开启。再次开启的服务单将视同已经提交的服务单，用户能够正常检索并在服务单中发送消息，与客户服务人员进行沟通，并就重新发现的问题与企业达成一致。

3. 删除

如果已经归档的服务单数量过多，难以检索，企业员工可以选择将服务单删除来释放一些资源。一般来说，已经删除的服务单不能够再被检索到；除非用户或企业员工有备份服务单，他们不能够再获取已经删除的服务单的信息。

1.2.2 服务流程

本段将用故事的方式讲述在本模式中一次典型的客户服务流程：

我们假设用户“小光”在“海德林公司”购买了一台仍在保修期内的笔记本电脑。在正常使用过程中，电脑的硬盘出现了故障，不能继续读取数据。按照保修服务的规则，他可以联系公司的客户服务人员将故障件寄回，并换取一件良好的硬盘。

此时，小光需要登录本系统，在“海德林公司”的页面中点击“申请服务”的按钮。此时，系统会自动检查小光是否符合申请条件，并创建一个仅小光和公司客户服务人员能够看到的聊天室。在双方的系统中，这个聊天室被命名为“笔记本电脑硬盘故障”。

由于这个聊天室能够被任何属于海德林公司的客户服务人员检索到，很快一名经过良好培训的客户服务人员“μ”就看到了这个聊天室。μ查看了聊天室的标题，发现这是一个硬件方面的问题，不属于他的工作范围，于是他联系了属于公司硬件售后服务的部门，并且在聊天室中回复了小光，告知他问题已经收到；并且在等待硬件售后服务部门的员工回复时，μ和小光进行了充分友善的交流，使问题被详细地描述并暴露给各个员工。

由于μ联络了硬件服务部门，他们能够在系统中很快发现这个问题。部门下属的员工“小洛”在完成了对上一个用户的服务后立刻注意到了这个聊天室中有需要解决的问题。

得益于 μ 已经和小光聊了不少，他仅仅浏览了一些聊天记录，就能够判断出小光的需求，并且根据公司规定的服务流程，将公司提前准备好的申请表格发送给了小光。在填写完所有的材料之后，线上的服务部分就结束了。接下来，只需要等待小光将快递单号发送给客户服务人员，并完成线下的资源交换。

1.3 项目中涉及的开源项目和其他依赖

1.3.1 MyBatis

MyBatis 是一款在 Java 项目中被广泛应用的优秀持久层框架，它不仅支持自定义 SQL，还免除了几乎所有的 JDBC 代码、参数配置和结果集的获取。该框架可以通过简单的 XML 文件或 Java 注解来配置并映射原始类型、接口为数据库中的记录^[1]。

1.3.2 RuoYi-Vue

本项目基于开源项目 RuoYi-Vue 进行二次开发。RuoYi-Vue 是基于 SpringBoot、Spring Security、Jwt、Vue 的前后端分离的后台管理系统^[3]；RuoYi-Vue 是一个 Java EE 企业级快速开发平台，基于经典技术组合（Spring Boot、Spring Security、MyBatis、Jwt、Vue），内置大量实用模块，并且支持集群、多数据源和分布式事务^[4]。是一个优秀的快速开发框架。

该项目遵从 MIT 协议开源，非常适合快速构建功能强大的项目。目前，该项目在 Gitee 已获得了 26813 枚 Star（数据统计自 2023.05.09）。MIT 协议规定：只要包含著作权声明和本许可声明，任何人皆可免费获得本软件和相关文档文件（“软件”）副本的许可，不受限制地处理本软件^[2]。MIT 协议宽松的条件，使其成为了开源软件中备受欢迎的一种许可。

第二章 需求分析

2.1 总体描述

2.1.1 产品概述

在本产品中，系统由用户端和服务端两个端构成，根据不同的用户角色构建不同的页面，使用权限控制来控制不同用户的不同访问范围。

1. 用户端

用户端主要负责显示界面的构建与渲染，接收服务端转发的数据，经过一定处理逻辑后将其显示出来。此外，对不同用户的不同角色会提供以下服务：

用户可以通过本产品对合作企业提出售后服务申请，并与企业的客服人员进行沟通交流、对不满意的企业进行投诉；并且能够查询并保存已有的服务记录。在经过用户与企业的双向认证之后，用户可以成为某个企业的员工。此时，用户的身份会转变成员工，拥有对应公司的服务单查看权限。

使用本软件的用户可以对运营公司提出申请，认证成为企业官方账号。经过认证的企业账号将拥有自己的企业频道，拥有企业频道中的所有权限。一般来说，一个企业频道将由多个分类组成，每个分类由多个聊天频道组成，并且拥有独立的权限设置。使用本软件的企业在申请入驻平台并通过审核后，可以认证某个用户，使其成为属于本公司的员工；能够更改员工所属部门、管理企业内部的员工数据和信息可见度；能够安排客服人员与客户沟通、对用户的请求进行回复、分类归档，并且能够查询以前的服务记录和员工或用户在服务过程中对系统执行的各种操作。此外，企业还能够收到用户对企业的投诉，并可以对各个客服部门的情况进行可视化分析。

开发人员和运维人员可以在登录后查看系统的运行日志、错误信息以及其他用户的操作记录。

2. 服务端

服务端主要进行消息的转发，在用户在登录后，根据用户的权限，所属部门以及使用的功能，为用户提供对应的数据。

2.1.2 系统目标

能够存储一定数量的服务单信息，并能够有效地对服务单数据进行操作和管理。

能够存储一定量使用者的信息，并能够有效地对使用者的信息进行操作和管理。

具有一定的安全性，为常见的攻击方式提供可靠的防护。

能够便利企业对客服人员的管理，用户与企业售后服务人员的沟通，并且能够一定程度上降低客户服务工作的成本。

2.1.3 定义及缩写词

表 2 - 1 缩写词对照表

缩写	含义
HTML	Hyper Text Mark-up Language
HTTP	Hyper Text Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
JDBC	Java Database Connectivity
SQL	Structured Query Language
UC	Use Case
PC	Personal Computer

1. 服务单

服务单是由客户提出的一个或多个服务申请组成的、客服人员 and 用户进行的一次会话；也是客户阐明自己的需求，客服人员对其进行服务的依据。

2. 用户

狭义上，用户在本文中指代需要企业提供售后服务，申请服务单、要求企业为其提供服务的用户。用户可以向服务提供商(通常是企业)申请服务。

广义上，所有注册了本产品账号的自然人或使用本产品接口的计算机软件都可以称为用户，用户根据实际情境的不同可以作为员工或企业用户存在，在一定条件下它们之间可以互相转换。

3. 员工

如无特殊说明，在本文中指代使用本产品的企业人员，包括所有属于企业的人员(包括决策人员、管理人员和普通员工)。企业人员的认证通常是由企业发出的，经过用户确认后用户可以被认证为企业的员工，拥有企业内数据的可见权限。

员工主要负责解答用户的问题，并满足他们的合理需求。一个企业中可能有许多员工。这些员工可以有不同的部门、不同的权限，并且可以分别进行管理。

4. 企业

如无特殊说明，在本文中指代通过认证入驻本产品、拥有企业频道、能够使用所有的企业功能的用户。

5. 运营人员

如无特殊说明，在本文中指代全部或一部分实际部署使用本产品并进行盈利或非盈利

活动的自然人或法人。这类人员通常拥有一台或多台服务器，或持有使用它们的权限；能够部署本产品并稳定运行，为其他用户提供服务。一般来说，运营人员还需要负责对本产品所使用的软件进行维护及升级。

6. 企业频道

经过认证的企业会创建一个包含该企业所有信息（包括但不限于服务单、员工等）的企业频道，在一个企业频道下能够包含多个分类频道。

如果企业的管理用户（或拥有权限的企业员工）需要对企业进行管理，也需要在企业频道中进行对应操作。

7. 分类频道

分类频道主要用来对服务单进行分类管理，并且通过关联部门的形式进行权限的控制。一般来说，员工仅能够查看当前部门及下属部门的数据。

8. 频道

如无特殊说明，在本文中指代用户端中用于企业员工和用户进行沟通的文字聊天频道，一般以聊天室的形式存在。

9. 分组

如无特殊说明，在本文档中指代服务单的分组。

在单个服务单难以满足用户的需要时，往往需要提出一组服务单来描述用户的需求。此时，用户提出的一组服务单称为一个分组。如果制定了单个用户服务单申请数量上限，则一组服务单共同占用一个名额。

10. 用户端

用户端是一个使用 Vue 框架开发的网页应用，它为所有使用本产品的用户构建易于使用、用户友好的图形化界面。用户端负责接受服务器传来的数据，并将其进行渲染。用户可以在用户端方便地进行各种操作。

根据不同的用户角色，用户端会渲染不同的界面：普通用户会看到其已经申请的服务单和不同的企业；企业用户能够看到自己企业的信息并能够对其进行对应的设置；运营人员能够看到系统的后台数据日志。

11. 服务端

服务端为运行在独立服务器上的管理软件，负责数据的转发与筛选，和数据库通信以及保障通信安全。

2.1.4 产品功能

本产品涉及的功能如下：

用户端注册、用户端登录、重置密码、服务单转发、搜索客服评价、查看评价详情、查看综合评价、添加问题类型、删除问题类型、搜索问题类型、修改问题类型、搜索投诉、

删除投诉、查看投诉、搜索入驻申请、同意入驻申请、驳回入驻申请、搜索客服、添加客服、注销客服、切换客服状态、搜索企业、添加企业、注销企业、切换企业状态、申请服务单、关闭服务单、重开服务单、删除服务单、发送消息、服务评价、投诉服务、企业入驻申请、查看日志、添加员工角色、删除员工角色、修改员工角色、搜索员工角色、赋予员工角色、撤销员工角色、自定义回复模板、搜索回复模板、修改回复模板、删除回复模板。

将上述所有功能全部列出过于冗杂，本文将会重点描述企业入驻申请和申请服务单两个功能。

2.1.5 用户特征

表 2-2 用户群体

角色	职业	年龄	特征和需求
用户	不限	不限	有问题需要企业提供的客户服务
员工	企业的客服人员	不限	有工作经验
运营人员	企业的管理人员、法人或董事会成员等具有决策能力的人员、以及需要涉及到相关功能的企业员工，也可以是组织或个人。	不限	部署并使用本软件，通过其功能进行盈利或非盈利性质的活动；需要管理客服人员并及时得到反馈
管理员	软件的开发者和企业的技术人员，也可以是组织或个人	不限	有工作、编码经验，需要对软件进行维护

2.2 具体需求

2.2.1 企业入驻申请

1. 窗口标题

入驻申请

2. 窗口标识符

EnterpriseImparting

3. 目的

运营人员需要在进驻前确认企业的信息，防止违反相关法律法规的企业使用本产品。故尚未进驻的企业为了使用本产品，需要进行经营资质、经营范围等的审查；在通过审查后，才能够使用本产品

4. 界面布局

公司入驻

① 等待提交申请

② 已提交申请

③ 已通过审批

公司名称

请输入公司名称

公司地址

请输入公司地址

联系方式

请输入电子邮箱

成立时间

请选择公司成立时间

营业执照

+

请上传大小不超过 5MB 格式为 png/jpg/peg 的文件

其他材料

上传文件

请上传大小不超过 5MB 格式为 doc/xls/ppt/txt/pdf 的文件

服务单数量限制

请输入公司服务单数量限制

确定

重置

清空

图 2-1 公司入驻申请界面

5. 数据项描述

表 2-3 企业入驻申请数据项

名称	类型	精度	有效范围	格式	长度限制	度量单位	缺省	是否可空	数据来源	备注
企业 Id	字符串	N A	NA	^[0-9a-zA-Z]*\$	20 字	NA	无	否	系统	
企业名称	字符串	N A	NA	^[0-9a-zA-Z\u4e00-\u9fa5]*\$	20 字	NA	无	否	用户	
地址	字符串	N A	NA	^[0-9a-zA-Z\u4e00-\u9fa5]*\$	40 字	NA	无	否	用户	
成立时间	日期	年	历史日期	^d*\$	NA	NA	无	否	用户	软件校验
联系方式	字符串	N A	NA	\w[-\w.+]*@([A-Za-z0-9][-A-Za-z0-9]+\.)+[A-Za-z]+	20 字	NA	无	否	用户	邮箱
营业执照	字符串	N A	NA	^[^s]+\$	100 字	NA	无	否	用户	图像路径

其他材料	字符串	N A	NA	^[^s]+\..pdf\$	100 字	NA	无	否	用户	存储 路径
------	-----	--------	----	----------------	----------	----	---	---	----	----------

6. 数据项之间的关系

无

7. 窗口有效性

从用户点击“公司入驻”按钮开始，到用户离开公司入驻申请页面为止

8. 与其它输入/输出的关系

作为企业入驻审批、查询企业信息、修改企业信息的输入

9. 窗口布局

层叠

10. 反馈信息

系统弹窗提示操作成功

11. 命令的执行方式

用户点击侧边栏的“企业入驻”按钮，填写完信息后点击“确定”按钮执行

2.2.2 申请服务单

1. 窗口标题

申请服务单

2. 窗口标识符

ApplyTicket

3. 目的

有客户服务需求的用户向企业请求服务，在这个窗口中填写必要的信息组成一张服务单，向企业的客户服务人员说明遇到的问题以及需求的服务类型

4. 界面布局

图 2-2 申请服务单界面布局

5. 数据项描述

表 2-4 申请服务单数据项表述

名称	类型	精度	有效范围	格式	长度限制	度量单位	缺省	是否可空	数据来源	备注
服务单 ID	整型	NA	NA	^\d\$	64 位	NA	无	否	系统	
用户 ID	整型	NA	NA	^\d\$	64 位	NA	无	是	系统	
服务单标题	字符串	NA	NA	^[0-9a-zA-Z\u4e00-\u9fa5]\$	100 字	字	无	否	用户	
服务单类型	整型	NA	NA	^\d\$	64 位	NA	无	否	用户	

6. 数据项之间的关系

无

7. 窗口有效性

从用户点击“”

8. 与其它输入/输出的关系

作为关闭服务单、重开服务单、删除服务单的输入。

接受搜索问题类型的输出。

9. 窗口布局

层叠

10. 反馈信息

系统弹出提示框

11. 命令的执行方式

用户进入企业详情页面，点击“申请服务”按钮执行

2.3 功能需求

由于本项目采用面向对象的方法进行开发，因此利用 Use-Case 图描述系统的功能需求。

本项目涉及的 Use-Case 图见图 2-5 运营人员 Use-Case 用例图、图 2-3 用户 Use-Case 用例图、图 2-4 企业 Use-Case 用例图。

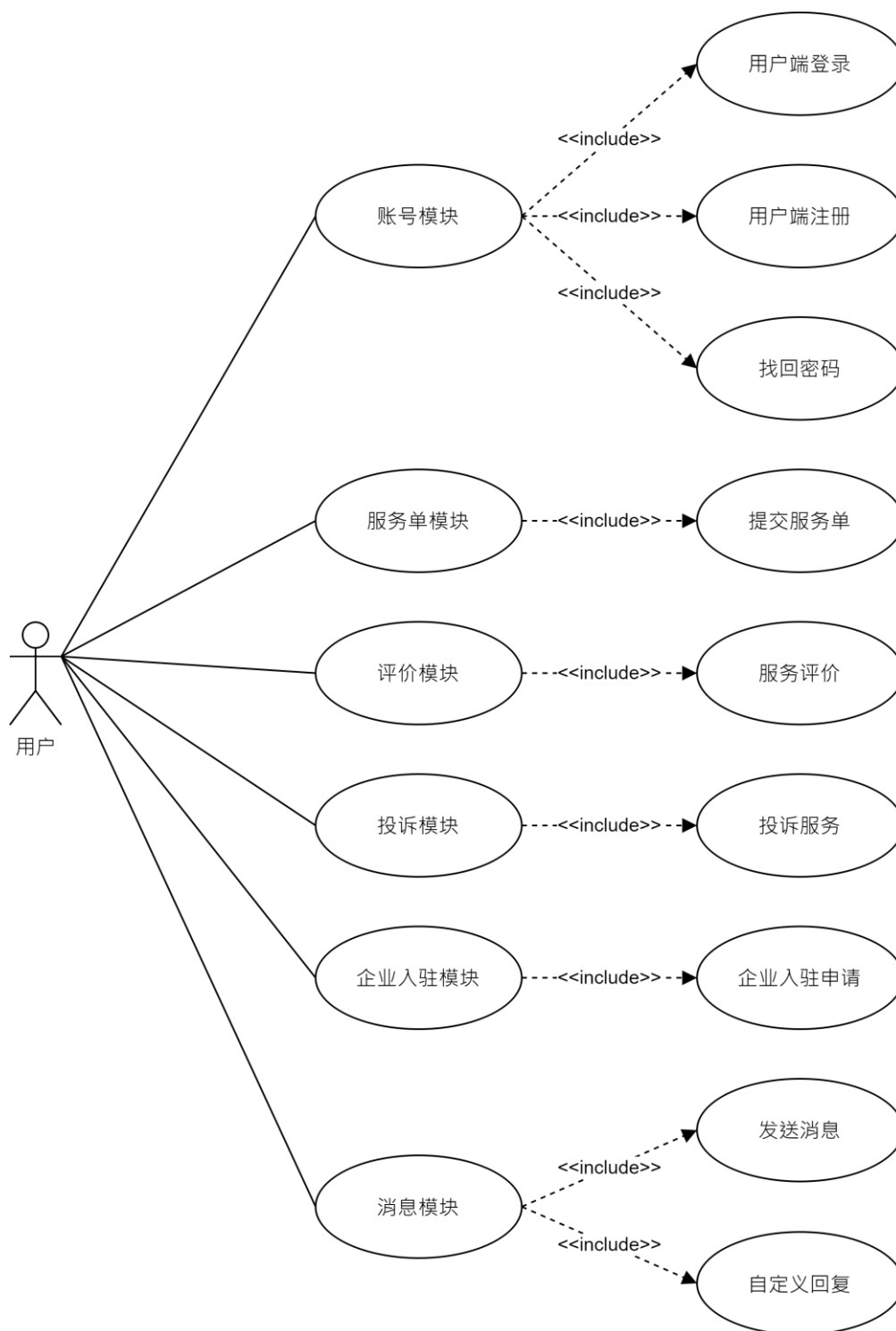


图 2 - 3 用户 Use-Case 用例图

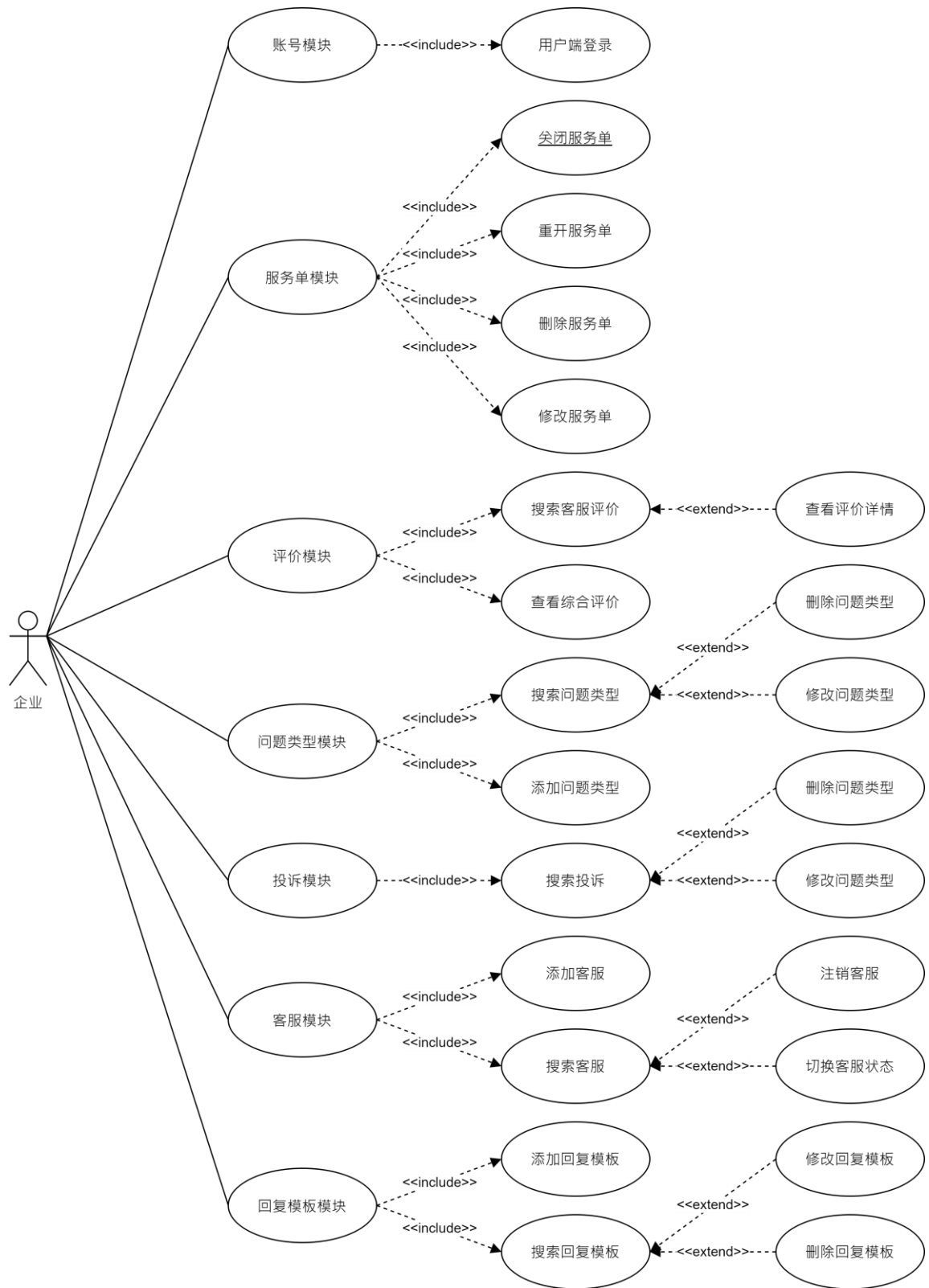


图 2 - 4 企业 Use-Case 用例图

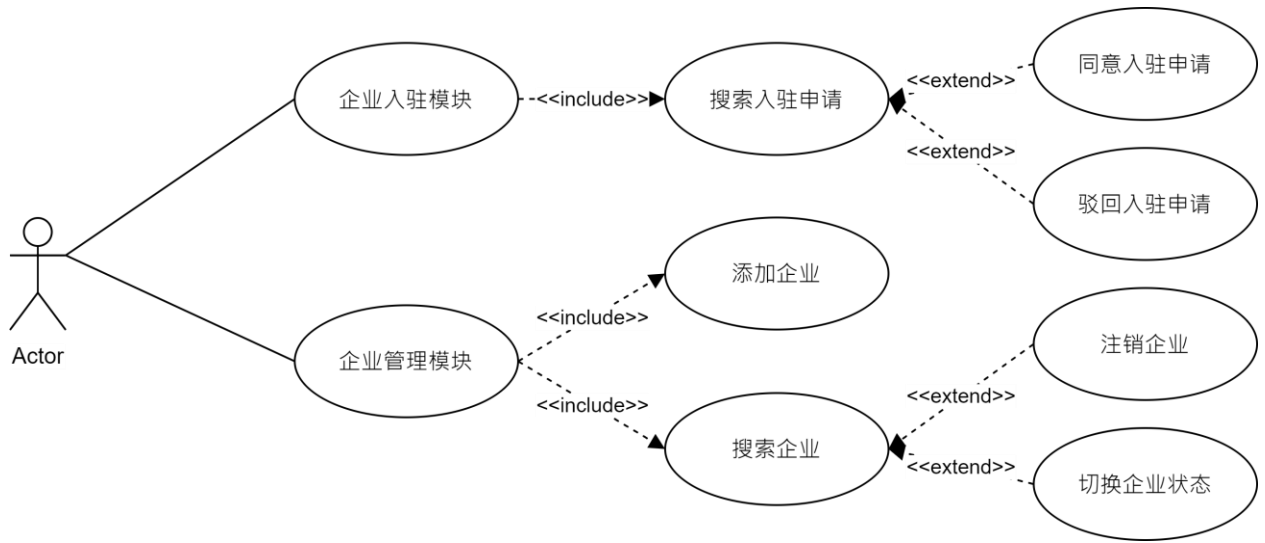


图 2-5 运营人员 Use-Case 用例图

2.3.1 用例列表

表 2-5 易生活服务系统用例列表描述了本系统中全部的用例。受限于篇幅，本文中仅选取企业入驻申请、申请服务单两个用例进行详细描述。

表 2-5 易生活服务系统用例列表

序号	用例名	用例标识符
1	申请服务单	ApplyTicket
2	提交服务单	SubmitTicket
3	关闭服务单	CloseTicket
4	重开服务单	ReopenTicket
5	删除服务单	DeleteTicket
6	自定义回复	SetReply
7	服务单转发	TicketForward
8	服务评价	Feedback
9	投诉服务	Complaints
11	搜索客服评价	SearchServerEvaluation
12	查看评价详情	SeeEvaluationDetails

13	查看综合评价	SeeComprehensiveEvaluation
14	添加问题类型	AddQuestionType
15	删除问题类型	DeleteQuestionType
16	搜索问题类型	SearchQuestionType
17	修改问题类型	ModifyQuestionType
18	搜索投诉	SearchComplaint
19	查看投诉	SeeComplaint
20	删除投诉	DeleteComplaint
21	企业入驻申请	EnterpriseImparting
22	搜索入驻申请	SearchSettlementApplication
23	同意入驻申请	AgreeSettlementApplication
24	驳回入驻申请	RejectSettlementApplication
25	自定义角色	CompanyRoleManage
26	搜索客服	SearchServer
27	添加客服	AddServer
28	注销客服	CancelServer
29	重置客服密码	ResetServerPassword
30	切换客服状态	ExchangeServerStatus
31	搜索企业	SearchEnterprise
32	添加企业	AddEnterprise
33	注销企业	CancelEnterprise
34	重置企业密码	ResetEnterprisePassword
35	切换企业状态	ExchangeEnterpriseStatus
36	修改员工角色	ModifyCompanyRole
37	撤销员工角色	CancelCompanyRole

38	赋予员工角色	GrantCompanyRole
39	查看日志	ReviewLogs
40	客户端登录	ClientLogin
41	企业端登录	EnterpriseLogin

2.3.2 用例表

1. 企业入驻审批用例表

表 2-6 企业入驻审批用例表

用例标识符	EnterpriseImparting		
用例名称	企业入驻审批		
用例创建者	姜山	用例最后修改者	姜山
用例创建时间	2022 年 9 月 20 日	用例最后修改时间	2023 年 2 月 25 日
操作者	企业代表		
描述	企业对企业入驻申请进行审批		
前置条件	企业代表成功登录，系统处于用户端首页(UserInterface)		
后置条件	系统向企业信息表(companyInfoList)中写入一条新的数据，返回用户端首页(UserInterface)		
主事件流	企业代表	系统	
	1 企业代表点击用户端首页(UserInterface)的企业入驻申请按钮	2 系统进入企业入驻申请界面(EnterpriseImpartingView)。	
	3 企业代表填写公司名称、联系方式等信息，并上传营业执照和其他辅助材料，点击“确定”按钮	4 系统核验填入的信息是否符合要求，若不符合，进入子事件流 a；若符合，系统填入公司入驻所需的默认信息，并向企业信息表(CompanyInfoList)中写入一条新的数据，之后返回公司入驻界面并弹出申请提交提示框(ApproveSubmitDialog)	

子事件流 a	1 系统拦截本次提交，标红需要填写的项目并返回主事件流 a
异常处理	


 已提交申请

图 2-6 申请提交提示框

2. 申请服务单用例表

表 2-7 申请服务单用例表

用例标识符	SubmitTicket		
用例名	提交服务单		
用例的创建者	姜山	用例的最后修改者	姜山
用例的创建时间	2022 年 7 月 13 日	用例的最后修改日期	2023 年 3 月 28 日
操作者	用户		
描述	用户对需求的服务进行描述，填写并提交一张服务单申请获取企业的售后服务。		
前置条件	用户登录用户端，系统正常运行且位于服务单管理界面(TicketManager)。		
后置条件	将服务单信息登记入服务单信息表(TicketInfoList)中，返回服务单管理界面(TicketManager)		
主事件流	用户	系统	
	1 用户点击企业名称旁边的“申请服务”按钮	2 系统检查服务单信息表(TicketInfoList)中是否存在该用户已经申请的服务单，并且校验总数是否超过企业设置的上限。如果超过，则进入子事件流 a；否则进入新建服务单页面(ApplyTicketDialog)。	
	3 用户在新建服务单页面(ApplyTicketDialog)中填写信息，选择对应的问题类型，并点击“提交”按钮	4 系统在服务单信息表(TicketInfoList)中添加一条数据，弹出服务单申请成功提示框(TicketAppliedDialog)并返回。若添加失败，则进入子事件流 b;返回服务单管理界面(TicketManager)。	

子事件流 a	1 系统弹出服务单申请超出上限提示框(TicketLimitExceedDialog)，返回用户端主页面。。
子事件流 b	1 系统弹出服务单添加失败提示框(ApplyError)，返回服务单管理界面并请求新的服务单信息表(TicketInfoList)。
异常处理	



图 2-7 服务单申请成功提示框



图 2-8 申请数量超出上限提示框



图 2-9 服务单添加失败提示框

2.4 数据库需求

本系统使用的数据库，支持数据库的增加、删除、修改、查询等基本操作，可以对字符串、数字、日期、时间等基本数据类型进行存储。数据实体关系图如图 2-10 易生活服务系统数据实体关系图（图中未出现部分 RuoYi-Vue 框架原生内容）：

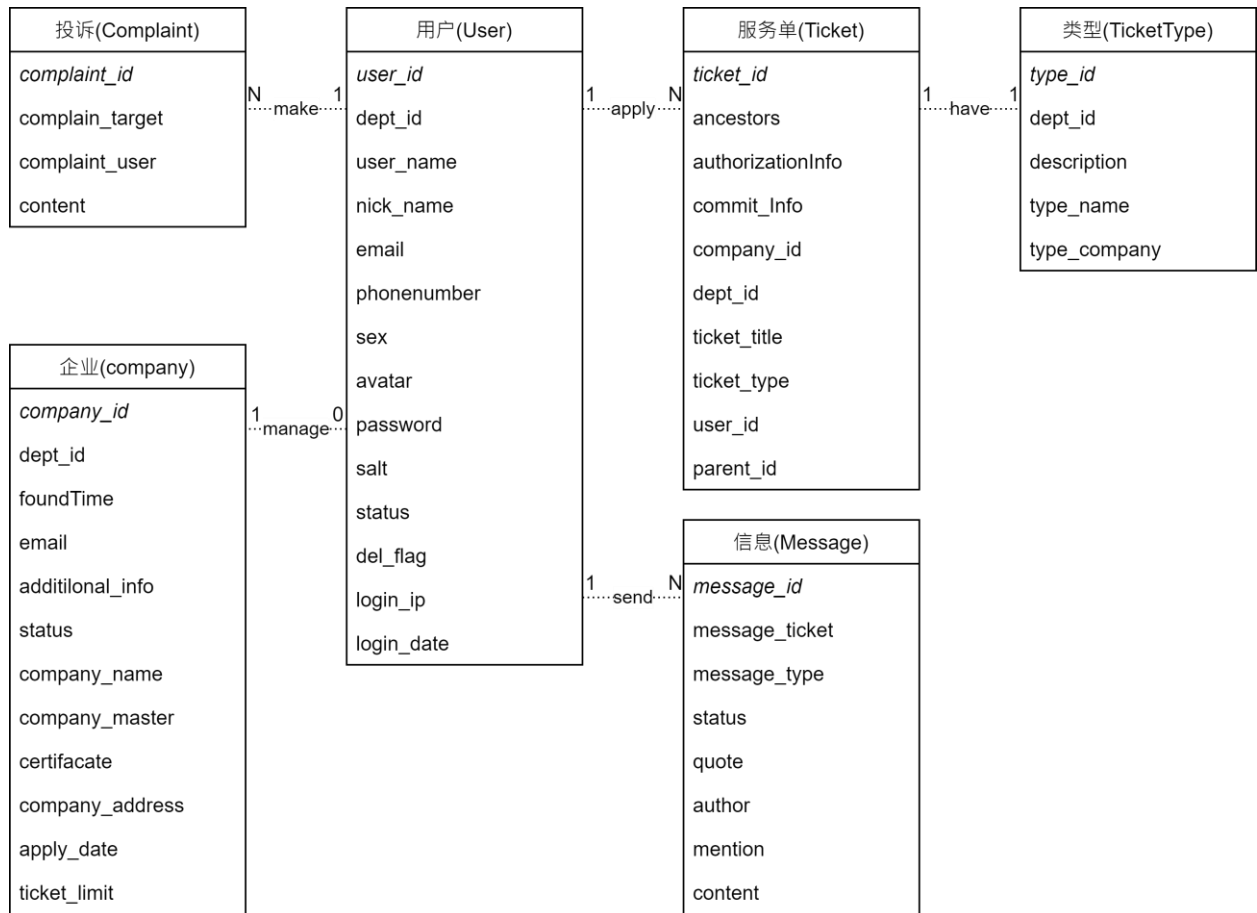


图 2 - 10 易生活服务系统数据实体关系图

2.5 系统的质量属性

本节中，主要从可靠性、可获得性、保密性、可维护性四个方面来描述系统的质量。

2.5.1 可靠性

本系统的可靠性为 99%，每年停机维护时间不超过 88 小时；运行时漏洞不超过一个。

2.5.2 可获得性

本系统在维护期间用户无法正常获得系统提供的相关服务，其他时间可正常使用。

2.5.3 保密性

本系统无论是客户、企业还是管理员想要获得系统服务必须用账号密码登录后才能获取。当密码发生变动时需要使用短信验证码修改或者线下联系管理员帮忙修改。

2.5.4 可维护性

如无特殊情况，服务端大型维护不超过 1 天，小型维护不超过 3 个小时。发生特殊情况将适当增加维护时间。

用户端程序允许热更新。

2.6 需求模型

2.6.1 静态模型

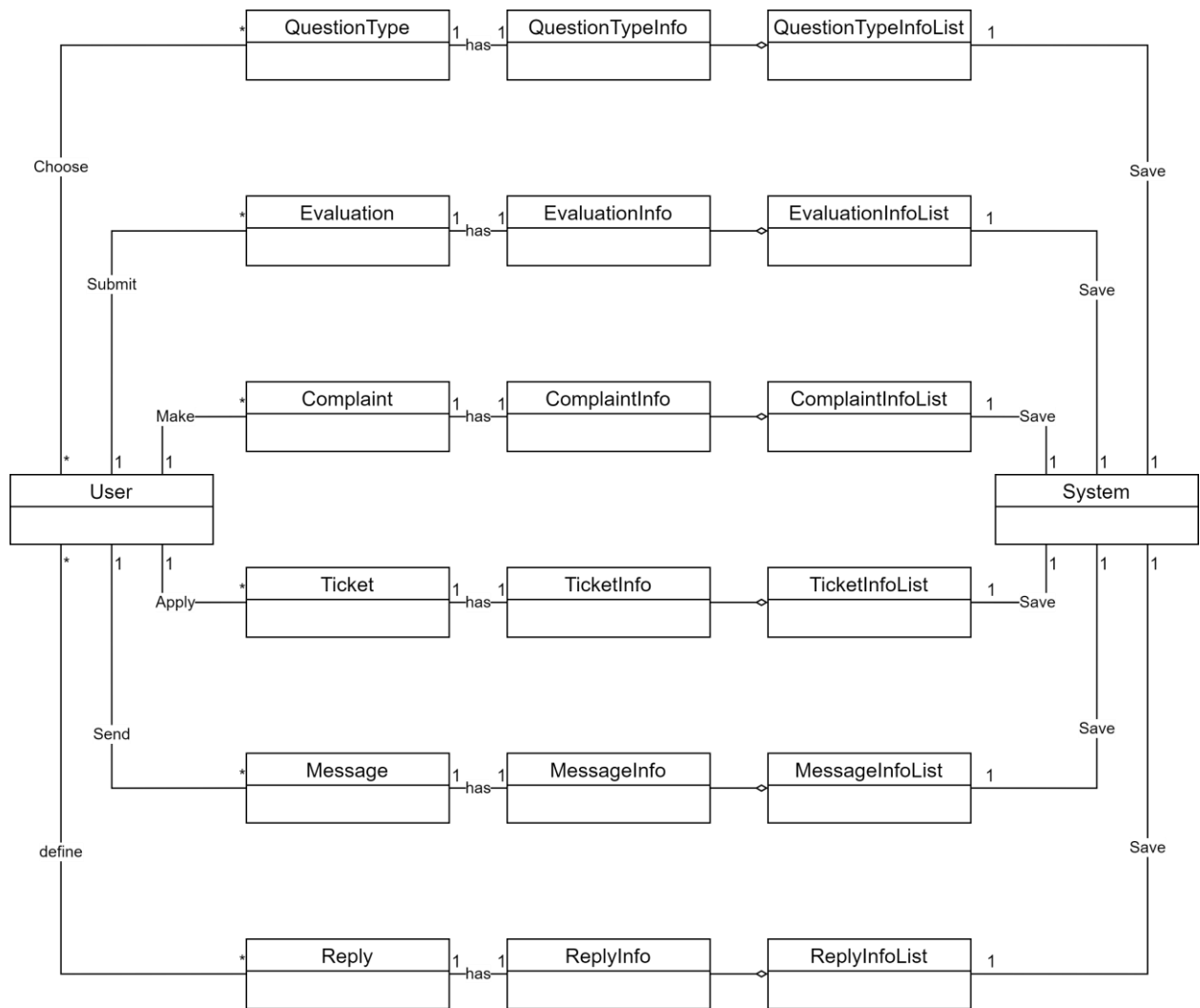


图 2 - 11 与用户有关的静态模型

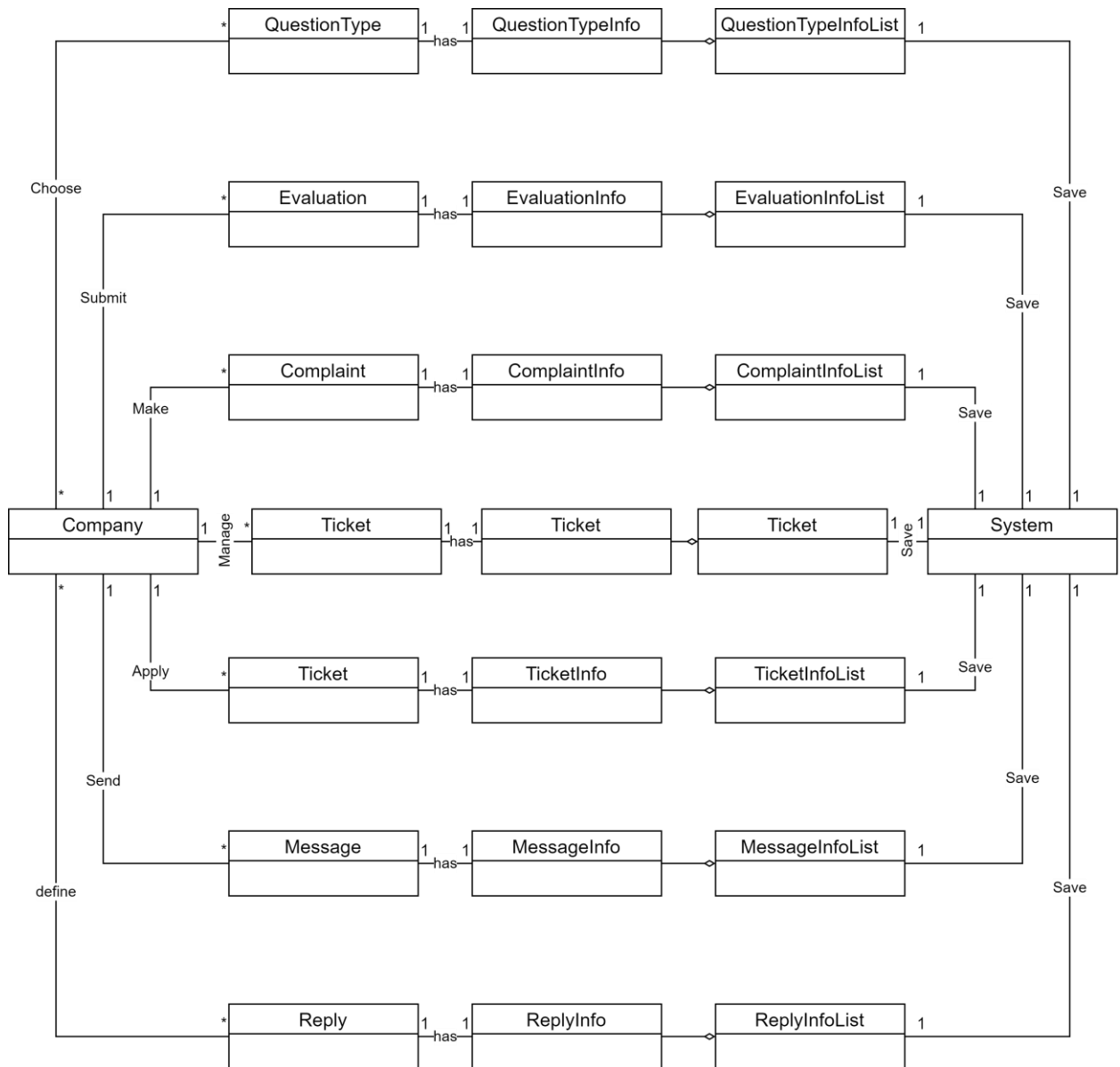


图 2 - 12 与企业有关的静态模型

2.6.2 动态模型

1. 企业入驻申请序列图

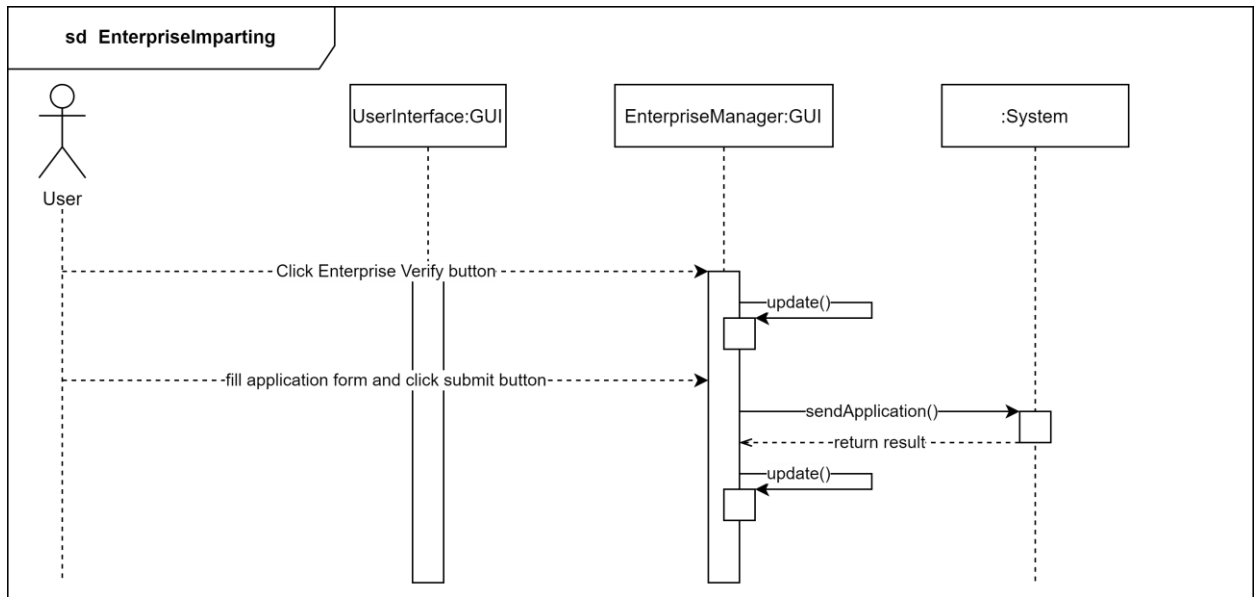


图 2 - 13 企业入驻申请序列图

注:

UserInterface: 用户端主界面

EnterpriseManager: 企业数据管理界面

System::sendApplication(Company company): Company

前置条件: 输入的数据参照 表 3.10 企业入驻申请数据项

后置条件: 系统向企业信息表(CompanyInfoList)中添加一条数据并返回。若操作成功, 返回带有申请数据的数据集合; 若操作失败, 返回 Null。

2. 申请服务单序列图

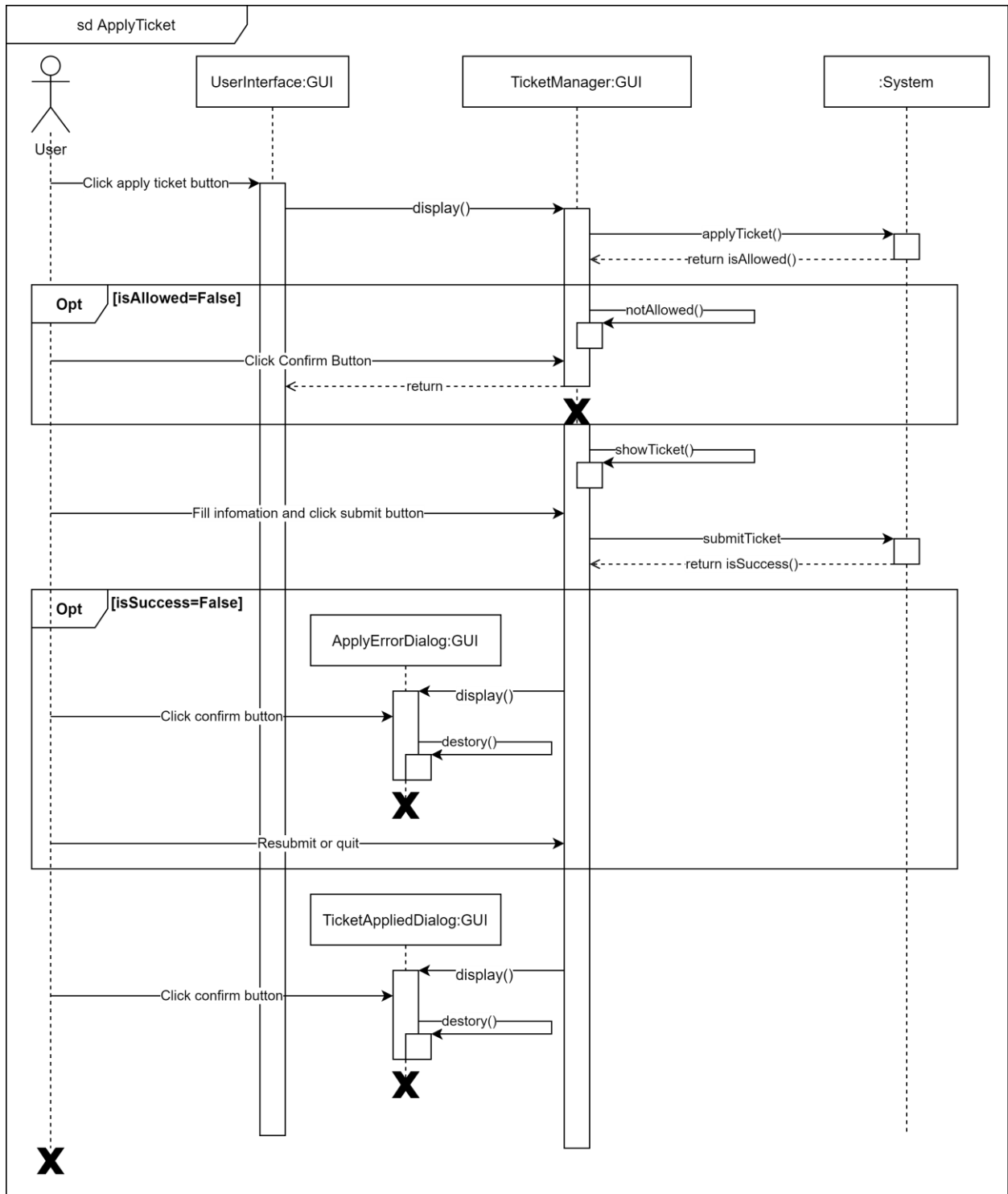


图 2 - 14 申请服务单序列图

注：

UserInterface: 用户端主界面

TicketManager: 服务单管理界面

ApplyErrorDialog: 提交失败提示框

TicketAppliedDialog: 提交成功提示框

System::applyTicket(String user): Bool

前置条件: user 是当前登录的用户编号, 参照申请服务单数据项表述。

后置条件: 系统在服务单信息表(TicketInfoList)和企业信息表(CompanyInfoList)中查询: 用户是否已经耗尽其在某个服务器中的服务单配额, 若未耗尽, 则在服务单信息表(TicketInfoList)中更新数据并返回 True; 若已耗尽, 则返回 False

System::submitTicket(Ticket ticket): Ticket

前置条件: ticket 参照申请服务单数据项表述,

后置条件: 系统向服务单信息表(TicketInfoList)中保存或修改一条数据, 若成功, 将更新后的数据返回; 若失败, 则返回 Null。

第三章 体系结构设计

3.1 部署视图

3.1.1 概述

本系统的部署视图见图 3-1 易生活服务系统部署图：

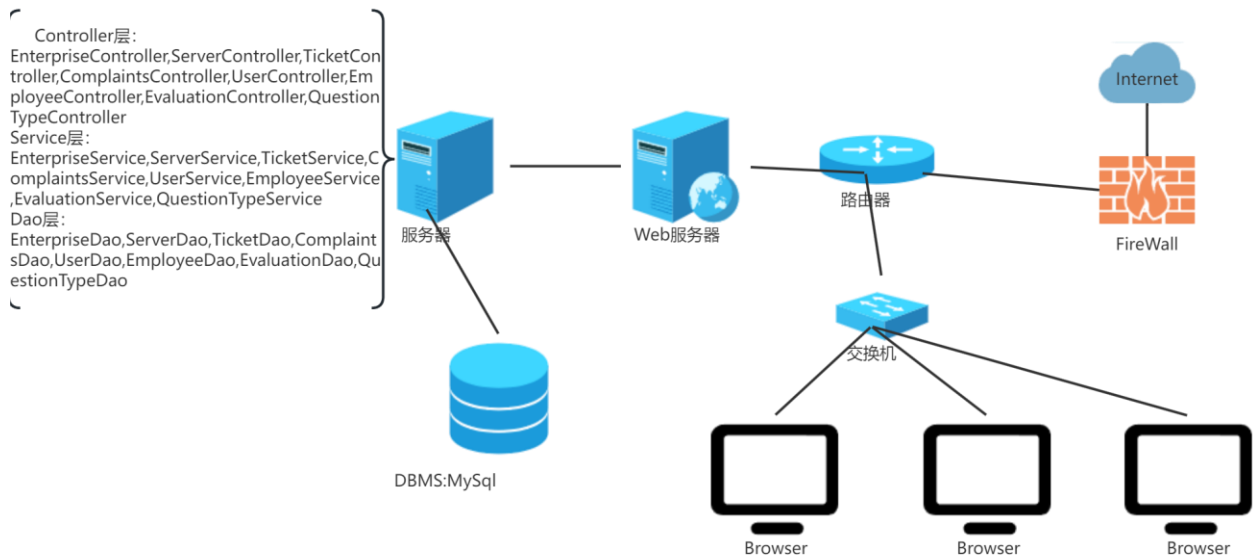


图 3-1 易生活服务系统部署图

3.1.2 部署政策

本系统的部署政策见表 3-1 易生活服务系统部署政策表：

表 3-1 易生活服务系统部署政策表

	用户端	服务端
浏览器名称	Google Chrome（谷歌浏览器）	Google Chrome（谷歌浏览器）
浏览器版本	86.0（64 位）版本以上	86.0（64 位）版本以上
应用操作系统	Windows10 及以上	Windows10 及以上
应用服务器软件	Apache Tomcat 9.0 及以上	Apache Tomcat 9.0 及以上
数据服务器操作系统	Windows 10 及以上	Windows 10 及以上
数据服务器数据库版本	SqlServer2017 及以上	SqlServer2017 及以上

3.2 逻辑视图

3.2.1 概述

本系统采用 B/S 模式，共分为七层。这七层分别为：

View（用户交互层）、Controller（控制层）、IService（业务服务接口层）、Service（业务服务层）、Mapper（数据访问层）、Dao（领域模型层）和 the object value class（值对象类包）。

本系统包图见图 3-2 易生活服务系统包图：

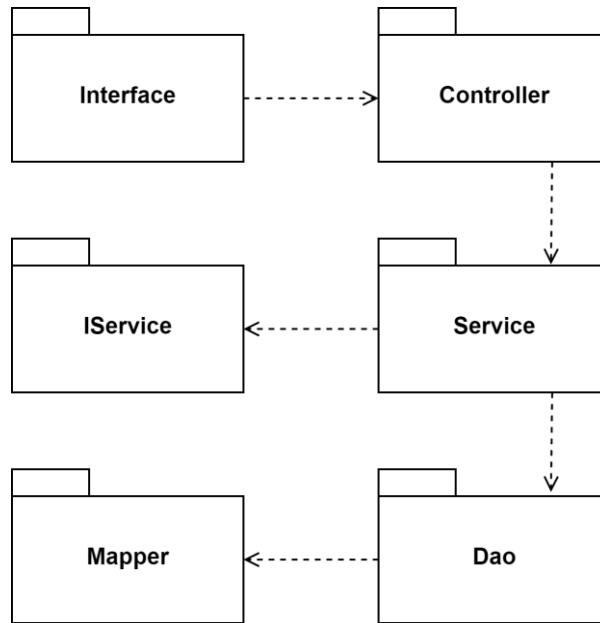


图 3-2 易生活服务系统包图

3.2.2 包或子系统

1. 问题类型类关系图

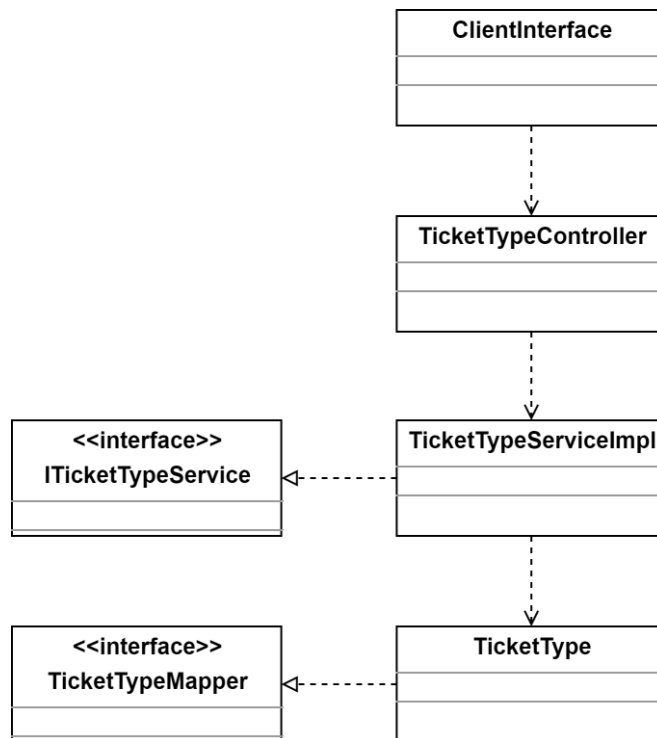


图 3 - 3 问题类型关系图

2. 投诉类关系图

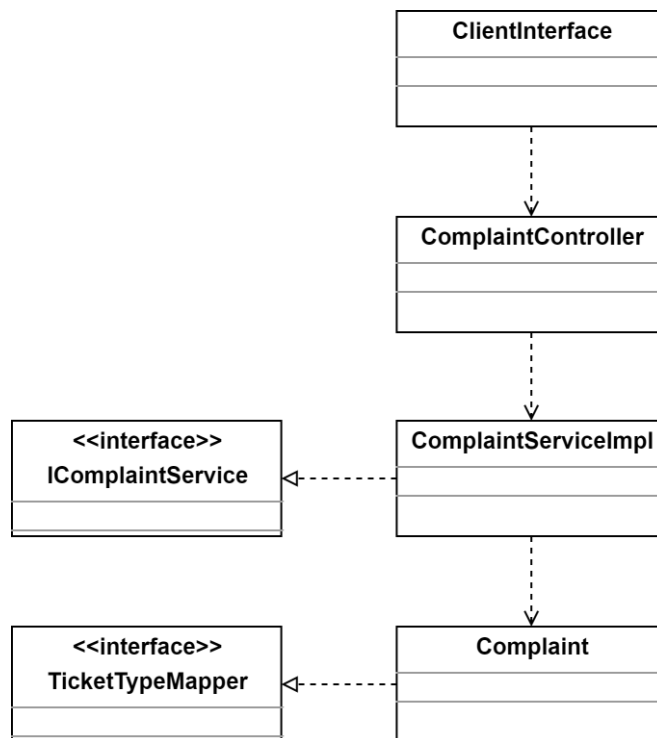


图 3 - 4 投诉关系图

3. 评价类关系图

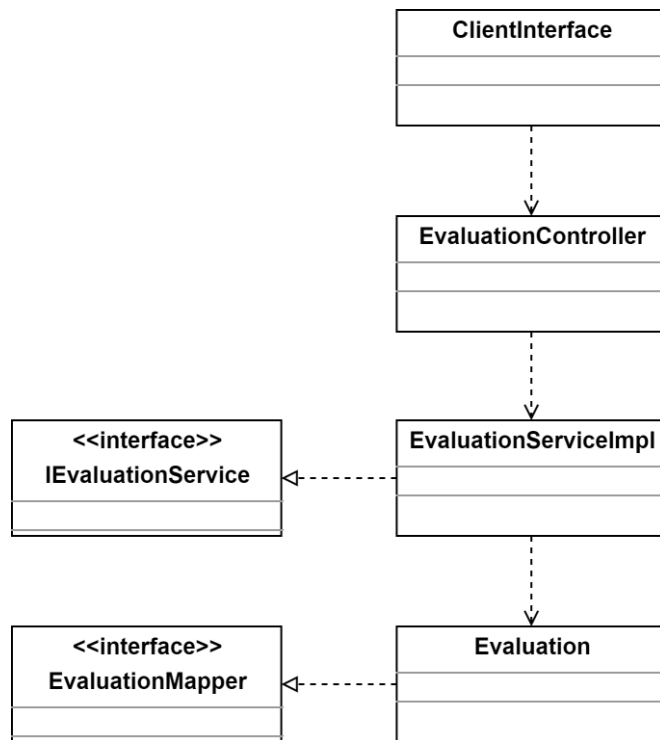


图 3 - 5 评价关系图

4. 用户类关系图

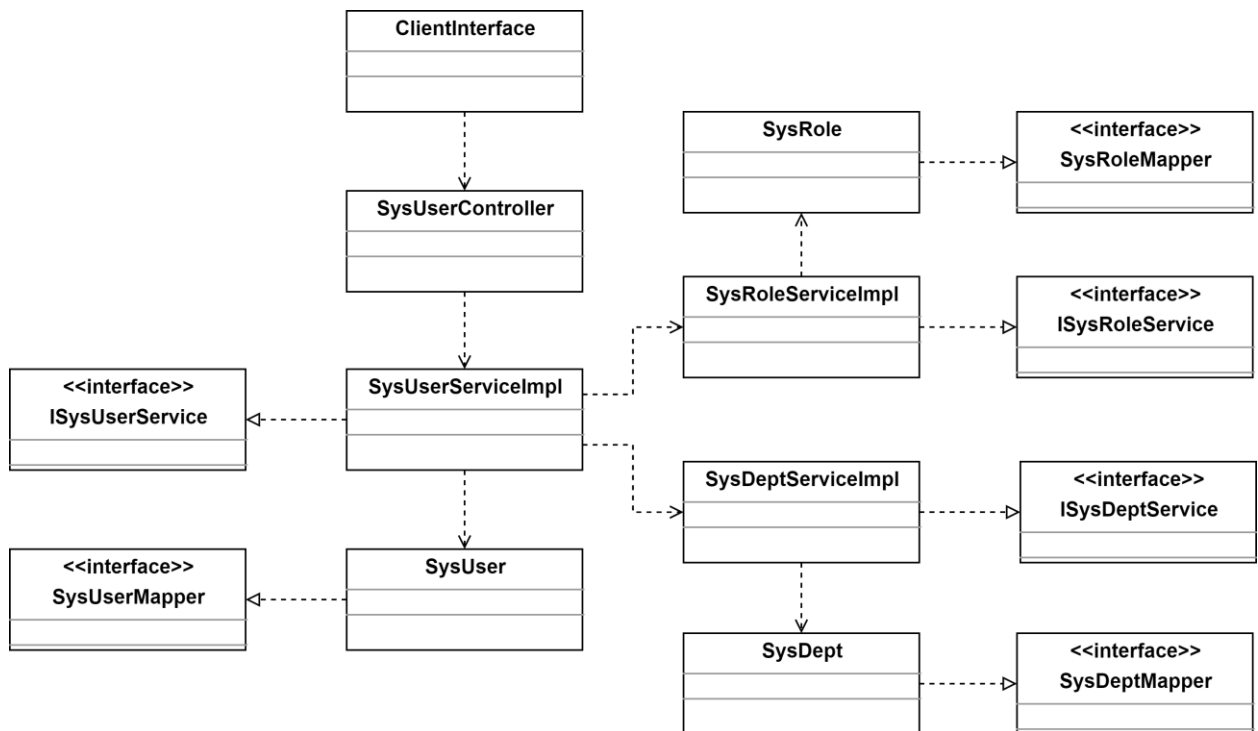


图 3 - 6 用户关系图

5. 公司类关系图

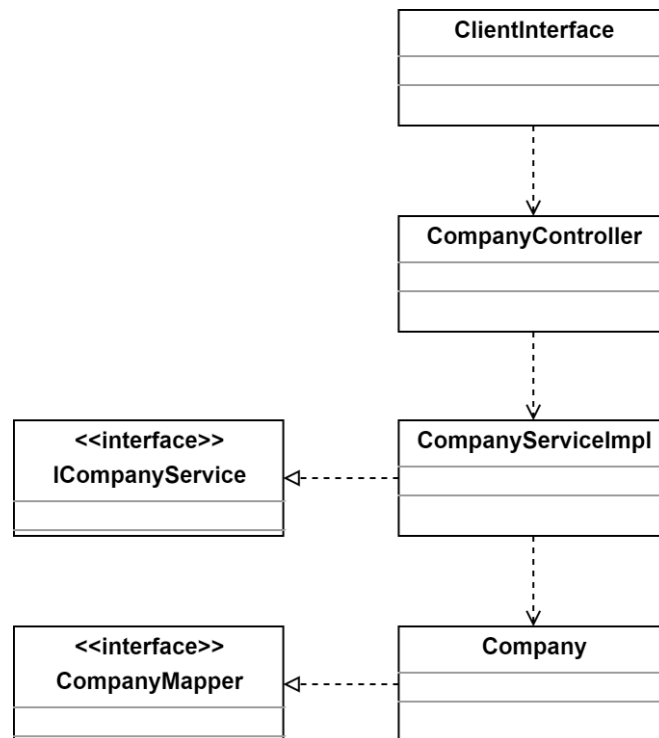


图 3 - 7 公司关系图

3.3 进程视图

3.3.1 概述

1) 企业入驻申请的进程视图

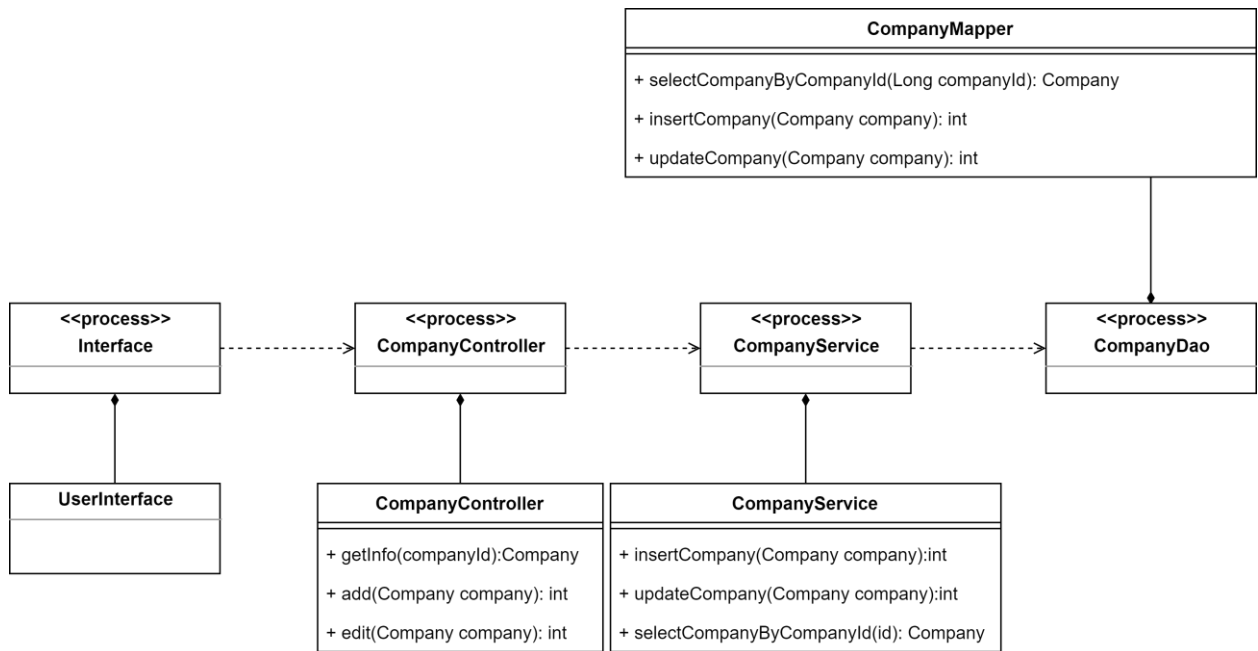


图 3 - 8 企业入驻申请进程视图

3.3.2 过程间通信机制

系统客户端与应用服务器间通过 JSON 数据传输格式传输数据，通过 URL 将客户端信息提交到应用服务器。应用服务器与数据库使用 JDBC 连接，通过 TDS 协议进行数据通信。应用服务器与设备之间使用 TCP 通信。

3.4 Use-Case 视图

1. 企业入驻申请

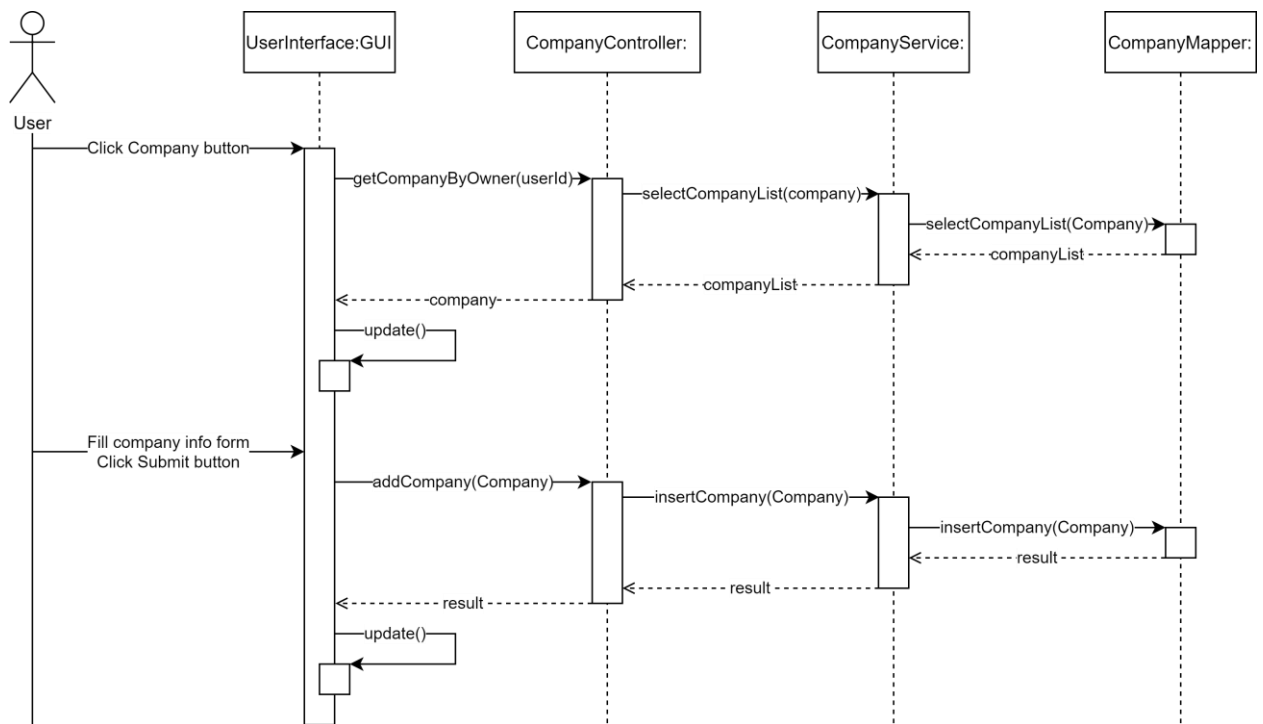


图 3 - 9 企业入驻申请 Use-Case 图

注:

UserInterface::update():void

CompanyController::getCompanyByOwner(Long userId):Company

CompanyService::selectCompanyList(Company company): List<Company>

CompanyMapper::selectCompanyList(Company company):List<Company>

CompanyController::addCompany(Company company):int

CompanyService::insersrtCompany(Company company):int

CompanyMapper::insersrtCompany(Company company):int

2. 申请服务单

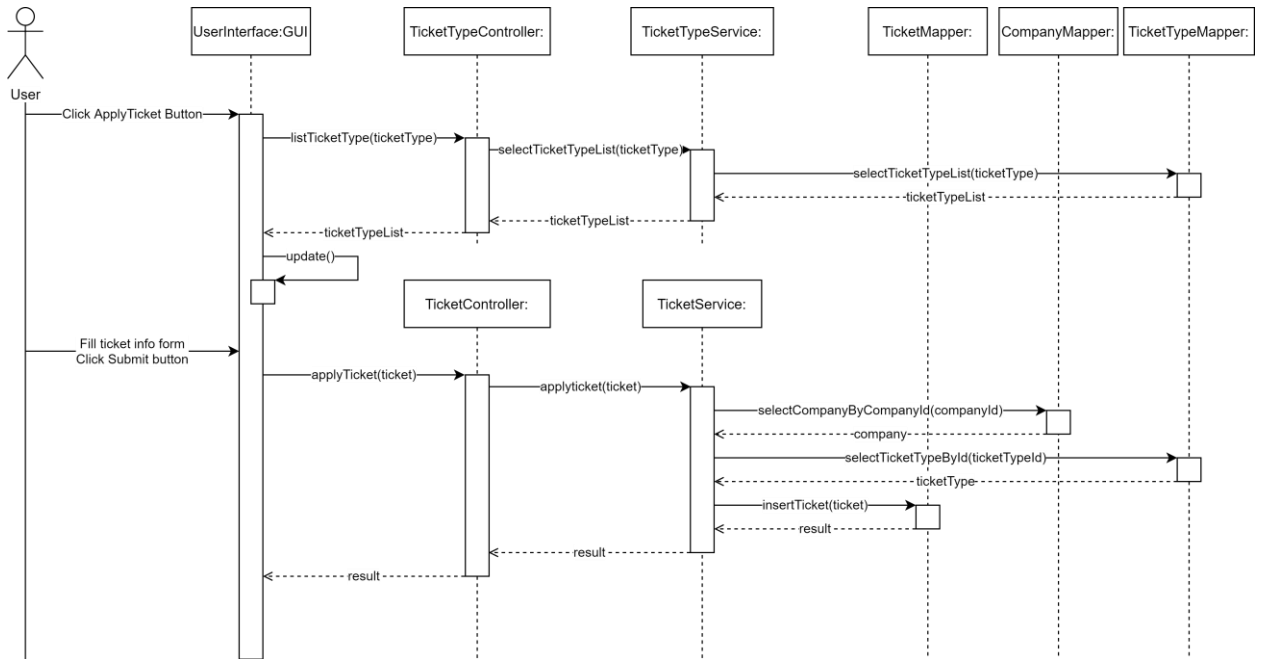


图 3 - 10 申请服务单 Use-Case 视图

注：

TicketTypeController::listTicketType(ticketType):List<TicketType>

TicketTypeService::selectTicketTypeList(ticketType):List<TicketType>

TicketTypeMapper::selectTicketTypeList(ticketType):List<TicketType>

TicketController::applyTicket(ticket):int

TicketService::applyTicket(ticket):int

CompanyMapper::selectCompanyByCompanyId(companyId):Company

TicketTypeMapper::selectTicketTypeById(ticketTypeId):TicketType

TicketMapper::insertTicket(ticket):int

3.5 实现视图

3.5.1 业务流程层

1. CompanyController

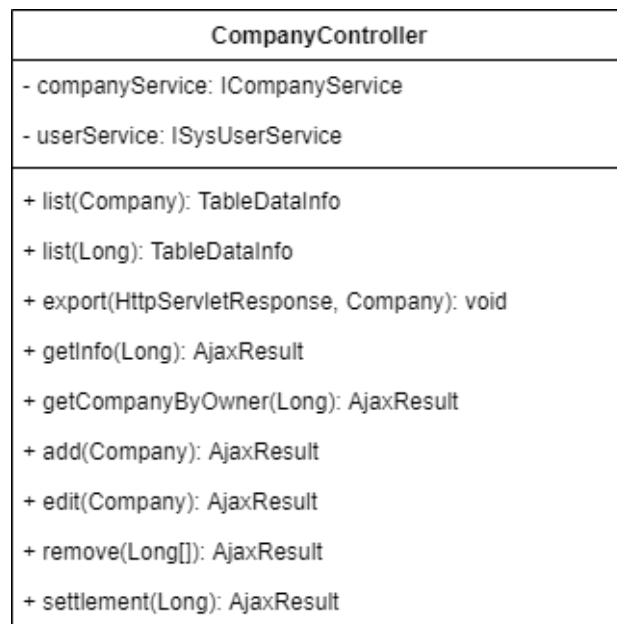


图 3 - 11 CompanyController 类

方法介绍及伪码:

CompanyController::list(Company company): TableDataInfo

前置条件: 查询条件(company)不为空。

后置条件: 系统根据传入的约束条件从企业信息表(CompanyInfoList)中查询指定范围内符合条件的数据组成的集合, 将其封装成用于前后端交互时的 JSON 格式并返回。

```
CompanyController::list(Company company): TableDataInfo {
    startPage();
    list ← companyService.selectCompanyList(company);
    return getDataTable(list);
}
```

CompanyController::list(Long companyId): TableDataInfo

前置条件: 公司 Id(companyId)不为空。

后置条件: 系统根据传入的约束条件从用户信息表(UserInfoList)中查询所有符合条件的数据组成的列表, 将其封装成用于前后端交互时的 JSON 格式并返回。

```
CompanyController::list(Long companyId): TableDataInfo {
    List<SysUser> list ← userService.selectUserByCompany(companyId);
    return getDataTable(list);
}
```

CompanyController::getInfo(Long companyId): AjaxResult

前置条件：公司 Id(companyId)不为空。

后置条件：系统从公司信息表(CompanyInfoList)中查询指定公司的数据，将其封装成用于前后端交互的 JSON 格式并返回。

```
CompanyController::getInfo(Long companyId): AjaxResult {
    return
    AjaxResult.success(companyService.selectCompanyByCompanyId(companyId));
}
```

CompanyController::getCompanyByOwner(Long userId): AjaxResult

前置条件：用户 Id(userId)不为空。

后置条件：系统从公司信息表(CompanyInfoList)中查询指定公司的数据，将其封装成用于前后端交互的 JSON 格式并返回。

```
CompanyController::getCompanyByOwner(Long userId): AjaxResult {
    Company query ← new Company();
    query.setCompanyMaster(userId);
    List<Company> resList ← companyService.selectCompanyList(query);
    If (resList.size() > 0) {
        return AjaxResult.success(resList.get(0));
    } else {
        return AjaxResult.success();
    }
}
```

CompanyController::add(Company company): AjaxResult

前置条件：公司信息(company)不为空。

后置条件：系统设置新公司的初始信息，向公司信息表(CompanyInfoList)中新增一条数据并返回操作结果。

```
CompanyController::add(Company company): AjaxResult {
    company.setCompanyMaster(getLoginUser().getUserId());
    company.setApplyDate(DateUtils.getNowDate());
    company.setStatus(0);
    return toAjax(companyService.insertCompany(company));
}
```

CompanyController::edit(Company company): AjaxResult

前置条件：公司信息(company)不为空。

后置条件：系统在公司信息表(CompanyInfoList)中修改指定公司的数据并返回操作结果。

```
CompanyController::edit(Company company): AjaxResult {  
    return toAjax(companyService.updateCompany(company));  
}
```

CompanyController::remove(Long[] companyIds): AjaxResult

前置条件：公司 Id 列表(companyIds)不为空。

后置条件：系统从公司信息表(CompanyInfoList)中删除指定公司的数据并返回操作结果。

```
CompanyController::remove(Long[] companyIds): AjaxResult {  
    return toAjax(companyService.deleteCompanyByCompanyIds(companyIds));  
}
```

CompanyController::settlement(Long companyId): AjaxResult

前置条件：公司 Id(companyId)不为空。

后置条件：系统从公司信息表(CompanyInfoList)中修改指定公司的状态为“已入驻”并返回操作结果。

```
CompanyController::settlement(Long companyId): AjaxResult {  
    return toAjax(companyService.updateCompany(company));  
}
```

2. TicketTypeController

TicketTypeController
- ticketTypeService: ITicketTypeService
+ list(TicketType): TableDataInfo + export(HttpServletResponse, Company): void + getInfo(Long): AjaxResult + add(TicketType): AjaxResult + edit(TicketType): AjaxResult + remove(Long[]): AjaxResult

图 3 - 12 TicketTypeController 类

方法介绍及伪码:

TicketTypeController::list(TicketType ticketType): TableDataInfo

前置条件: 查找约束条件(ticketType)不能为空。

后置条件: 系统在服务单类型信息表(TicketTypeInfoList)中查找所有符合要求的服务单类型并返回。

```
TicketTypeController::list(TicketType ticketType): TableDataInfo {
    startPage();
    List<TicketType> list ← ticketTypeService.selectTicketTypeList(ticketType);
    return getDataTable(list);
}
```

TicketTypeController::getInfo(Long typeId): AjaxResult

前置条件: 服务单类型 Id(typeId)不能为空。

后置条件: 系统在类型信息表(TicketTypeInfoList)中查找对应编号的服务单类型并返回。

```
TicketTypeController::getInfo(Long typeId): AjaxResult{
    return AjaxResult.success(ticketTypeService.selectTicketTypeById(typeId));
}
```

TicketTypeController::add(TicketType ticketType): AjaxResult

前置条件: 服务单类型信息(ticketType)不能为空。

后置条件: 系统向服务单类型信息表(TicketTypeInfoList)中写入一条数据, 返回操作结果。

```
TicketTypeController::add(TicketType ticketType): AjaxResult {
    return toAjax(ticketTypeService.insertTicketType(ticketType));
}
```

}

TicketTypeController::edit(TicketType ticketType): AjaxResult

前置条件：服务单类型信息(ticketType)不能为空。

后置条件：系统在服务单类型信息表(TicketTypeInfoList)中查找对应编号的服务单类型，将其修改后返回操作结果。

```
TicketTypeController::edit(TicketType ticketType): AjaxResult {
    return toAjax(ticketTypeService.updateTicketType(ticketType));
}
```

TicketTypeController::remove(Long[] typeIds): AjaxResult

前置条件：服务单类型编号列表(typeIds)不能为空。

后置条件：系统在服务单类型信息表(TicketTypeInfoList)中查找对应编号的服务单类型，将对应信息删除后返回操作结果。

```
TicketTypeController::remove(@PathVariable Long[] typeIds): AjaxResult {
    return toAjax(ticketTypeService.deleteTicketTypeByTypeIds(typeIds));
}
```

3. TicketController

TicketController
- companyService: ICompanyService - ticketService: ITicketService
+ list(Ticket): TableDataInfo + export(HttpServletResponse, Company): void + getInfo(Long): AjaxResult + listParentTicket(Long): TableDataInfo + add(Company): AjaxResult + edit(Company): AjaxResult + remove(Long[]): AjaxResult + apply(Ticket): AjaxResult

图 3 - 13 TicketController 类

方法及伪码介绍：

```
TicketController::list(Ticket ticket): TableDataInfo {
    startPage();
    List<Ticket> list ← ticketService.selectTicketList(ticket);
    return getDataTable(list);
}
```

TicketController::getInfo(Long ticketId): AjaxResult

前置条件：服务单编号(ticketId)不能为空。

后置条件：系统从服务单信息表(ticketInfoList)中查找对应编号的服务单，将其封装为用于前后端通信的 JSON 格式并返回。

```
TicketController::getInfo(Long ticketId): AjaxResult{
    return AjaxResult.success(ticketService.selectTicketByTicketId(ticketId));
}
```

TicketController:: listChildTicket (Long ticketId): TableDataInfo

前置条件：服务单编号(ticketId)不能为空。

后置条件：系统从服务单信息表(ticketInfoList)中查找指定编号服务单的所有子服务单，将其封装为用于前后端通信的 JSON 格式并返回。

```
TicketController:: listChildTicket (Long ticketId): TableDataInfo {
    Ticket target ← new Ticket();
    target.setParentId(ticketId);

    startPage();
    List<Ticket> list ← ticketService.selectTicketList(target);
    return getDataTable(list);
}
```

TicketController::add(Ticket ticket): AjaxResult

前置条件：服务单信息(ticket)不能为空。

后置条件：系统向服务单信息表(TicketInfoList)中写入一条数据，返回操作的结果。

```
TicketController::add(Ticket ticket): AjaxResult{
    return toAjax(ticketService.insertTicket(ticket));
}
```


TicketController::edit(Ticket ticket): AjaxResult

前置条件：服务单信息(ticket)不能为空。

后置条件：系统向服务单信息表(TicketInfoList)中更新一条数据，返回操作的结果。

```
TicketController::edit(Ticket ticket): AjaxResult {  
    return toAjax(ticketService.updateTicket(ticket));  
}
```

TicketController::remove(Long[] ticketIds) AjaxResult

前置条件：服务单编号列表(ticketIds)不能为空。

后置条件：系统从服务单信息表(TicketInfoList)中删除符合条件的数据，返回操作的结果。

```
TicketController::remove(Long[] ticketIds) AjaxResult{  
    return toAjax(ticketService.deleteTicketByTicketIds(ticketIds));  
}
```

TicketController::apply(Ticket ticket): AjaxResult

前置条件：服务单信息(ticket)不能为空。

后置条件：系统向服务单信息表(TicketInfoList)中写入一条数据，返回操作的结果。

```
TicketController::apply(Ticket ticket): AjaxResult {  
    return toAjax(ticketService.applyTicket(ticket));  
}
```

3.5.2 业务逻辑层

1. CompanyServiceImpl

CompanyServiceImpl
- companyMapper: CompanyMapper - subscribeMapper: SubscribeMapper
+ selectCompanyByCompanyId(Long): Company + selectCompanyList(Company): List<Company> + insertCompany(Company): int + updateCompany(Company): int + deleteCompanyByCompanyIds(Long[]): int + deleteCompanyByCompanyId(Long): int + companySettlement(Long): int

图 3 - 14 CompanyServiceImpl 类

方法介绍及伪码:

CompanyServiceImpl::selectCompanyByCompanyId(Long companyId): Company

前置条件: 公司编号(companyId)不能为空。

后置条件: 系统从公司信息表(CompanyInfoList)中选择对应的公司信息并将其返回。

```
CompanyServiceImpl::selectCompanyByCompanyId(Long companyId): Company{
    return companyMapper.selectCompanyByCompanyId(companyId);
}
```

CompanyServiceImpl::selectCompanyList(Company company):List<Company>

前置条件: 公司信息(company)不能为空。

后置条件: 系统从公司信息表(CompanyInfoList)中选择对应的公司信息组成的列表, 并将其返回。

```
CompanyServiceImpl::selectCompanyList(Company company):List<Company> {
    return companyMapper.selectCompanyList(company);
}
```

CompanyServiceImpl::insertCompany(Company company): int

前置条件: 公司信息(company)不能为空。

后置条件: 系统向公司信息表(CompanyInfoList)中新增一条公司信息, 并将结果返回。

```
CompanyServiceImpl::insertCompany(Company company): int {
    company.setCreateTime(DateUtils.getNowDate());
    return companyMapper.insertCompany(company);
}
```

```
}
```

CompanyServiceImpl::updateCompany(Company company) :int

前置条件：公司信息(company)不能为空。

后置条件：系统在公司信息表(CompanyInfoList)中修改一条公司信息，并将结果返回。

CompanyServiceImpl::updateCompany(Company company) :int {

```
    company.setUpdateTime(DateUtils.getNowDate());
```

```
    return companyMapper.updateCompany(company);
```

```
}
```

CompanyServiceImpl::deleteCompanyByCompanyIds(Long[] companyIds): int

前置条件：公司编号列表(companyIds)不能为空。

后置条件：系统在公司信息表(CompanyInfoList)中删除所有编号在传入参数中的数据，并将操作结果返回。

CompanyServiceImpl::deleteCompanyByCompanyIds(Long[] companyIds): int {

```
    return companyMapper.deleteCompanyByCompanyIds(companyIds);
```

```
}
```

CompanyServiceImpl::deleteCompanyByCompanyId(Long companyId): int

前置条件：公司编号(companyId)不为空。

后置条件：系统在公司信息表(CompanyInfoList)中删除一条数据，并返回操作结果。

CompanyServiceImpl::deleteCompanyByCompanyId(Long companyId): int {

```
    return companyMapper.deleteCompanyByCompanyId(companyId);
```

```
}
```

CompanyServiceImpl::companySettlement(Long companyId):int

前置条件：公司编号(companyId)不为空。

后置条件：系统从公司信息表(CompanyInfoList)中修改指定公司的状态为“已入驻”并返回操作结果。

CompanyServiceImpl::companySettlement(Long companyId):int {

```
    Company company ← companyMapper.selectCompanyByCompanyId(companyId);
```

```
    if(company.getStatus() == 0){
```

```
        company.setStatus(1);
```

```

        return companyMapper.updateCompany(company);
    }else {
        return 0;
    }
}

```

2. TicketTypeServiceImpl

TicketTypeServiceImpl
- ticketTypeMapper: TicketTypeMapper - companyMapper: CompanyMapper - deptMapper: SysDeptMapper
+ selectTicketTypeById(Long): TicketType + selectTicketTypeList(TicketType): List<TicketType> + insertTicketType(TicketType): int + updateTicketType(TicketType): int + deleteTicketTypeByIds(Long[]): int + deleteTicketTypeById(Long): int - createTypePermission(Long): Long

图 3 - 15 TicketTypeServiceImpl 类

方法介绍及伪码

TicketTypeServiceImpl::selectTicketTypeById(Long typeId): TicketType

前置条件：服务单类型编号(typeId)不可为空。

后置条件：系统从服务单类型信息表(TicketTypeInfoList)中查找对应编号的服务单类型并将其返回。

```

TicketTypeServiceImpl::selectTicketTypeById(Long typeId): TicketType{
    return ticketTypeMapper.selectTicketTypeById(typeId);
}

```

TicketTypeServiceImpl::selectTicketTypeList(TicketType ticketType): List<TicketType>

前置条件：服务单类型信息(ticketType)不为空。

后置条件：系统在服务单类型信息表(TicketTypeInfoList)中查找所有符合条件的服务单类型，将其组织成列表形式并返回。

```

TicketTypeServiceImpl::selectTicketTypeList(TicketType ticketType): List<TicketType> {
    return ticketTypeMapper.selectTicketTypeList(ticketType);
}

```

```
}
```

`TicketTypeServiceImpl::insertTicketType(TicketType ticketType):int`

前置条件：服务单类型信息(ticketType)不为空。

后置条件：系统向服务单类型信息表中插入一条新的数据，并返回操作结果。

```
TicketTypeServiceImpl::insertTicketType(TicketType ticketType):int {
    temp ← new TicketType();
    temp.setTypeNames(ticketType.getTypeNames());
    temp.setTypeCompany(ticketType.getTypeCompany());
    if (0 != ticketTypeMapper.selectTicketTypeList(temp).size()) {
        throw new ServiceException("类型名称不能重复!");
    }
    deptId ← createTypePermission(ticketType.getTypeCompany());
    ticketType.setDeptId(deptId);
    ticketType.setCreateTime(DateUtils.getNowDate());
    dept ← deptMapper.selectDeptById(deptId);
    dept.setDeptName(ticketType.getTypeNames());
    deptMapper.updateDept(dept);
    return ticketTypeMapper.insertTicketType(ticketType);
}
```

`TicketTypeServiceImpl::updateTicketType(TicketType ticketType):int`

前置条件：服务单类型信息(ticketType)不能为空。

后置条件：系统在服务单类型信息表(ticketTypeInfoList)中修改指定的数据，并返回操作结果。

```
TicketTypeServiceImpl::updateTicketType(TicketType ticketType):int {
    ticketType.setUpdateTime(DateUtils.getNowDate());
    return ticketTypeMapper.updateTicketType(ticketType);
}
```

`TicketTypeServiceImpl::deleteTicketTypeByTypeIds(Long[] typeIds):int`

前置条件：服务单类型编号列表(typeIds)不能为空。

后置条件：系统在服务单类型信息表(ticketTypeInfoList)中删除所有编号存在于服务单类型

编号列表(typeIds)中的数据，并返回操作结果。

```
TicketTypeServiceImpl::deleteTicketTypeByTypeIds(Long[] typeIds):int {
    return ticketTypeMapper.deleteTicketTypeByTypeIds(typeIds);
}
```

TicketTypeServiceImpl::deleteTicketTypeById(Long typeId):int

前置条件：服务单类型编号(typeId)不能为空。

后置条件：系统在服务单类型信息表(ticketTypeInfoList)中删除对应编号的服务单类型(TicketType)，并返回操作结果。

```
TicketTypeServiceImpl::deleteTicketTypeById(Long typeId):int {
    return ticketTypeMapper.deleteTicketTypeById(typeId);
}
```

TicketTypeServiceImpl::createTypePermission(Long companyId): Long

前置条件：公司编号(companyId)不能为空。

后置条件：系统在部门信息表(DeptInfoList)中插入一条数据，并返回新增的部门编号。

```
TicketTypeServiceImpl::createTypePermission(Long companyId): Long {
    Company company ← companyMapper.selectCompanyByCompanyId(companyId);
    targetDeptId ← company.getDeptId();
    tempDept ← deptMapper.selectDeptById(company.getDeptId());
    if (!UserConstants.DEPT_NORMAL.equals(tempDept.getStatus())) {
        throw new ServiceException("类型停用，不允许新增");
    }
    tempDept.setAncestors(tempDept.getAncestors() + "," + tempDept.getDeptId());
    tempDept.setParentId(tempDept.getDeptId());
    tempDept.setDeptName(IdUtils.fastSimpleUUID().substring(25));
    tempDept.setDeptId(null);
    tempDept.setCreateBy(SecurityUtils.getLoginUser().getUsername());
    tempDept.setCreateTime(DateUtils.getNowDate());
    System.out.println("[DEBUG]" + tempDept.toString());
    deptMapper.insertDept(tempDept);
    return deptMapper.selectLastInsertId();
}
```

3. TicketService

TicketServiceImpl
- ticketMapper: TicketMapper - ticketTypeMapper: TicketTypeMapper - companyMapper: CompanyMapper - deptMapper: SysDeptMapper
+ selectTicketByTicketId(Long): Ticket + selectTicketList(Ticket): List<Ticket> + insertTicket(Ticket): int + updateTicket(Ticket): int + deleteTicketByTicketIds(Long[]): int + deleteTicketByTicketId(Long): int + applyTicket(Ticket): int - ticketLimitCheck(Company): boolean

图 3 - 16 TicketService 类

构造方法及伪码：

TicketService::selectTicketByTicketId(Long ticketId): Ticket

前置条件：服务单编号(ticketId)不能为空。

后置条件：系统在服务单信息表(TicketInfoList)中选择对应编号的服务单信息并返回。

```
TicketService::selectTicketByTicketId(Long ticketId): Ticket {
    return ticketMapper.selectTicketByTicketId(ticketId);
}
```

TicketService::selectTicketList(Ticket ticket): List<Ticket>

前置条件：服务单信息(ticket)不能为空。

后置条件：系统在服务单信息表(TicketInfoList)中选择满足限定条件的服务单信息，将其组成列表并返回。

```
TicketService::selectTicketList(Ticket ticket): List<Ticket>{
    return ticketMapper.selectTicketList(ticket);
}
```

TicketService::insertTicket(Ticket ticket): int

前置条件：服务单信息(ticket)不能为空。

后置条件：系统向服务单信息表(TicketInfoList)中新增一条数据，并返回操作结果。

```
TicketService::insertTicket(Ticket ticket): int {
    ticket.setCreateTime(DateUtils.getNowDate());
    return ticketMapper.insertTicket(ticket);
}
```

TicketService::updateTicket(Ticket ticket): int

前置条件：服务单信息不能为空(ticket)。

后置条件：系统向服务端信息表(TicketInfoList)中修改一条指定的数据，并返回操作结果。

```
TicketService::updateTicket(Ticket ticket): int {
    ticket.setUpdateTime(DateUtils.getNowDate());
    return ticketMapper.updateTicket(ticket);
}
```

TicketService::deleteTicketByTicketIds(Long[] ticketIds): int

前置条件：服务单编号列表(ticketIds)不能为空。

后置条件：系统从服务单信息表(TicketInfoList)中查找所有编号存在于服务单编号列表(ticketIds)中的数据，将其删除并返回操作结果。

```
TicketService::deleteTicketByTicketIds(Long[] ticketIds): int {
    return ticketMapper.deleteTicketByTicketIds(ticketIds);
}
```

TicketService::deleteTicketByTicketId(Long ticketId): int

前置条件：服务单编号 (ticketId)不能为空。

后置条件：系统从服务单信息表(TicketInfoList)中查找对应编号的数据，将其删除并返回操作结果。

```
TicketService::deleteTicketByTicketId(Long ticketId) {
    return ticketMapper.deleteTicketByTicketId(ticketId);
}
```


TicketService::applyTicket(Ticket ticket): int

前置条件：服务单信息(ticket)不能为空。

后置条件：系统尝试在服务单信息表(TicketInfoList)中新增一条数据，并返回操作结果。

```
TicketService::applyTicket(Ticket ticket): int {
    Company company ←
companyMapper.selectCompanyById(ticket.getCompanyId());
    TicketType type ←
ticketTypeMapper.selectTicketTypeById(ticket.getTicketType());
    if (!ticketLimitCheck(company)) {
        throw new ServiceException("申请数量过多，请关闭一些服务单后再尝试");
    }
    ticket.setDeptId(type.getDeptId());
    ticket.setStatus(1);
    return ticketMapper.insertTicket(ticket);
}
```

TicketService::ticketLimitCheck(Company company): Bool

前置条件：公司信息(company)不能为空。

后置条件：系统检查当前用户是在给定公司否申请过多服务单，并返回结果。

```
TicketService::ticketLimitCheck(Company company): Bool {
    if (null == company.getTicketLimit()) {
        company.setTicketLimit(2);
    }
    Ticket temp ← new Ticket();
    temp.setUserId(SecurityUtils.getLoginUser().getUserId());
    temp.setCompanyId(company.getCompanyId());
    Long count ← ticketMapper.selectTicketCountByUserAndCompany(temp);
    if (count > company.getTicketLimit()) {
        return false;
    }
    return true;
}
```

3.5.3 数据访问层

数据访问层使用 **MyBatis** 进行实现；在本段中，所有的数据映射使用 **XML** 文件来描述后端与数据库间进行的数据交互，具体的代码实现将由 **Mybatis** 框架自动配置。

数据访问层中的数据权限控制采用 **AOP** 方法，通过权限控制注解实现。在实现过程中，**SQL** 代码片段储存在 **params.dataScope** 中。

DataScopeAspect
+ doBefore(JoinPoint, DataScope): void
handleDataScope(JoinPoint, DataScope): void
+ dataScopeFilter(JoinPoint, SysUser, String, String): void
- clearDataScope(JoinPoint): void

图 3 - 17 DataScopeAspect 类

1. CompanyMapper

CompanyMapper
+ selectCompanyByCompanyId(Long): Company
+ selectCompanyList(Company): List<Company>
+ insertCompanyList(Company): int
+ updateCompanyList(Company): int
+ deleteCompanyByCompanyId(Long): int
+ deleteCompanyByCompanyIds(Long[]): int

图 3 - 18 CompanyMapper 接口

接口对应 **XML** 及 **SQL** 语句：

SQL 结果映射标签：

```
<resultMap type="Company" id="CompanyResult">
    <result property="companyId" column="company_id"/>
    <result property="deptId" column="dept_id"/>
    <result property="companyName" column="company_name"/>
    <result property="companyAddress" column="company_address"/>
    <result property="email" column="email"/>
    <result property="foundTime" column="found_time"/>
    <result property="certificate" column="certificate"/>
</resultMap>
```

```

<result property="additionalInfo" column="additional_info"/>
<result property="applyDate" column="apply_date"/>
<result property="companyMaster" column="company_master"/>
<result property="ticketLimit" column="ticket_limit"/>
<result property="status" column="status"/>
<result property="createBy" column="create_by"/>
<result property="createTime" column="create_time"/>
<result property="updateBy" column="update_by"/>
<result property="updateTime" column="update_time"/>
</resultMap>

```

SQL 基础查询语句:

```

<sql id="selectCompanyVo">
    SELECT
        company_id, dept_id, company_name, company_address,
        email, found_time, certificate, additional_info, apply_date,
        company_master, ticket_limit, status, create_by,
        create_time, update_by, update_time
    FROM
        fea_company
</sql>

```

CompanyMapper::selectCompanyList(Company company): List<Company>

前置条件: 公司信息(company)不能为空。

后置条件: 系统从公司信息表(CompanyInfoList)中查找所有符合条件的数据, 将其组成列表并返回。

```

<select
    id="selectCompanyList"
    parameterType="Company"
    resultMap="CompanyResult"
>
    <include refid="selectCompanyVo" />
    <where>

```

```

<if test="companyName != null  and companyName != ""
    >and company_name like concat('%', #{companyName}, '%')
</if>
<if test="companyAddress != null  and companyAddress != ""
    >and company_address like concat('%', #{companyAddress}, '%')
</if>
<if test="email != null  and email != "">and email = #{email}</if>
<if test="foundTime != null ">and found_time = #{foundTime}</if>
<if test="certificate != null  and certificate != ""
    >and certificate = #{certificate}</if>
>
<if test="additionalInfo != null  and additionalInfo != ""
    >and additional_info = #{additionalInfo}</if>
>
<if test="applyDate != null ">and apply_date = #{applyDate}</if>
<if test="companyMaster != null "
    >and company_master = #{companyMaster}</if>
>
<if test="ticketLimit != null ">and ticket_limit = #{ticketLimit}</if>
<if test="status != null ">and status = #{status}</if>
${params.dataScope}
</where>
</select>

```

CompanyMapper::selectCompanyByCompanyId: Company

前置条件：公司编号(companyId)不能为空。

后置条件：系统从公司信息表(CompanyInfoList)中选择对应的公司信息并将其返回。

```

<select
    id="selectCompanyByCompanyId"
    parameterType="Long"
    resultMap="CompanyResult">
    <include refid="selectCompanyVo"/>

```

```

        where company_id = #{companyId}
    </select>

```

CompanyMapper::insertCompany(Company company): int

前置条件：公司信息(company)不能为空。

后置条件：系统向公司信息表(CompanyInfoList)中新增一条公司信息，并将结果返回。

```

<insert
    id="insertCompany"
    parameterType="Company"
    useGeneratedKeys="true"
    keyProperty="companyId">
    insert into fea_company
    <trim prefix="(" suffix=")" suffixOverrides=",">
        <if test="companyName != null and companyName != "">company_name,</if>
        <if test="companyAddress != null">company_address,</if>
        <if test="email != null">email,</if>
        <if test="foundTime != null">found_time,</if>
        <if test="certificate != null">certificate,</if>
        <if test="additionalInfo != null">additional_info,</if>
        <if test="applyDate != null">apply_date,</if>
        <if test="companyMaster != null">company_master,</if>
        <if test="ticketLimit != null">ticket_limit,</if>
        <if test="status != null">status,</if>
        <if test="createBy != null and createBy != "">create_by,</if>
        <if test="createTime != null">create_time,</if>
        <if test="updateBy != null">update_by,</if>
        <if test="updateTime != null">update_time,</if>
    </trim>
    <trim prefix="values (" suffix=")" suffixOverrides=",">
        <if test="companyName != null
            and companyName != "">
            #{companyName},
        </if>

```

```

        <if test="companyAddress != null">#{companyAddress},</if>
        <if test="email != null">#{email},</if>
        <if test="foundTime != null">#{foundTime},</if>
        <if test="certificate != null">#{certificate},</if>
        <if test="additionalInfo != null">#{additionalInfo},</if>
        <if test="applyDate != null">#{applyDate},</if>
        <if test="companyMaster != null">#{companyMaster},</if>
        <if test="ticketLimit != null">#{ticketLimit},</if>
        <if test="status != null">#{status},</if>
        <if test="createBy != null and createBy != "">#{createBy},</if>
        <if test="createTime != null">#{createTime},</if>
        <if test="updateBy != null">#{updateBy},</if>
        <if test="updateTime != null">#{updateTime},</if>
    </trim>
</insert>

```

CompanyMapper::updateCompany(Company company) :int

前置条件：公司信息(company)不能为空。

后置条件：系统在公司信息表(CompanyInfoList)中修改一条公司信息，并将结果返回。

```

<update id="updateCompany" parameterType="Company">
    update fea_company
    <trim prefix="SET" suffixOverrides=",">
        <if test="companyName != null and companyName != "">company_name =
#{companyName},</if>
        <if test="companyAddress != null">company_address =
#{companyAddress},</if>
        <if test="email != null">email = #{email},</if>
        <if test="foundTime != null">found_time = #{foundTime},</if>
        <if test="certificate != null">certificate = #{certificate},</if>
        <if test="additionalInfo != null">additional_info = #{additionalInfo},</if>
        <if test="applyDate != null">apply_date = #{applyDate},</if>
        <if test="companyMaster != null">company_master = #{companyMaster},</if>
        <if test="ticketLimit != null">ticket_limit = #{ticketLimit},</if>
    </trim>

```

```

        <if test="status != null">status = #{status},</if>
        <if test="createBy != null and createBy != "">create_by = #{createBy},</if>
        <if test="createTime != null">create_time = #{createTime},</if>
        <if test="updateBy != null">update_by = #{updateBy},</if>
        <if test="updateTime != null">update_time = #{updateTime},</if>
    </trim>
    where company_id = #{companyId}
</update>

```

CompanyMapper::deleteCompanyByCompanyIds(Long[] companyIds): int

前置条件：公司编号列表(companyIds)不能为空。

后置条件：系统在公司信息表(CompanyInfoList)中删除所有编号在传入参数中的数据，并将操作结果返回。

```

<delete id="deleteCompanyByCompanyIds" parameterType="String">
    delete from fea_company where company_id in
    <foreach item="companyId" collection="array" open="(" separator="," close=")">
        #{companyId}
    </foreach>
</delete>

```

CompanyMapper::deleteCompanyByCompanyId(Long companyId): int

前置条件：公司编号(companyId)不为空。

后置条件：系统在公司信息表(CompanyInfoList)中删除一条数据，并返回操作结果。

```

<delete id="deleteCompanyByCompanyId" parameterType="Long">
    delete from fea_company where company_id = #{companyId}
</delete>

```

2. TicketTypeMapper

TicketTypeMapper
+ selectTicketTypeByCompanyId(Long): TicketType + selectTicketTypeByTypeId(TicketType): List<TicketType> + insertTicketType(TicketType): int + updateTicketTypeList(TicketType): int + deleteTicketTypeByTypeId(Long): int + deleteTicketTypeByTypeIds(Long[]): int

图 3 - 19 TicketTypeMapper 接口

接口对应 XML 及 SQL 语句:

SQL 结果映射标签:

```
<resultMap type="TicketType" id="TicketTypeResult">
    <result property="typeId" column="type_id"/>
    <result property="typeCompany" column="type_company"/>
    <result property="deptId" column="dept_id"/>
    <result property="typeName" column="type_name"/>
    <result property="description" column="description"/>
    <result property="createBy" column="create_by"/>
    <result property="createTime" column="create_time"/>
    <result property="updateBy" column="update_by"/>
    <result property="updateTime" column="update_time"/>
</resultMap>
```

SQL 基础查询语句:

```
<sql id="selectTicketTypeVo">
    SELECT
        type_id,
        type_company,
        dept_id,
        type_name,
        description,
        create_by,
        create_time,
        update_by,
```



```

        update_time
    FROM
        fea_type
</sql>

```

TicketTypeMapper::selectTicketTypeList(TicketType ticketType): List<TicketType>

前置条件：服务单类型信息(ticketType)不为空。

后置条件：系统在服务单类型信息表(TicketTypeInfoList)中查找所有符合条件的服务单类型，将其组织成列表形式并返回。

```

<select id="selectTicketTypeList" parameterType="TicketType"
        resultMap="TicketTypeResult">
    <include refid="selectTicketTypeVo"/>
    <where>
        <if test="typeCompany != null ">and type_company = #{typeCompany}
        </if>
        <if test="deptId != null ">and dept_id = #{deptId}</if>
        <if test="typeName != null and typeName != "">and type_name like
            concat('%', #{typeName}, '%')
        </if>
        <if test="description != null and description != "">and
            description like concat('%', #{description},
            '%')
        </if>
        ${params.dataScope}
    </where>
</select>

```

CompanyMapper::selectTicketTypeById(Long typeId): TicketType

前置条件：服务单类型编号(typeId)不可为空。

后置条件：系统从服务单类型信息表(TicketTypeInfoList)中查找对应编号的服务单类型并将其返回。

```

<select id="selectTicketTypeById" parameterType="Long"
        resultMap="TicketTypeResult">

```

```
<include refid="selectTicketTypeVo"/>
where type_id = #{typeId}
</select>
```

CompanyMapper::insertTicketType(TicketType ticketType):int

前置条件：服务单类型信息(ticketType)不为空。

后置条件：系统向服务单类型信息表中插入一条新的数据，并返回操作结果。

```
<insert id="insertTicketType" parameterType="TicketType"
      useGeneratedKeys="true" keyProperty="typeId">
  insert into fea_type
  <trim prefix="(" suffix=")" suffixOverrides=",">
    <if test="typeCompany != null">type_company,</if>
    <if test="deptId != null">dept_id,</if>
    <if test="typeName != null">type_name,</if>
    <if test="description != null">description,</if>
    <if test="createBy != null">create_by,</if>
    <if test="createTime != null">create_time,</if>
    <if test="updateBy != null">update_by,</if>
    <if test="updateTime != null">update_time,</if>
  </trim>
  <trim prefix="values (" suffix=")" suffixOverrides=",">
    <if test="typeCompany != null">#{typeCompany},</if>
    <if test="deptId != null">#{deptId},</if>
    <if test="typeName != null">#{typeName},</if>
    <if test="description != null">#{description},</if>
    <if test="createBy != null">#{createBy},</if>
    <if test="createTime != null">#{createTime},</if>
    <if test="updateBy != null">#{updateBy},</if>
    <if test="updateTime != null">#{updateTime},</if>
  </trim>
</insert>
```

CompanyMapper::updateTicketType(TicketType ticketType):int

前置条件：服务单类型信息(ticketType)不能为空。

后置条件：系统在服务单类型信息表(ticketTypeInfoList)中修改指定的数据，并返回操作结果。

```
<update id="updateTicketType" parameterType="TicketType">
    update fea_type
    <trim prefix="SET" suffixOverrides=",">
        <if test="typeCompany != null">type_company = #{typeCompany},</if>
        <if test="deptId != null">dept_id = #{deptId},</if>
        <if test="typeName != null">type_name = #{typeName},</if>
        <if test="description != null">description = #{description},</if>
        <if test="createBy != null">create_by = #{createBy},</if>
        <if test="createTime != null">create_time = #{createTime},</if>
        <if test="updateBy != null">update_by = #{updateBy},</if>
        <if test="updateTime != null">update_time = #{updateTime},</if>
    </trim>
    where type_id = #{typeId}
</update>
```

CompanyMapper::deleteTicketTypeByTypeIds(Long[] typeIds):int

前置条件：服务单类型编号列表(typeIds)不能为空。

后置条件：系统在服务单类型信息表(ticketTypeInfoList)中删除所有编号存在于服务单类型编号列表(typeIds)中的数据，并返回操作结果。

```
<delete id="deleteTicketTypeById" parameterType="Long">
    delete from fea_type where type_id = #{typeId}
</delete>
```

CompanyMapper::deleteTicketTypeById(Long typeId):int

前置条件：服务单类型编号(typeId)不能为空。

后置条件：系统在服务单类型信息表(ticketTypeInfoList)中删除对应编号的服务单类型(TicketType)，并返回操作结果。

```
<delete id="deleteTicketTypeByTypeIds" parameterType="String">
    delete from fea_type where type_id in
    <foreach item="typeId" collection="array" open="(" separator=","
```

```

        close="")">
        #{typeId}
    </foreach>
</delete>

```

3.5.4 领域模型层

本层主要介绍 BaseEntity 类。该类为本系统中所有实体类的基类，用于存储本系统中所有实体共有的属性。

BaseEntity
- createTime: Date - params: Map<String, Object> - remark: String - createBy: String - updateBy: String - updateTime: Date - searchValue: String

图 3 - 20 BaseEntity 类

该类没有构造方法或其他方法的代码。

3.6 数据视图

3.6.1 逻辑设计

物理模型设计图

1. CompanyInfoList(企业信息表)

表 3 - 2 CompanyInfoList 物理模型

名称	数据类型	长度限制	缺省值	可缺省	可空
company_id	int	20	NA	否	否
company_name	varchar	50	NA	否	否
company_address	varchar	100	NA	否	是
email	varchar	50	NA	否	是
found_time	datetime	NA	当前时间	是	否

certificate	varchar	NA	NA	否	否
additional_info	varchar	NA	NA	否	是
apply_date	datetime	NA	当前时间	是	否
ticket_limit	int	4	2	是	否
status	int	4	0	是	否

2. UserInfoList(用户信息表)

表 3 - 3 UserInfoList 物理模型

名称	数据类型	长度限制	缺省值	可缺省	可空
user_id	int	20	NA	否	否
dept_id	int	20	NA	否	是
user_name	varchar	30	NA	否	是
nick_name	varchar	30	NA	否	是
user_type	varchar	2	00	是	是
email	varchar	50	空字符串	是	是
phonenummer	varchar	11	空字符串	是	是
sex	char	1	0	是	是
avatar	varchar	100	空字符串	是	是
password	varchar	100	空字符串	是	是
status	char	1	0	是	是
del_flag	char	1	0	是	是
login_ip	varchar	128	空字符串	是	是
login_date	datetime	NA	当前时间	是	是
create_by	varchar	64	空字符串	是	是
create_time	datetime	NA	当前时间	是	是
update_by	varchar	64	空字符串	是	是

update_time	datetime	NA	当前时间	是	是
remark	varchar	500	空字符串	是	是

3. TicketTypeInfoList

表 3 - 4 TicketTypeInfoList 物理模型

名称	数据类型	长度限制	缺省值	可缺省	可空
type_id	int	20	NA	否	否
dept_id	int	20	NA	否	是
type_company	int	20	NA	否	是
type_name	varchar	32	空字符串	是	是
description	varchar	64	空字符串	是	是

3.6.2 数据访问机制

客户、企业和管理员通过账号密码登录，账号密码需要通过 MySQL 匹配验证，根据不同角色权限进行数据操纵。

第四章 详细设计

4.1 用户交互层

4.1.1 企业入驻审批

```
@RestController
@RequestMapping("/feature/company")
public class CompanyController extends BaseController {
    @Autowired
    private ICompanyService companyService;

    @Autowired
    private ISysUserService userService;

    /**
     * 新增公司管理
     */
    @PreAuthorize("@ss.hasPermi('feature:company:add')")
    @Log(title = "公司管理", businessType = BusinessType.INSERT)
    @PostMapping
    public AjaxResult add(@RequestBody Company company) {
        // 初始化公司数据
        // 公司所有者为当前登录用户
        company.setCompanyMaster(getLoginUser().getUserId());
        // 公司申请提交日期为当前日期
        company.setApplyDate(DateUtils.getNowDate());
        // 初始状态为等待审批
        company.setStatus(0);
        return toAjax(companyService.insertCompany(company));
    }
}

import store from "@store";
import {
```

```

    getCompanyByMaster,
    addCompany,
    updateCompany,
  } from "@/api/feature/company";
export default {
  name: "SettlementForm",
  data() {
    return {
      // 表单参数
      form: {},
      // 表单校验
      rules: {
        email: [
          { required: true, message: "电子邮箱不能为空", trigger: "blur" },
        ],
        companyName: [
          { required: true, message: "公司名称不能为空", trigger: "blur" },
        ],
        certificate: [
          { required: true, message: "请提交营业执照", trigger: "blur" },
        ],
        foundTime: [
          { required: true, message: "成立时间不能为空", trigger: "blur" },
        ],
      },
    };
  },
  props: {},
  methods: {
    queryCompany() {
      getCompanyByMaster(this.$store.state.user.userId).then((response) => {
        console.log(response);
        if (response.data) {

```



```

        this.form = response.data;
        this.resetForm("form");
    } else {
        this.reset();
    }
});
},
handleSubmit() {
    this.$refs["form"].validate((valid) => {
        if (valid) {
            if (this.form.companyId != null) {
                updateCompany(this.form).then((response) => {
                    this.$modal.msgSuccess("公司信息修改成功");
                    this.queryCompany();
                });
            } else {
                addCompany(this.form).then((response) => {
                    this.$modal.msgSuccess("申请提交成功");
                    this.queryCompany();
                });
            }
        }
    });
},
reset() {
    this.form = {
        companyId: null,
        companyName: null,
        companyAddress: null,
        email: null,
        foundTime: null,
        certificate: null,
        additionalInfo: null,
    }
}

```

```

        applyDate: null,
        companyMaster: null,
        ticketLimit: null,
        status: null,
        createBy: null,
        createTime: null,
        updateBy: null,
        updateTime: null,
    };
    this.resetForm("form");
},
},
created() {
    this.queryCompany();
},
};

```

4.1.2 申请服务单

```

/**
 * 服务单 Controller
 *
 * @author isghrina
 * @date 2023-04-18
 */
@RestController
@RequestMapping("/feature/ticket")
public class TicketController extends BaseController {
    @Autowired
    private ITicketService ticketService;

    @Autowired
    private ICompanyService companyService;
}

```

```

/**
 * 申请服务单
 */
@PreAuthorize("@ss.hasPermi('feature:ticket:add')")
@Log(title = "服务单", businessType = BusinessType.INSERT)
@PostMapping("/apply")
public AjaxResult apply(@RequestBody Ticket ticket) {
    return toAjax(ticketService.applyTicket(ticket));
}
}

import { getCompany } from "@api/feature/company";
import { listTicketType } from "@api/feature/ticketType";
import {
    listSubscribe,
    subscribeCompany,
    unsubscribeCompany,
} from "@api/feature/subscribe";
import { applyTicket } from "@api/feature/ticket";

export default {
    name: "ApplyTicketForm",
    props: {
        companyId: undefined,
    },
    data() {
        return {
            form: {
                ticketTitle: undefined,
                ticketType: undefined,
                status: 0,
            },
            rules: {},

```

```

        typeList: [],
    };
},
methods: {
    fetchTicketType() {
        let param = {
            companyId: this.companyId,
        };
        listTicketType(param).then((response) => {
            console.log("fetching type...");
            this.typeList = response.rows;
        });
    },
    reset() {
        this.form = {
            ticketTitle: undefined,
            ticketType: undefined,
        };
    },
    ticketLimitOverflowError(){
        this.$modal.msgError("申请数量达到上限，请关闭后再尝试申请。");
    },
    ticketApplyError(){
        this.$modal.msgError("出现错误，请联系管理员");
    },
    submitSuccessModal(){
        this.$modal.msgSuccess("服务单申请成功");
    },
    handleSubmit() {
        this.form.userId = this.$store.state.user.userId;
        this.form.companyId = this.companyId;
        applyTicket(this.form).then(
            (response) => {

```

```

        this.$modal.msgSucess("已成功申请服务单");
    },
    (reject) => {
        this.$modal.msgError("出现错误， 请检查");
    }
);
},
},
created() {
    if (!this.companyId) {
        return;
    }
    getCompany(this.companyId).then((response) => {
        this.detail = response.data;
        this.loading = false;
    });
    this.fetchTicketType();
},
};

```

第五章 软件测试

5.1 测试对象和概要

5.1.1 测试方法与工具

表 5 - 1 测试方法与工具列出了本系统测试使用的方法与工具。

表 5 - 1 测试方法与工具

测试内容	测试方法	测试工具	备注
单元测试	白盒；手工	Junit	
功能	黑盒；手工；回归；工具自动	QTP	
功能	黑盒；工具自动	Selenium	
性能	黑盒；手工；回归；工具自动	JMeter	

5.1.2 测试环境与配置

本系统的功能测试与性能测试使用相同配置，通过虚拟机、虚拟环境隔离测试数据。

1. 数据库服务器配置

测试环境的数据库使用腾讯云提供的云数据库。其主要信息如表 5-2 数据库服务器配置所示：

表 5-2 数据库服务器配置

机器名 (IP)	CPU	内存	软件环境 (数据库版本)
cdb-63qhwpqg.cd.tencentcdb.com	1 核	1000MB 内存	MySQL5.7

2. 应用服务器与客户端配置

应用服务器与客户端使用本地硬件，其主要配置信息如表 5-3 应用服务器与客户端配置所示：

表 5-3 应用服务器与客户端配置

机器名 (IP)	CPU	内存	软件环境 (操作系统、应用软件)
127.0.0.1	Intel Core I7-9750h	16GB 2667MHz	Windows 10 Chrome 113.0.5672.93 (Official Build) (64-bit)

5.1.3 单元测试用例设计

本段中将重点介绍申请服务单功能的测试情况。

1. TicketService::insertTicket(TicketType ticketType)

待测试代码如下：

@Override

@Transactional

```
public int insertTicketType(TicketType ticketType) {
    TicketType temp = new TicketType();
    temp.setTypeNames(ticketType.getTypeNames());
    temp.setTypeCompany(ticketType.getTypeCompany());
    if (0 != ticketTypeMapper.selectTicketTypeList(temp).size()) {
        throw new ServiceException("类型名称不能重复！");
    }
    Long deptId = createTypePermission(ticketType.getTypeCompany());
}
```

```

        ticketType.setDeptId(deptId);
        ticketType.setCreateTime(DateUtils.getNowDate());
        SysDept dept = deptMapper.selectDeptById(deptId);
        dept.setDeptName(ticketType.getTypeName());
        deptMapper.updateDept(dept);
        return ticketTypeMapper.insertTicketType(ticketType);
    }

    private Long createTypePermission(Long companyId) {
        Long targetDeptId;
        Company company = companyMapper.selectCompanyByCompanyId(companyId);
        targetDeptId = company.getDeptId();
        SysDept tempDept = deptMapper.selectDeptById(company.getDeptId());
        if (!UserConstants.DEPT_NORMAL.equals(tempDept.getStatus())) {
            throw new ServiceException("类型停用，不允许新增");
        }
        tempDept.setAncestors(tempDept.getAncestors() + "," + tempDept.getDeptId());
        tempDept.setParentId(tempDept.getDeptId());
        tempDept.setDeptName(IdUtils.fastSimpleUUID().substring(25));
        tempDept.setDeptId(null);
        tempDept.setCreateBy(SecurityUtils.getLoginUser().getUsername());
        tempDept.setCreateTime(DateUtils.getNowDate());
        deptMapper.insertDept(tempDept);
        return deptMapper.selectLastInsertId();
    }
}

```

分析以上代码，可得程序流程图如图 5 - 1 TicketTypeServiceImpl::insertTicketType 流程图所示：

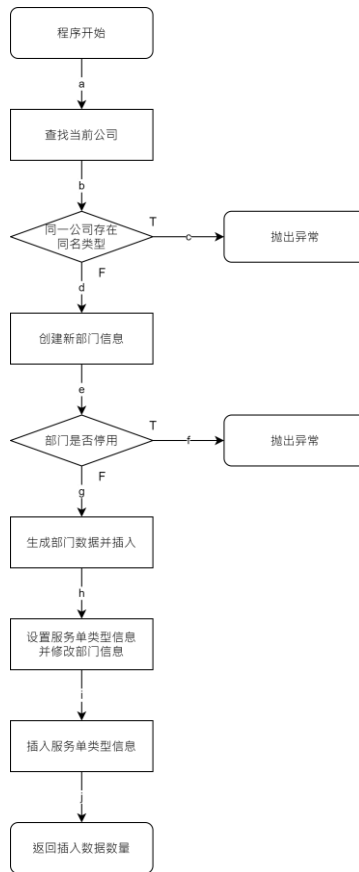


图 5 - 1 TicketTypeServiceImpl::insertTicketType 流程图

对于该方法，由于逻辑较为简单，可以直接使用判定/条件覆盖法编写测试用例。该功能的测试用例如表 5 - 4 insertTicketType 的判定覆盖测试用例：

表 5 - 4 insertTicketType 的判定覆盖测试用例

用例编号	输入	期望输出	路径
T-BC-TT1	TicketType{typeName:"测试类型 1",typeCompany:1}，输入两次	error	abc
T-BC-TT2	TicketType{typeName:"测试类型 2",typeCompany:1} (进行测试前，编号为 1 的公司已停用)	error	abdef
T-BC-TT3	TicketType{ typeName:"测试类型 3", typeCompany:1}	success	abdeghij

5.1.4 功能测试用例设计

1. 申请入驻

由于本系统对于数据范围限制主要由前端实现，且验证过程已被框架封装，此处仅描述系统中使用的验证条件、测试用例及设计方法；由于代码中多处使用字符串格式校验，

难以对输入情况进行穷举，故采用等价类划分法能够比较有效地设计测试用例。

1) 分析代码，划分等价类

对于如下验证条件：

```
rules: {
  email: [
    {
      required: true,
      message: "电子邮箱不能为空",
      trigger: "blur",
    },
    {
      type: "email",
      message: "电子邮箱格式不正确",
      trigger: "blur",
    },
  ],
  companyName: [
    { required: true, message: "公司名称不能为空", trigger: "blur" },
  ],
  certificate: [
    { required: true, message: "请提交营业执照", trigger: "blur" },
  ],
  foundTime: [
    { required: true, message: "成立时间不能为空", trigger: "blur" },
  ],
  ticketLimit: [
    {
      type: "number",
      min: 0,
      max: 2147483647,
      message: "请填写 0-2147483647 之间的数字",
      transform: num=>{
        return Number(num);
      }
    }
  ]
}
```

```

    }
  },
],
},
foundTimePickerOptions:{
  disabledDate: this.dateFilter,
},

```

可以分析得到等价类：

表 5 - 5 companyName 的等价类

编号	输入条件	所属类别
1	长度 0-20，满足正则表达式的字符串	有效等价类
2	长度超过 20 的字符串	无效等价类
3	不满足正则表达式的字符串	无效等价类
4	空字符串	无效等价类

注：companyName 的正则表达式为 $^{[0-9a-zA-Z\backslash u4e00-\backslash u9fa5]}^*\$$

表 5 - 6 companyAddress 等价类

编号	输入条件	所属类别
5	长度 0-40，满足正则表达式的字符串	有效等价类
6	长度超过 40 的字符串	无效等价类
7	不满足正则表达式的字符串	无效等价类
8	空字符串	无效等价类

注：companyAddress 的正则表达式为 $^{[0-9a-zA-Z\backslash u4e00-\backslash u9fa5]}^*\$$

表 5 - 7 email 的等价类

编号	输入条件	所属类别
9	长度 0-20，满足正则表达式的字符串	有效等价类
10	长度超过 20 的字符串	无效等价类

11	不满足正则表达式的字符串	无效等价类
12	空字符串	无效等价类

注：email 的正则表达式为 \w[-\w.]*@([A-Za-z0-9][-A-Za-z0-9]+\.)+[A-Za-z]+

表 5 - 8 foundTime 的等价类

编号	输入条件	所属类别
13	在当前日期之前的日期	有效等价类
14	当前日期之后的日期	无效等价类

据上述等价类划分，得到等价类如表 5-9 企业入驻申请功能等价类划分：

表 5-9 企业入驻申请功能等价类划分

输入		有效等价类	编号	无效等价类	编号
变量名	类型				
companyName	字符串	满足表 2-3 企业入驻申请数据项所述数据项描述的数据	1	companyName 为空	2
				companyName 不满足正则表达式	3
companyAddress	字符串			companyAddress 为空	4
				companyAddress 不满足正则表达式	5
email	字符串			email 为空	6
				email 不符合格式要求	7
foundTime	日期			foundTime 为空	8
				foundTime 是未来日期	9
ticketLimit	数字			ticketLimit 为负	10
				ticketLimit 大于 2147483647	11

2) 根据划分的等价类设计测试用例如表 5-10 申请入驻功能测试用例（空值已使用斜体标出）：

表 5-10 申请入驻功能测试用例

用例编号	输入					期望输出	覆盖等价类
	companyName	companyAddress	email	foundTime	ticketLimit		
1	测试公司	测试地址	t@t.com	2022-01-01	2	success	1
2	<i>空字符串</i>	测试地址	t@t.com	2022-01-01	2	error	2
3	1-2+3	测试地址	t@t.com	2022-01-01	2	error	3
4	测试公司	<i>空字符串</i>	t@t.com	2022-01-01	2	error	4
5	测试公司	1.234	t@t.com	2022-01-01	2	error	5
6	测试公司	测试地址	<i>空字符串</i>	2022-01-01	2	error	6

7	测试公司	测试地址	t@t	2022-01-01	2	error	7
8	测试公司	测试地址	t@t.com	NULL	2	error	8
9	测试公司	测试地址	t@t.com	9105-01-01	2	error	9
10	测试公司	测试地址	t@t.com	2022-01-01	-1	error	10
11	测试公司	测试地址	t@t.com	2022-01-01	4e10	error	11

5.1.5 性能测试用例设计

表 5 - 11 企业入驻申请性能测试用例设计

功能	当在线用户达到高峰时，查询信息正常。保证 200 个以内用户可以同时访问系统，能够正常查询数据。					
目的	测试系统 200 个以内的用户同时在线时能否正常查询信息。					
方法	采用 JMeter 的录制工具录制一个信息查询过程，然后利用其完成测试，要监视数据库服务器和 Web 服务器的性能。其中查询的信息符合表 2-3 企业入驻申请数据项。					
并发用户数与事务执行情况						
并发用户数	Sample	Average	Median	Min	Max	Error
100						
.....						
150						
200						

5.1.6 整体策略

本项目的特点：

1. 参与的测试人员不是第一次接触本系统

2. 系统已经做过一些测试，并且已经在运行
3. 相对于项目要做的事情来说，时间进度非常紧（要建立一个基本完善的测试规范、要设计整套测试用例和执行一轮完整的测试）

4. 本次项目测试的只对系统进行一轮测试

根据以上特点，制定本项目的测试过程策略如下：

1. 以 80/20 原理为指导。

尽量做到在有限的时间里发现尽可能多的缺陷（尤其是严重缺陷）

2. 测试计划与需求制定、用例设计同步进行
3. 必须制定测试需求。

通过确定要测试的内容和各自的优先级、重要性，使测试设计工作更有目的性，在需求的指导下设计出更多更有效的用例。

4. 逐步完善测试用例库。

测试用例库的建设是一个不断完善的过程，我们要在有限的时间里，先设计出一整套的测试用例，重要的部分用例需要设计得完善一些，一般部分的则指出测试的要点，在以后的测试工作中再不断去完善测试用例库。

5. 测试过程要受到控制。

根据事先定义的测试执行顺序进行测试，并填写测试记录表，保证测试过程是受控的。

6. 确定重点。

测试重点放在各子系统的功能实现上，问题较多的服务单管理模块和公司管理模块则是重中之重。

7. 不测试实现技术。

本次测试不对依赖的产品和开源项目代码实现的核心技术（数据映射、数据验证等）进行测试验证。

测试技术

1. 本项目采用白盒测试和黑盒测试技术。
2. 本项目测试过程中采用 Junit、QTP、JMeter 等测试工具。

依据标准

本次测试中测试文档、测试用例的编写、具体的执行测试以及测试中各项资源的分配和估算，都是以本系统的需求规格说明书、概要设计说明书和详细设计说明书为标准，软件的执行以系统逻辑设计构架为依据。

5.1.7 测试范围

制定本次项目测试范围的依据为：

1. 各个最小单元
2. 各模块所包含的功能

要测试的模块或子系统:

表 5 - 12 要测试的模块或功能

测试内容	测试范围
功能测试	<p>服务单模块</p> <p>评价模块模块</p> <p>投诉模块</p> <p>企业管理模块</p> <p>消息模块</p> <p>问题类型模块</p> <p>客服模块</p> <p>回复模板模块</p>
性能测试	<p>一、 模块</p> <p>对两个模块进行性能测试:</p> <p>1、消息模块</p> <p>2、企业管理模块</p> <p>二、 数据量</p> <p>以 MySQL 数据库中存在十万条记录为标准, 测试如下性能数据:</p> <p>1、新数据入库性能</p> <p>2、修改数据性能</p> <p>3、模块各功能性能</p> <p>三、 硬件配置</p> <p>测试腾讯云服务器给定配置下的性能</p>

不测试的模块或功能

表 5 - 13 不测试的模块或功能

模块或功能	说明
使用了 Mybatis 的一般功能	使用逻辑简单的外部代码，为保证项目工期，不做测试
用户信息模块	该模块是开源框架中原有模块，不做测试
部门管理模块	该模块是开源框架中原有模块，不做测试

5.2 测试内容和执行情况

5.2.1 功能测试

表 5 - 14 功能测试概要情况

模块名称	开始时间	结束时间	用例数	用例通过数	问题数	用例通过率
			个	个	个	%
公司管理 模块	2023-05-01	2023-05-01	22	22	0	100

5.2.2 性能测试

表 5 - 15 性能测试概要情况

功能	当在线用户达到高峰时，查询信息正常。保证 200 个以内用户可以同时访问系统，能够正常查询数据。					
目的	测试系统 200 个以内的用户同时在线时能否正常查询信息。					
方法	采用 JMeter 的录制工具录制一个信息查询过程，然后利用其完成测试，要监视数据库服务器和 Web 服务器的性能。其中查询的信息符合表 2-3 企业入驻申请数据项。					
并发用户数与事务执行情况						
并发用户数	Sample	Average	Median	Min	Max	Error
100		1034ms	1145ms	732ms	2043ms	0

.....						
150		1255ms	1298ms	728ms	1973ms	0
200		1543ms	1432ms	812ms	5329ms	0

5.3 覆盖分析

表 5 - 16 测试覆盖率

模块名称	用例个数	执行数	各模块测试覆盖率(%)	未/漏测分析和原因
公司管理 模块	22	22	100	无

5.4 缺陷统计与分析

在测试过程中未发现缺陷。

5.5 测试结论

系统已通过测试，达到需求规定水准。

参考文献

- [1] MyBatis.org. 2009. mybatis. mybatis.org. [OL] MyBatis.org, 2023.03.13. [2023.05.01] <https://mybatis.org/mybatis-3/>.
- [2] Open Source Initiative. 2012. Open Source Initiative OSI - The MIT License:Licensing. Open Source Initiative. [OL] Open Source Initiative, 2012.09.25. [2023.05.01.] <https://opensource.org/license/mit/>. OSI.
- [3] RuoYi. 2023. RuoYi-Vue: 基于 SpringBoot, Spring Security, JWT, Vue & Element 的前后端分离权限管理系统, 同时提供了 Vue3 的版本. Gitee - 企业级 DevOps 研发效能平台. [OL] 2023.01.01. [2023.05.01] https://gitee.com/y_project/RuoYi-Vue.
- [4] ruoyi. 2018. 介绍 | RuoYi. RuoYi 在线文档. [OL] 2018. [2023.05.01] <http://doc.ruoyi.vip/ruoyi/document/htsc.html#%E6%B3%A8%E8%A7%A3%E5%8F%82%E6%95%B0%E8%AF%B4%E6%98%8E-4>.
- [5] 焦子铉. 基于心流理论的当代大学生内生驱动教学研究[J]. 徐州工程学院学报(社会科学版), 2022, 37(03): 99-108.
- [6] 涂哲皓. D 保险公司客户服务流程优化研究[D]. 电子科技大学, 2022. DOI:10.27005/d.cnki.gdzku.2022.001064.
- [7] 张宇欣. “互联网+”燃气客户服务优化与升级[D]. 北京建筑大学, 2018.
- [8] 高源. WHDX 政企客户服务流程优化研究[D]. 华中科技大学, 2021. DOI:10.27157/d.cnki.ghzku.2021.007109.
- [9] 张智尧, 涂畅, 蔡嘉荣. 多模态情绪识别技术在银行客户服务的实现和应用[J]. 金融科技时代, 2022, No.325(09): 33-42.

致谢

四载春秋，回首再看。象牙塔里，朝阳逐秋水。再登临，已是时光荏苒，流年不再。临别六月间，是我的指导老师何凯在定稿之前给予了多次耐心的指导，才使论文能够顺利完成。虽万千言语，难表一丝谢意。

有人说，我们生活在名为无知的小岛上，周围是无边无际黑色的，名为知识的海洋；而上帝对于人最仁慈的施舍，便是不能够了解其中的知识哪怕一星半点——对真理的追求令人疯狂，令航船在呼啸的暴风中迷失方向。

即使如此，有着明灯的引航，小船也能够突破骤雨狂风，向深海行去。有时，人也能和灯塔一样。有的人，甚至比灯塔照的更亮、更远。在难忘的三万五千零四十个小时中，为我指明前路的姜平、毛艳艳、宋英杰、唐焕玲、肖进杰、徐猛、张斌等老师(姓名按照姓氏字母排序，排名不分先后)表示诚挚的谢意。是你们给予我无私帮助，才让我在这无边的大海中不至迷途。只要有光的指引，行驶在夜雾里的人，也能抵抗海浪的拉扯，找到正确的航向。

征蓬远去上苍穹，苍穹孤烟别落日；日落飞锦绣长河，天地壮我行色。即将踏出校门走向社会的时点，对旧日的沉湎亦是一种对未来的辜负。某虽不才，愿将此间事理篆于心间，未来行走在这片大地之时，这些似春水的回忆也会称为温暖的港湾。

山东工商学院
本科生毕业论文（设计）开题报告
（2023 届）

学生姓名 _____

院（部） _____

专 业 _____

指导教师（签名） _____

年 月 日

本科毕业论文开题报告表

学生姓名		所属学院		专业	
班级				学号	
指导教师	姓名		职称		
开 题 报 告 内 容	题目（中、英文）：				
	选题的目的和意义：				
	（可根据情况加页）				
	研究方向、思路和重点：				
	（可根据情况加页）				
	文献资料（包括与本课题相关的国内、外研究现状的资料）：				
（可根据情况加页）					
进度安排：					
指导教师 意见	<div style="text-align: right;"> 指导教师签名： 年 月 日 </div>				
本科 毕业 论文 指导 小组 意见	<div style="text-align: right;"> 年 月 日 </div>				

山东工商学院

本科生毕业论文指导记录

姓名: 年(班)级:

专业: 学 号:

题目：

日期	指导方式(面谈、电话、微信、QQ、E-mail 等)	指导内容	指导意见

(此表仅供参考, 可附页)

指导教师（签名）：_____