蚂蚁热帮(Ant Help)

——订单、消息、活动管理

摘　要

通过绪论、需求分析、概要设计、详细设计与实现、测试和总结等各章对订单、消息、活动管理模块进行了详细的分析与设计。针对各个部分的需要，绘制了相关的图示和表格，用以达到最优的项目设计文档说明效果。

本系统是针对第三方生活服务平台的管理分析而设计。主要包括用户权限管理、订单管理和社区管理三大模块。并采用了 O2O 式的电子商务模式来开拓生活服务领域的发展，力求改善人们的同城生活服务，提供多元化的专人速递速成服务，快速响应，解决急需等任何个人生活需求。

本系统作为一个生活服务类平台，用户可以在该平台上发布任何合法的个人需求，在支付成功后可以对其他用户可见，进而得到其他用户的进一步响应—接单和抢单。抢单成功后，接单者于线下完成发布者的需求，完成后将完成情况反馈给发布者，发布者在确认接单者完成无误后，该订单交易结束。

本系统采用的 C#进行编码，同时采用微软公司的 Visual studio 2015 和微软公司的 Microsoft SQL Server 2014 开发环境进行开发。系统的结构组成主要有客户端、web 服务和数据库访问层以及数据库持久化层。是系统层次明确，逻辑清晰，便于编码和后期维护。

关键词：生活服务　　O2O　　个人需求　　C#

The Design and Implementation of AntHelp Life Service System

——Order , Message and Activity Management

ABSTRACT

This paper through the introduction, demand analysis, outline design, detailed design and implementation, testing and summary of the statistical process management module for the statistical process of the user , order and community. For each part of the need to draw the relevant diagrams and tables, in order to achieve the best results of the project design documentation.

The system is designed for the management and analysis of the third party life service platform. It mainly includes three modules: user rights management, order management and community management. And the use of e-commerce model to explore the development of the type 020 service areas of life, and strive to improve people's city life service, providing a wide range of special courier service express, fast response, etc. any personal life needs urgent.

This system is a life service platform, users can publish any personal legal demand on the platform, in the payment after the success can be visible to other users, and then further response to other users of the orders and grab one. After the success of a single order, the purchaser completes the publisher's requirements under the line. After completion, the feedback will be issued to the publisher, and the publisher will conclude the order transaction after the confirmation of the order is completed.

This system uses the C# to carry on the code, simultaneously uses the Microsoft Corp Studio Visual 2015 and the Microsoft Corp SQL Server Microsoft 2014 development environment to carry on the development. The structure of the system consists of client, web service, database access layer and database persistence layer. System level is clear, clear logic, easy to encode and post maintenance.

Key Words: The service of life O2O Personal demand C#

目　　录

第一章 绪论

1.1 项目背景与现状

1.1.1 项目背景简介

随着团购网站和移动端应用软件的迅速发展，020 模式的电子商务正在被人们认识和接受，020 的发展改变了传统的电子商务模式与格局。020 的概念非常广泛，只要产业链中既可涉及到线上，又可涉及到线下，就可以统称为 020。网上购买服务，我们司空见惯的是打车、家政服务等，但在以个人为单位进行其他方面的生活服务则少很多。因此国内外在相关方面的研究还比较缺乏，也不是非常成熟和完善。国内外学者对消费者消费行为的研究大约有十年多的时间，其研究方向和内容也较为广泛。

在人们越来越注重生活质量的当下，个人需求更是加快了互联网对生活服务类产品的开发和投入。国内电子商务和团购网站的全面普及，以及近两年来网约车软件以及共享单车等移动端应用软件的迅速发展，都体现了 020 电子商务模式在生活服务领域发展的重要性。这种电子商务模式的优势也显而易见，对于商家和企业而言，它能够带来真实的客户流和数据统计，能够更好的实现线上虚拟经济与线下实体经济的完美融合；对于客户而言，020 也提供了更丰富、更全面的商家服务信息，能够让用户更加方便快捷的体验到相应的产品和服务；而对于服务提供商来说，这种电子商务模式可以为他们带来更加大规模、高粘度的用户，帮助他们获得商家资源和充沛的现金流。所以 020 实现的“三方共赢”的商务模式。所以，020 生活服务以强大的生命力和商业价值，正在发展成为电子商务领域的新的主干发展线。

1.1.2 当前国内外现状

近年来，随着 Internet 的迅速崛起，我们从最初的信息发布阶段，极速的迈入了电子商务阶段，成为了我们这个商业信息社会的神经系统。随着互联网信息碎片以及云计算技术的愈发成熟，互联网营销模式顺势而出，020 式的生活服务已经成为了人们生活中不可忽视的一部分。在国内，随着网络的飞速发展，网络营销已经广泛的映射到我们的日常生活中。020 的概念从诞生到爆发，仅仅用了不到两年的时间。2010 年王兴推出的美团网更是拉动了国内 020 的迅速发展。如今美团网作为一家本地服务类的电子商务企业，以超低折扣的优质服务和广泛销售和推广渠道，使得消费者与商家也越来越信赖于它。王兴本人指出，美团网与其他转型传统 B2C 的团购网站大不相同，在支付方面，中国团购更习惯于第三方支付。2016 年，美团已成为国内最受大众喜欢的团购网站。陈浩然在《美团网：服务电商的逆袭》一文中提到中国的团购网站不是简单模仿国外 Groupon 的模式，而是一种改进与创新的模式。极大展

示了 020 模式下的生活服务的发展潜力。

Martin Pasquier 在《中国为何掀起了 020 革命》(2015.5)中提到了在中国掀起 020 热潮的三大重要因素，一是高质量和高可靠性的代码对现实生活和线上产品的桥接；二是人们对语音服务和远程服务信任的快速增长；三是手机支付，手机支付是促进 020 模式在中国取得巨大实施的关键因素。

国内的生活服务平台短时间内的迅速崛起，极大展示了 020 模式下的生活服务的发展潜力。从餐饮订购的饿了么，到团购服务的美团，到便利出行的滴滴，无一不采用 020 的电子商务模式以提供三方生活服务平台，提供更便捷、有效的生活方式，提高生活质量。

1.2 选题意义与系统主要工作

1.2.1 选题目的及意义

随着社会的发展与进步，人们更加注重生活质量，这样的趋势注定互联网也更加投入和开发生活服务类的产品。蚂蚁热帮是致力于更快捷、更有效的提高生活质量的一个第三方生活服务平台。短短 5 年内，国内的生活服务平台如雨后春笋般迅速崛起，从餐饮订购的饿了么，到团购服务的美团，到便利出行的滴滴，无一不采用 020 的电子商务模式。蚂蚁热帮同时也采用了这种 020 模式，力求改善人们的同城生活服务，提供多元化的专人速递速成服务，快速响应，解决急需等任何个人化的生活需求，例如：想找人帮忙购物、取件、送件，或是找代驾、帮排队等。

蚂蚁热帮的意义在于提供更方便、更便捷、更有效的生活方式，提高办事效率，改变人们的生活节奏，因而提高生活质量。同时，也拓宽人们的日常生活圈，提升人与人之间的交际和信任。

其应用范围有：代购代帮、代理销售、家政服务、宠物之家、汽车服务、休闲服务、二手处理等等一系列生活服务。

1.2.2 本系统主要工作

做好三方服务平台对提升商业竞争力和个人生活质量有重要意义。平台发布过程可以解决个人所需，提供方便快捷、高响应的发单接单。平台管理可以使用户更方便的查看个人的发单接单详情等信息。

1.系统相关用户角色以及对各种角色的用户信息进行管理。

2.在用户的注册登录功能中，添加系统的 Token 认证，保障用户信息的安全。

3.系统各项功能的访问权限的管理，对不同的用户开放不同的权限进行操作。

4.发布者可以进行下单请求，并且由系统分发订单信息。

5. 其他用户抢单接单进行对订单的后续流程的操作。不同的用户角色对订单有不同的操作策略。

6. 管理员权限可以对所有用户、订单、消息及投诉等信息进行查看和管理。

7. 管理员可以对用户发送系统消息。

8. 用户可以对系统提出意见或反馈，也可以对其他用户非法操作提出的投诉信息。

9. 系统管理员发起优惠活动，用户可以在规定期间内进行活动的参与。

10. 社区模块类似于微信朋友圈等社交媒体的沟通平台，用户可以在平台内进行交流分享。

第二章 系统开发环境及相关技术简介

2.1 系统开发环境简介

2.1.1 Microsoft Visual Studio 2015 简介

Visual Studio 作为一个基于组件的软件开发工具，有强大的集成环境，可用于构建功能强大、性能出众的各类应用程序。Visual Studio 2015 于 2014 年 11 月 13 日正式投入市场使用，它可以帮助各类开发人员打造跨平台的应用程序，从 Windows 到 Linux,甚至 iOS 和 Android。Visual Studio 2015 发布以来具有以下新增功能：

1.与云集成，内置工具可让所有的 NET Core、Azure 应用程序、服务、Docker 容器等全面集成。无缝体验让你感觉如同在 Azure 数据中心工作一样。

2.有效协作，直接管理任意提供程序托管的团队项目，包括 Visual Studio Team Services、Team Foundation Server 或 GitHub。或者，使用新的“打开任意文件夹”功能，无需通过正式项目或解决方案即可快速打开并处理几乎所有代码文件。

3.交付优质移动应用，借助 Xamarin 的高级调试、分析工具以及单元测试生成功能，与以往相比你可以更快、更轻松地构建、连接和调整适用于 Android、iOS 和 Windows 的本机移动应用。你还可以选择使用 Apache Cordova 开发移动应用，或构建 C++跨平台库。

4.提高语言水平，Visual Studio 继续加强对最新编程语言功能的支持。无论使用 C#、Visual Basic、C++、TypeScript、F# 还是使用第三方语言（例如 JavaScript），在整个开发体验中你都能获得一流的功能支持。

5.性能得以优化，Visual Studio 包含了对日常使用的核心功能的大量性能改进。你还会发现在整个开发生命周期中，启动速度显著加快，内存占用大大降低，并且响应能力明显提高。

2.1.2 Microsoft SQL Server 2014 简介

Microsoft SQL Server 2014 在延续了历史版本的各项优势以外，更是添加了诸多的新特性。这些新的特性使我们的使用更加的方便快捷，提高了我们的开发效率。下面为 Microsoft SQL Server 2014 的新特性：

1.全程加密技术(Always Encrypted)，全程加密技术(Always Encrypted)支持在 SQL Server 中保持数据加密，只有调用 SQL Server 的应用才能访问加密数据。

2.动态数据屏蔽(Dynamic Data Masking)，利用动态数据屏蔽功能，你可以将 SQL Server 数据库表中待加密数据列混淆，那些未授权用户看不到这部分数据。利用动态数据屏蔽功能，你还可以定义数据的混淆方式。

3.JSON 支持，在 SQL Server 2008 中，你现在可以在应用和 SQL Server 数据库引擎之间用 JSON 格式交互[1]。微软公司在 SQL Server 中增加了对 JSON 的支持，可以解析 JSON 格式数据然后以关系格式存储。

4.SQL SERVER 支持 R 语言，SQL Server 支持 R 语言处理以后，数据科学家们可以直接利用现有的 R 代码并在 SQL Server 数据库引擎上运行用[2]。这样我们就不用为了执行 R 语言处理数据而把 SQL Server 数据导出来处理。该功能把 R 语言处理带给了数据。

5.Query Store，如果你经常使用执行计划，你就会喜欢新版的 Query Store 功能。在 2016 之前的版本中，你可以使用动态管理试图(DMV)来查看现有执行计划。

6.行级安全(Row Level Security)，SQL 数据库引擎具备了行级安全特性以后，就可以根据 SQL Server 登录权限限制对行数据的访问[3]。限制行是通过内联表值函数过滤谓词定义实现的。安全策略将确保过滤器谓词获取每次“SELECT”或者“DELETE”操作的执行。

2.2 系统实现技术简介

2.2.1 AngularJS 2 简介

在一个互联网迅速膨胀的时代，“前端”在短短的几年内已经发展成为每一个程序员耳熟能详的单词，“PHP 虽然是最好的语言”，但它不是最流行的语言。近年来，Javascript 凭借它自身的优势，受到了广泛开发者的喜爱和青睐。因此，前端圈子近年来蓬勃发展，各种社区和前端框架也应用而生。如：在数据可视化方面，Processing 被带入到了 Web 领域产生了 Processing.js,还出现了 D3.js,通过使用 HTML、SVG 和 CSS 把数据鲜活形象的展现出来；在移动应用方面，Cordova 和 React Native 实现了一次开发，多平台发布；在服务端，V8 的性能将 JavaScript 带到了一个新的高度，Node.js 的诞生，迎来了 JS 的全栈时代[4]。

AngularJS 在前端膨胀发展的当下，成为了最受瞩目和广泛应用的前端框架之一，它作为一个 MVVM(Model-View-ViewModel)框架，将视图和模型层分离，使得软件的 UI 层更加细节化、可定制化，有低耦合、可重用性、独立开发、可测试等特性。

AngularJS 自身的优势也显而易见：

1. 模板功能强大丰富，自带丰富的 Angular 指令，并且是声明式的；
2. 包含模板、数据双向绑定、路由、模块化、服务、过滤器、依赖注入等特性功能；
3. 自定义的 Directive,比 jQuery 插件使用更加灵活，重用性高；

4. Ng 模块化大胆的引入了 Java 的一些东西，比如依赖注入等，加强了代码的可复用性；
5. 支持单元测试和 e2e-testing。

Angular2 在 Angular1 的基础上，形成了一个统一的、系统性的架构，在语言层面上，它使用 Typescript，在架构层面上，它使用 Component 的概念统摄全局，大大地简化了概念模型并降低了学习的成本[5]。它还提供了高度集成化的开发工具 angular/cli，它以 webpack 为内核，内置了 Karma/Jasmine/Protractor/TS compiler。CLI 目前已经发展的相当完善，并推出正式版，可在正式项目中使用。通过 CLI，开发者可以创建、开发、测试、构建项目，而不需要额外的依赖。

Angular 是一个面向未来的科技，它要求浏览器支持 ES6+，所以在开发过程中需要一个工具链进行解析编译等一系列自动化操作。如图 2.1 所示为 ES6 工具链图：

图2.1　ES6 工具链图

Fig.2.1　ES6 chain of tools

Systemjs(通用模块加载器)，支持 AMD、CommonJS、ES6 等各种格式的 JS 模块加载；

ES6-module-loader(ES6 模块加载器)，systemjs 会自动加载这个模块；

Traceur(ES6 转码器)，将 ES6 代码转换为当前浏览器支持的 ES5 代码。

Angular2 中，通过重新设计和引入新技术，摆脱了旧的技术框架束缚，在开发上达到了高整合性、高复用的组件化的开发[6]。在用户体验方面，对速度也有了极致追求，将之前的“多轮检查、直到稳定”策略改成了“一轮检查、直接稳定”策略。

2.2.2 Typescript 简介

Typescript 是微软开发的自由的，开源的编程语言，它是 Javascript 的强类型版本，语言中包含了后端开发人员熟悉的静态类型和基于类的面向对象编程特点，使得前后端的开发人员在语法的使用上更近了一步。TypeScript 是 JavaScript 类型的超集，它在编译期去掉类型和特有的语法，生成纯粹的 JavaScript 代码，最终在浏览器中运行[7]。TypeScript 可以在任何浏览器、任何计算机和任何操作系统上运行，并且是开源的。TypeScript 扩展了 JavaScript 的语法，所以任何现有的 JavaScript 程序可以不加改变的在 TypeScript 下工作。TypeScript 是为大型应用之开发而设计，而编译时它产生 JavaScript 以确保兼容性。

由于 Typescript 是 ECMAScript6 的超集，因此它具有以下几大特性：

1. 类型批注和编译时类型检查
2. 支持类：继承了可选的类型批注支持的 ES6 的类
3. 支持接口
4. 支持模块
5. 支持 lambda 函数
6. Generator

除了它较之 Javascript 的特性外，Typescript 有它先天的优势：

1. 高度兼容原生脚本语法；
2. 对语法的破坏性较少，容易上手；
3. 它讲究基于接口的强类型，因此也适用于原本的服务器开发者；
4. 提供了大量编辑器的集成，现已有大量库的.ts 元文件可用。

2.2.3 Git 简介

Git 是一款免费、开源的分布式版本控制系统，可以有效、高速的处理从小到大的项目版本管理。版本控制是一种记录若干文件内容变化，以便将来查阅特定版本修订情况的系统，在没有自动化的版本控制之前，开发者们习惯复制整个项目目录的方式来保存不同的版本，这种本地版本控制方式经常会导致开发者混淆所在的工作目录，一旦弄错文件丢了数据就没有办法撤销恢复。

为了让在不同系统上的开发者协同工作，集中化的版本控制系统(Centralized Version Control Systems,简称 CVCS)应运而生。比较常见的有 CVS，Subversion 以及 Perforce 等，都有一个单一的集中管理的服务器，来保存所有的文件修订版本[8]。这需要协同工作的人员连接到该服务器，并取出最新的文件或提交更新。这种版本控制很显而易见的缺点就是中央服务器的单点故障，会造成开发过程中莫大的损失。

不久后分布式版本控制系统的面世又开启了开发者的一场新旅程，较常见的有 Git,Mercurial,Bazaar 以及 Darcs 等等[9]。这种版本控制使得客户端并不只是提取最新版本

的文件快照，而是把原始的代码仓库完整的镜像下来。所以，在协同开发的过程中，万一服务器发生故障，事后也都可以用镜像来进行本地仓库的恢复，所以得到了广大开发者的青睐。

Git 作为如今最常用的协同开发版本控制系统之一，具有以下功能：

1. 从服务器上克隆完整的 Git 仓库到单机上，包括代码和版本信息等；
2. 在自己的机器上根据不同的开发目的，创建分支，修改代码；
3. 在单机上自己创建的分支上提交代码；
4. 在单机上合并分支；
5. 将服务器上最新版的代码拉取下来，跟自己的主分支合并；
6. 生成补丁(patch)，把补丁发送给主开发者；
7. 主开发者反馈，合理解决冲突；

使用 Git 进行版本控制的开发模式还具有以下优势：

1. 公共服务器压力和数据量都不会太大；
2. 采用分布式版本库，不需服务器端软件支持；
3. 适合分布式开发，强调个体；
4. 速度快、灵活；
5. 任意两个开发者之间都可以很容易的解决冲突；
6. 离线工作；
7. 协作开发，高效、便捷。

因而，在本项目中，为了方便小组成员之间的协作开发，我们采用了 Git 的分布式版本控制。

2.3 本章小结

本章对本课题的国内外发展情况和选题意义进行了调查及分析，明确了项目的背景信息，并对本系统所涉及的核心技术进行说明。本项目在开发过程中采用前后端的基本方式，后端主要使用以 C#语言为根基的个人研发框架作为主开发模式，前端采用了最新较为流行的 Web 应用框架——基于 Typescript 语言的 Angular2 进行开发。本章节也对该项目的开发环境进行了简要的设定，项目环境主要使用了 Visual studio 2015 和微软公司的 Microsoft SQL Server 2014。

第三章 系统定位与需求分析

3.1 背景简介

本题目基于近年来 020 商务模式的迅速发展，致力于更快捷、更有效的提高生活质量的第三方生活服务平台。力求改善人们的同城生活质量，提供多元化的专人速递速成服务，快速响应，解决急需等任何个人化的生活需求，例如：想找人帮忙购物、取件、送件，或是找代驾、帮排队、失物招领、宠物照看、二手处理等等。

3.2 网站设计目标

3.2.1 总目标

在平台上整合所有个人或团体大大小小的需求，让个人需求能够在平台被快速响应，无论是想找人帮忙购物、取件、送件，还是找代驾、帮排队、失物招领、宠物照看、二手处理等等，在平台上都能找到最快、最佳、最可靠的帮达人，最终达到客户的双方所需。

3.2.2 功能目标

1. 系统相关用户角色以及对各种角色的用户信息进行管理。
2. 在用户的注册登录功能中，添加系统的 Token 认证，保障用户信息的安全。
3. 系统各项功能的访问权限的管理，对不同的用户开放不同的权限进行操作。
4. 发布者可以进行下单请求，并且由系统分发订单信息。
5. 其他用户抢单接单进行对订单的后续流程的操作。不同的用户角色对订单有不同的操作策略。
6. 管理员权限可以对所有用户、订单、消息及投诉等信息进行查看和管理。
7. 管理员可以对用户发送系统消息。
8. 用户可以对系统提出意见或反馈，也可以对其他用户非法操作提出的投诉信息。
9. 系统管理员发起优惠活动，用户可以在规定期间内进行活动的参与。
10. 社区模块类似于微信朋友圈等社交媒体的沟通平台，用户可以在平台内进行交流分享。

3.2.3 性能目标

1. 友好性

实现网站的快速访问，减少用户等待时间，优化服务性能，提供便捷高响应的用户体验；

2. 安全性

项目所有数据传输都经过加密，同时个人信息访问需权限认证，增加安全系数，防止用户个人基本信息被窃取；

3. 数据库优化

数据库设计需合理，做到高效快速查询。

3.3 系统的需求结构

如图 3.1 所示，为系统需求结构图。

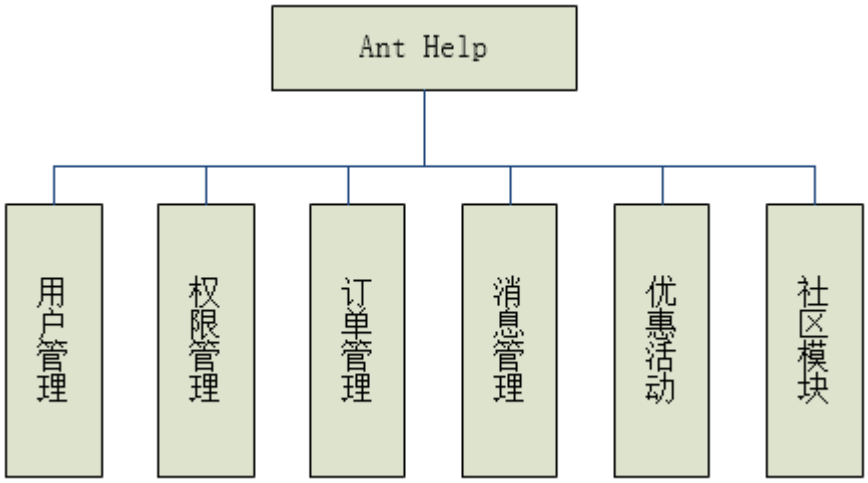

图3.1　系统需求结构图

Fig.3.1　System requirement structor diagram

需求结构说明：

1. 用户管理

在用户管理模块中，设计系统相关用户的角色以及对各种角色的用户信息进行管理。在对用户信息进行管理的同时还涉及到了用户的注册登录等功能，该模块有涉及到系统的 Token 认证相关概念。

2. 权限管理

该模块主要用于系统的各项功能的访问权限的管理，对不同的用户开放不同的权限进行操作，使得系统的稳定性和安全性得到了提高，加强了管理员对人员的管理以及系统的维护。

3. 订单管理

订单管理模块用户接收用户的下单请求，并且由系统分发订单信息，其他用户抢单接单进行对订单的后续流程的操作。不同的用户角色对订单有不同的操作策略。

4. 消息管理

消息管理模块主要用于管理员对用户发送系统消息，或者是用户对系统的意见或建议，以及用户对其他用户的投诉信息，可以向系统投诉信箱发送信息，系统管理员登录后可以查看用户发送来的投诉信息，并进行对投诉信息的处理。

5. 优惠活动

优惠活动模块，由系统发起的优惠，用户可以在规定期间内进行活动的参与。

6. 社区模块

社区模块类似于微信朋友圈等社交媒体的沟通平台，用户可以在平台内进行交流分享。

3.4 用例分析

3.4.1 参与者

在订单管理模块，参与者主要是发布者和接收者；在消息管理和活动管理模块，参与者主要是系统管理员。

3.4.2 用例图

涉及到发布者和接收者的用例图如图 3.2 所示。

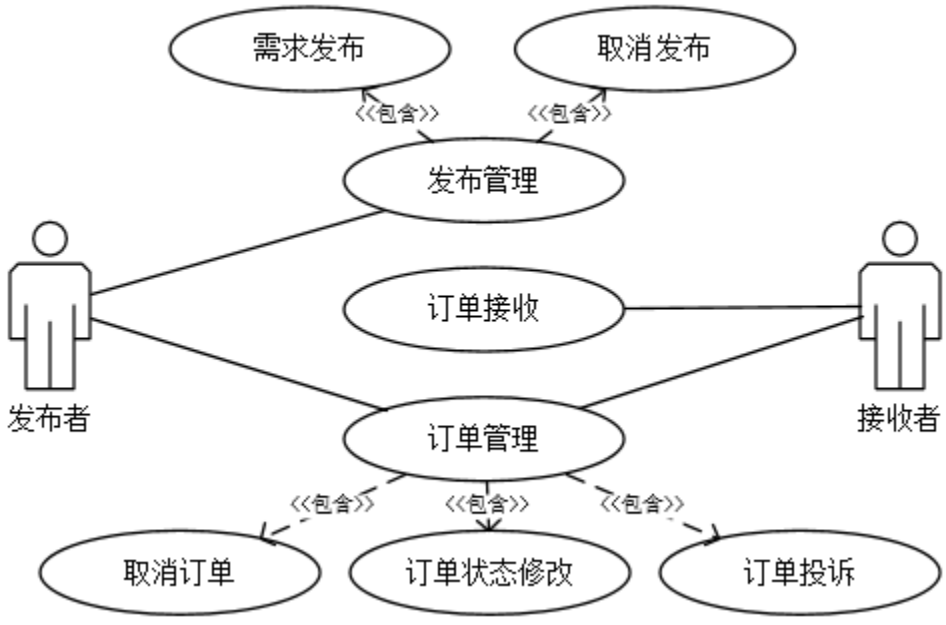

图3.2　发布者和接收者用例图

Fig.3.2　Publisher and Receiver use case diagram

关键用例描述如表3.1与表3.2所示：

表3.1 发布管理用例描述

Table3.1 Publish management use case description

描述项	说明
用例名称	发布管理
用例描述	发布者发布个人需求生成新订单，并且可以进行查询以及删除
参与者	发布者
前置条件	已经登录系统的用户，且身份验证合格
后置条件	如果这个用例成功，在系统中建立新的订单记录
基本操作流程	1，发布者输入发布信息 2，系统验证数据合法性 3，发布者处理异常数据 4，需求发布完成
可选操作流程	1，取消相关操作 2，数据输入不合法，操作失败
被泛化的用例	无
被包含的用例	需求发布，取消发布
被拓展的用例	无

表3.2 订单管理用例描述

Table3.2 Order management use case description

描述项	说明
用例名称	订单管理
用例描述	发布者对已发布订单、接收者对已接收订单的管理
参与者	发布者、接收者
前置条件	已经登录系统的用户，且身份验证合格
后置条件	如果这个用例成功，订单据相应操作而更新
基本操作流程	1，发布者/接收者登录系统 2，订单相关操作 3，系统进行合法性验证 4，操作成功
可选操作流程	1，取消相关操作 2，操作不合法时，操作失败
被泛化的用例	无
被包含的用例	取消订单、订单状态修改、订单投诉

被拓展的用例	无

涉及到系统管理员的用例图如图 3.3 所示。

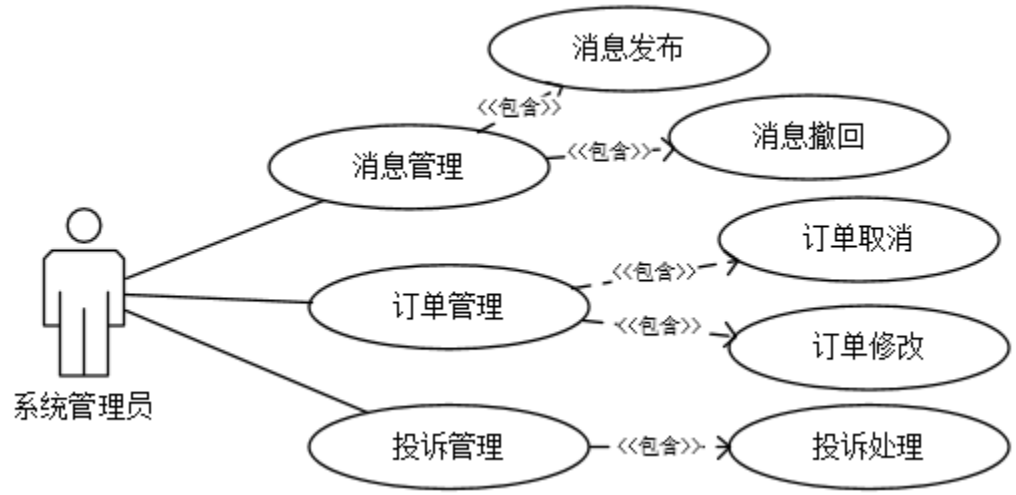

图3.3　系统管理员用例图

Fig.3.3　System manager use case diagram

关键用例描述如表3.3与表3.4所示：

表3.3　消息管理用例描述

Table3.3　Message management use case description

描述项	说明
用例名称	消息管理
用例描述	系统管理员对全站消息以及对用户推送消息的管理
参与者	系统管理员
前置条件	已经登录系统且具有管理员权限的用户，并且身份验证合格
后置条件	如果这个用例成功，系统中建立新的消息记录或进行相关消息操作
基本操作流程	1，管理员输入要推送的消息及所要推送的用户 2，系统验证数据合法性 3，管理员处理异常数据 4，消息推送完成
可选操作流程	1，取消相关操作 2，数据输入不合法，操作失败
被泛化的用例	无
被包含的用例	消息发布，消息撤回
被拓展的用例	无

表3.4 订单管理用例描述

Table3.4 Order management use case description

描述项	说明
用例名称	订单管理
用例描述	管理员对全站所有订单的管理
参与者	系统管理员
前置条件	已经登录系统且具有管理员权限的用户，并且身份验证合格
后置条件	如果这个用例成功，订单数据据相应操作而更新
基本操作流程	1，管理员登录系统 2，订单相关操作 3，系统进行合法性验证 4，操作成功
可选操作流程	1，取消相关操作 2，操作不合法时，操作失败
被泛化的用例	无
被包含的用例	订单取消、订单修改
被拓展的用例	无

3.5 可行性分析

3.5.1 社会可行性

本项目专门针对现代人们生活所需和生活服务类平台分析而设计。主要包括用户权限管理、订单管理和社区管理三大模块。以生活服务的电子商务的模式，突出客户间随时随地便捷式的交流，充分体现以消费者为中心，满足客户个性化、多样化服务的指导思想。为客户带来方便的、灵活的、高质量的服务，同时产生极大的社会经济价值。

法律因素方面。开发中所有商业软件都采用购买的方式使用正版软件。所有源代码为自主开发，自有知识产权。涉及到第三方控件和支付接口，也通过正规渠道进行商务购买。开发中使用的开源技术不存在法律纠纷。在用户使用可行性方面，用户对于电子采购交易平台的理解和接受需要一定时间周期。具体操作和维护人员需要进行专项培训。

在项目功能实施方面，我们力求做到以下几点：

1. 安全

（1）数据安全：网站所有数据的传输都经过了安全加密传输，避免数据被非法盗取，同时，个人数据访问需要分配权限，无权限不能访问。

（2）交易安全：所有资金先支付到平台管理，消费者确认满意后才能关闭交易。

2. 完善

（1）全面的内容：该服务网站不只是提供简单的商品交易信息，我们更注重生活中的任何个人所需，例如：找代驾、帮排队、找人帮购物、取件、送件，失物招领、宠物照看、二手处理等等。

（2）完善的功能：网站同时提供即时通信交流、评价、优惠活动等。

3. 便捷

在该网站平台上，可以发布或找到自己的任何所需，更加方便快捷，无需成本，更实惠有效。

综上所述，本项目没有违背法律和道德，力求达成一个更安全、更完善、更便捷的生活服务平台，所以在社会上是可行的。

3.5.2 经济可行性

本项目专门针对现代人们生活所需和生活服务类平台分析而设计。主要包括用户权限管理、订单管理和社区管理三大模块。该项目采用生活服务类电子商务的模式，突出客户间随时随地便捷式的交流，充分体现以消费者为中心，满足客户个性化、多样化服务的指导思想。为客户带来方便的、灵活的、高质量的服务，同时产生极大的社会经济价值。

在支出方面，搭建该系统需要组建开发团队，需要项目经理一名，架构师一名，分析员一名，测试员两名，程序员若干位，预计开发过程需要 6 个月。除了需要支出组建团队的费用外，硬件设备支出方面，需要内网数据库服务器 2 台， 存储系统，备份系统，Web 服务器 2 台，应用服务器 2 台，网络设备，还有安全设备。软件支出方面，需要服务器操作系统（应用内网和外网服务器操作系统 Unix、Linux 、Windows10），SQL server 数据库，备份软件，虚拟化软件，Web 中间件，杀毒软件等等。

其他费用除了项目中所需的固定资产投资，如人工费、材料费，其他支出流动资金还包括、差旅费用、调研费用等，最后还有系统的运行维护费用等等。

3.5.3 技术可行性

在功能目标能够如期达到的情况下，按对项目影响程序由大到小的依次排序为：资金、人员配置、软硬件设备。在技术可行性上具体体现在以下几个方面：

1. 开发技术：

本项目开发平台基于适用于 Android、iOS、Windows、Web 和云的应用等集于一体的 Visual Studio 2015，可以快速编码、轻松调试和整断、方便测试与发布。同时，为了方便团队的协作开发，采用 Git 版本控制器，可以更有效的解决代码冲突，提高团

队开发的效率等。

2. **系统功能分析与设计：**

   项目中采用面向对象的软件开发方法和思路，将抽象的事物以静态的、结构化的系统数据性质表示出来，提高了程序的灵活性、重用性和可维护性。

3. **UML 模型分析与设计：**

   UML 作为一种建模标准，是面向对象开发过程中必不可少的，它为软件开发的所有阶段提供模型化和可视化支持，便于开发和理解。

3.5.4 可行性分析结论

通过对系统的经济、技术、社会等可行性分析，可以确定该系统具有开发的完全必要性和可行性，予以立项开发。

3.6 本章小结

本章通过对系统的设计目标的分析对系统有了明确的定位，通过用例分析和可行性的详细分析，也明确了系统的需求，为以后系统的概要设计和详细设计打下了良好的基础。通过需求分析，确定该系统具有开发的必要性和可行性。

第四章 概要设计

概要设计的主要任务是把需求分析得到的系统模型转换为数据结构和软件结构。数据结构设计包括数据特征的描述、确定数据的结构特性、以及数据库的设计[10]。设计软件结构的具体任务是：确定一个复杂系统的功能模块划分、建立模块的层次结构及调用关系、确定模块间的接口及人机界面等的初步设计。

4.1 系统功能模块划分

4.1.1 功能模块图

消息管理、订单管理以及活动管理的功能模块图如图 4.1 所示。

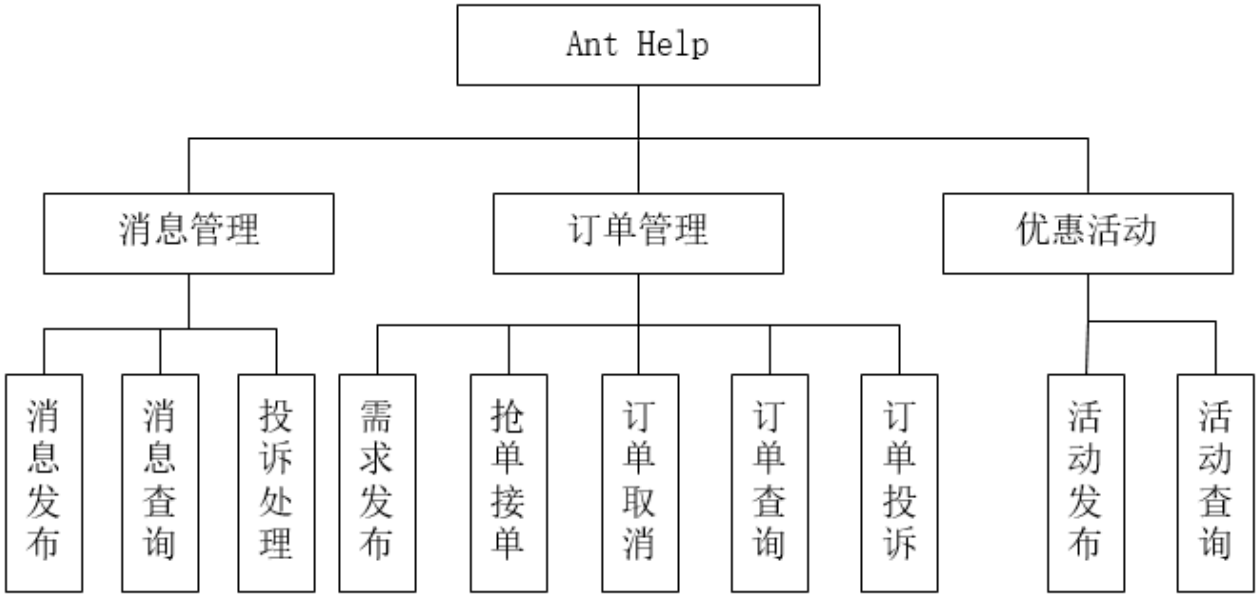

图4.1 消息、订单及活动管理功能模块图

Fig.4.1 Function module of message order and activity management diagram

1. **消息管理：**

(1)消息发布：用于管理员对用户发送系统消息，发送后在用户的消息栏内进行通知，用户可进行查看；

(2)消息查询：方便管理员进行消息的统一管理；

(3)投诉处理：投诉信息来源于网站投入使用后的各个流程，用户对系统的建议或对其他用户的投诉信息，可以向系统投诉信箱发送信息，系统管理员可以查看用户发送来的投诉信息，并进行对投诉信息的处理。

2. **订单管理：**

(1)需求发布：指用户在该平台上发布个人需求所生成的发布订单；

(2)抢单接单：用户之间具有抢单竞争，同一发布订单，只能被另一个用户所接单，同时接单时由系统自动分配，其他用户接单情况显示操作失败；

(3)订单取消：发布的订单在未接单的情况下，可以进行取消操作；

(4)订单查询：用户可对自己发布的订单及自己的接单进行条件帅选式的查询；

(5)订单投诉：在任何订单状态(待接单、待支付、待完成、待确认、已结束等)中，都可以进行投诉，投诉信息反馈给系统，有管理员权限即可可进行后续处理。

3. 优惠活动：

(1)活动发布：活动的发布只能由具有管理员权限的用户发起，发布完成后其他用户可见并参与优惠活动中；

(2)活动查询：方便管理员对优惠活动的统一管理。

4.2 数据库设计

数据库设计是应用系统及软件设计中必不可少的步骤之一，是应用开发系统化的核心技术。它用来满足用户的应用需求，能够有效地存储数据，进行方便快捷式的管理。好的数据库设计是软件设计好坏的评估标准之一，因此构造良好的数据模式具有重大的意义。

4.2.1 概念结构分析

数据库结构分析阶段是在需求分析的基础上，从属于概念结构设计阶段，将需求分析得到的用户需求 (如订单、用户、消息、投诉信息等)抽象为信息结构，即概念结构。

描述概念模型的直接有效表现途径是 E-R 图(Entity-Relationship,实体-联系模型)，设计出能够满足用户需求的各种实体，以及实体之间的关系，为以后的逻辑结构打下基础。根据上一章的需求分析，可以得出如图 4.2 所示的本人负责模块的实体及实体关系图。

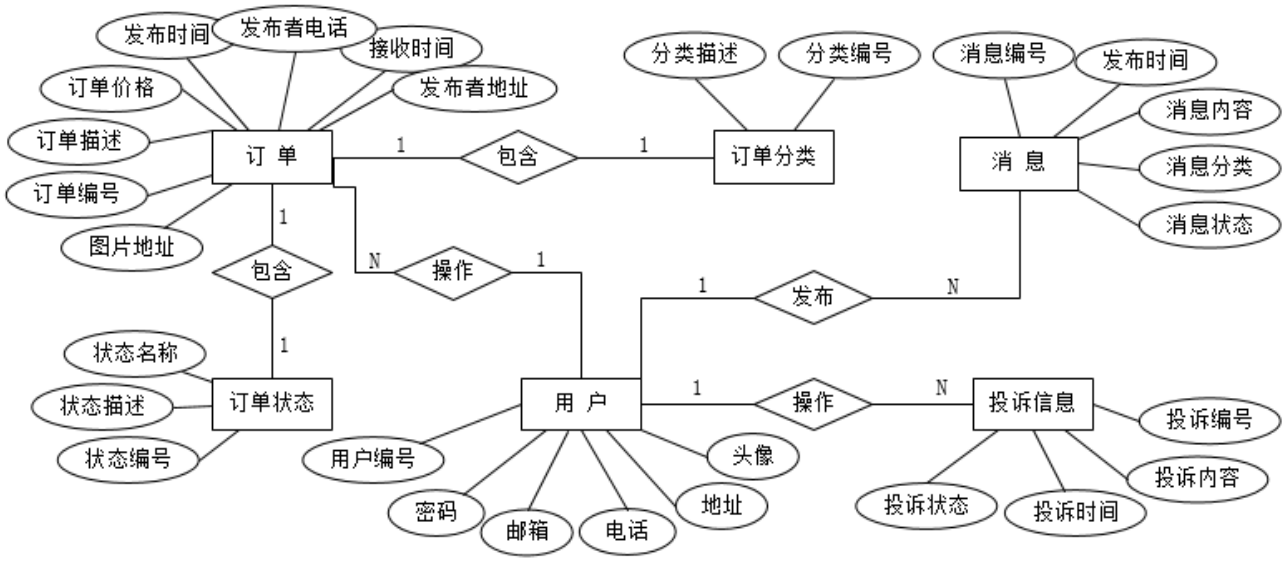

图 4.2　订单、消息管理 ER 图

Fig.4.2　Order and message management ER diagram

各个实体与实体联系的描述具体如下：

1. 实体：

用户实体(用户编号，密码，邮箱，电话，地址，头像)；

订单状态实体(状态编号，状态名称，状态描述)；

订单分类实体(分类编号，分类描述)；

订单实体(订单编号，订单描述，订单价格，发布时间，发布者电话，接收时间，发布者地址，图片地址)；

消息实体(消息编号，发布时间，消息内容，消息分类，消息状态)；

投诉信息实体(投诉编号，投诉内容，投诉时间，投诉状态)；

2. 实体关系：

(1)订单与订单状态之间是包含关系，一对一的关系，一个订单包含一个订单状态；

(2)订单与订单分类之间也是包含关系,一对一的关系,一个订单包含一个订单分类；

(3)用户与订单详情之间是操作关系，一对多的关系，一个用户可以拥有多个订单，并对其进行相关操作；

(4)用户与消息之间是发布关系，一对多的关系，一个用户可以发布多个消息；

(5)用户与投诉信息之间是操作关系，一对多的关系，一个用户可以发起多个投诉信息；

4.2.2 物理结构设计

在数据库设计的过程中，依据以上的概念结构分析，设计出 E-R 模型之后，充分考虑了

数据表的结构实现及数据表之间的联系，同时综合考虑数据流向和查询效率的情况下，设计了如下的物理结构设计，即数据表结构，如表 4.1-4.5 所示。

1. 订单表(Order)

表 4.1 订单基本数据表

Table 4.1 Order basic data table

字段名称	字段类型	是否为 NULL	含义
orderUid	uniqueidentifier	NOT NULL	订单编号
orderDescription	nvarchar(200)	NOT NULL	订单描述
orderCategoryId	int	NOT NULL	订单分类编号
publisherUid	uniqueidentifier	NOT NULL	发布者编号
publishTime	datetime	NOT NULL	发布时间
receiverUid	uniqueidentifier	NOT NULL	接收者编号
receiveTime	datetime	NOT NULL	接收时间
orderStatusId	int	NOT NULL	订单状态编号
orderValue	int	NOT NULL	订单价格
address	nvarchar(50)	NOT NULL	地址
phone	int	NOT NULL	电话

2. 订单分类表(OrderCategory)

表 4.2 订单状态数据表

Table 4.2 Order Status data table

字段名称	字段类型	是否为 NULL	含义
orderStatusId	int	NOT NULL	订单状态编号
orderStatusName	nvarchar(50)	NOT NULL	订单状态名称
orderStatusDescription	nvarchar(50)	NOT NULL	状态描述

3. 订单状态表(OrderStatus)

表 4.3 订单分类数据表

Table 4.3 Order Category data table

字段名称	字段类型	是否为 NULL	含义
CategoryId	Int	NOT NULL	订单分类编号
CategoryName	Nvarchar(100)	NOT NULL	分类名称
CategoryDescription	Nvarchar(100)	NOT NULL	分类描述

4. 消息表(Message)

表 4.4 消息数据表

Table 4.4 Message data table

字段名称	字段类型	是否为 NULL	含义
messageUid	uniqueidentifier	NOT NULL	消息编号
messageContent	Nvarchar(1000)	NOT NULL	消息内容
messagePusher	Nvarchar(100)	NOT NULL	消息发布者

messagePushTime	Datetime	NOT NULL	发布时间
messageCategoryId	Int	NOT NULL	消息分类编号
messagePushStatusId	Int	NOT NULL	消息状态编号
pushToUserUid	uniqueidentifier	NOT NULL	需推送的用户

5. 投诉信息表(Complain)

表 4.5 投诉信息数据表

Table 4.5 Complain infomation data table

字段名称	字段类型	是否为 NULL	含义
complainUid	uniqueidentifier	NOT NULL	主键 ID
complainContent	Nvarchar(100)	NOT NULL	投诉内容
complainUserUid	uniqueidentifier	NOT NULL	用户 ID
complainTime	datetime	NOT NULL	投诉时间
complainStatusId	int	NOT NULL	状态 ID

4.2.2 数据库表间关系

订单数据表与其他数据表间是主外键的关系，其他表中有相应的外键与其产生主外键约束关系。同时消息数据表中的 MessageUid 号作为主键也与其他表有主外键约束关系。如图 4.3 所示：

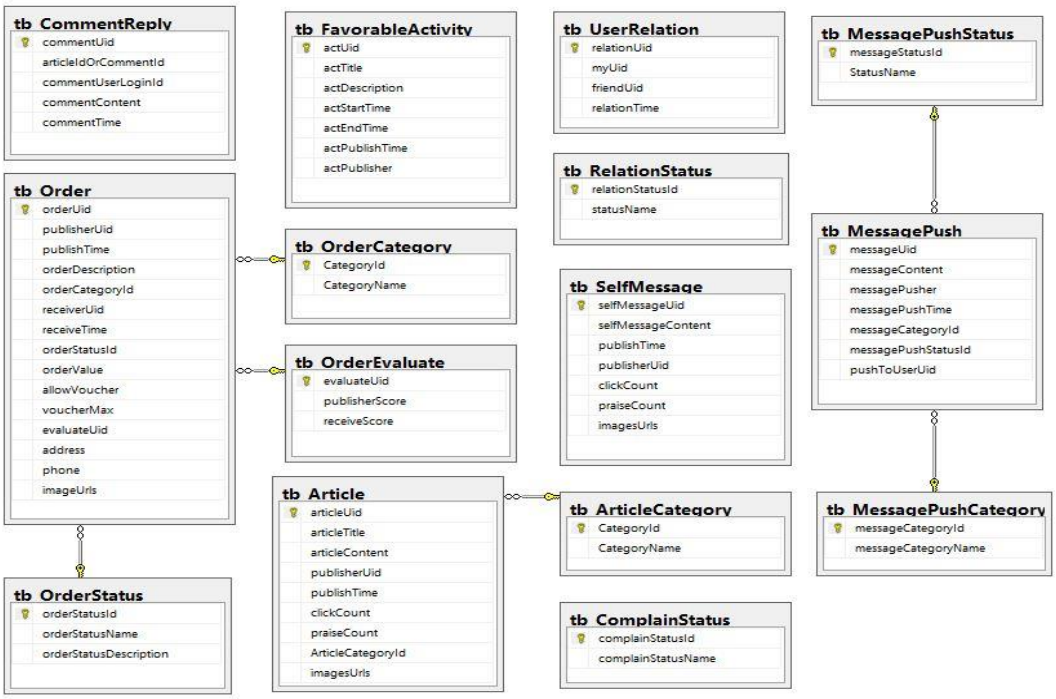

图 4.3 数据库表间关系图

Fig .4.3 Relationship between database tables diagram

4.3 本章小结

本章对统计过程管理进行了概要设计。本部分分析并绘制了该项目的功能模块图，明确了个人负责部分的功能模块。通过对数据特征描述的分析，确定了数据的结构特性，从而对该系统的数据库进行了设计。而后通过对系统结构和实体联系的分析，抽象出了系统的数据库表关系图。

第五章 详细设计

详细设计是概要设计的一个细化，是对每个功能模块的具体实现的表现。在软件开发过程中也是必不可少的步骤之一。

5.1 业务逻辑设计

根据上一章对本系统功能模块与实体的分析与数据库的设计等，可以逐步提取出系统的业务对象有：订单实体、订单分类实体、订单状态实体、用户信息实体、消息信息实体、投诉信息实体等，同时，初步得到了这些业务对象的属性以及对象之间的关系。本小节中将对业务的逻辑进行设计。

5.1.1 提取业务逻辑类

本系统中个人负责模块所用的业务逻辑类如下：

1. 实体类

实体类是用于对必须存储的信息和相关行为建模的类，它的存在通常都是永久性的，所具有的属性和关系也都是长期需要的，甚至在系统的整个生存期都需要。本系统涉及到的业务逻辑实体类包括：用户账户类、订单类、订单状态类、订单分类类、用户账目类、消息类、消息状态类和活动类。

2. 实体业务逻辑类

实体业务逻辑类封装了每个业务对象实体的基本增删改查操作以及查询操作。它还包括了一些复杂的对实体的操作流程，并提供相应的方法供控制器调用[11]。本系统中涉及到的实体业务逻辑类包括：用户账户逻辑类、订单逻辑类、订单状态逻辑类、订单分类逻辑类、用户账目逻辑类、消息逻辑类、消息状态逻辑类和活动逻辑类。

3. 控制器类

控制器类描述了一个业务用例所具有的事件流控制行为，控制一个用例中的事件顺序。控制类的实例通常控制其他对象，因此，它具有一定的协调性质，对用例的特有行为进行了封装。在详细分析了系统的业务逻辑智商，提取了本系统中的关键控制器类包括：用户控制类、用户金额控制类、订单控制类、消息控制类和活动控制类。

5.1.2 业务逻辑类分析

1．订单发布

在订单发布用例的逻辑关系中，需要涉及到三个控制器类(UserAccount 控制类、

UserMoney 控制类和 Order 控制类)，三个控制类的先后调用完成了整个的订单发布用例逻辑。在控制类的实现中，分别用事务的方式对 UserAccountService、UserAccountInfoService、OrderService、UserMoneyService、File 服务类进行了调用，服务类又通过统一的数据上下文完成对实体的操作。在各层类的调用中，不可避免的要使用到边界类，这里我们主要使用的边界类由 Helper_DG，由 qx_Frame 框架提供的帮助类库还有 Exception_DG 由 qx_Frame 提供的通用异常处理类，简化了业务逻辑的实现。订单发布用例逻辑类图如图 5.1 所示。

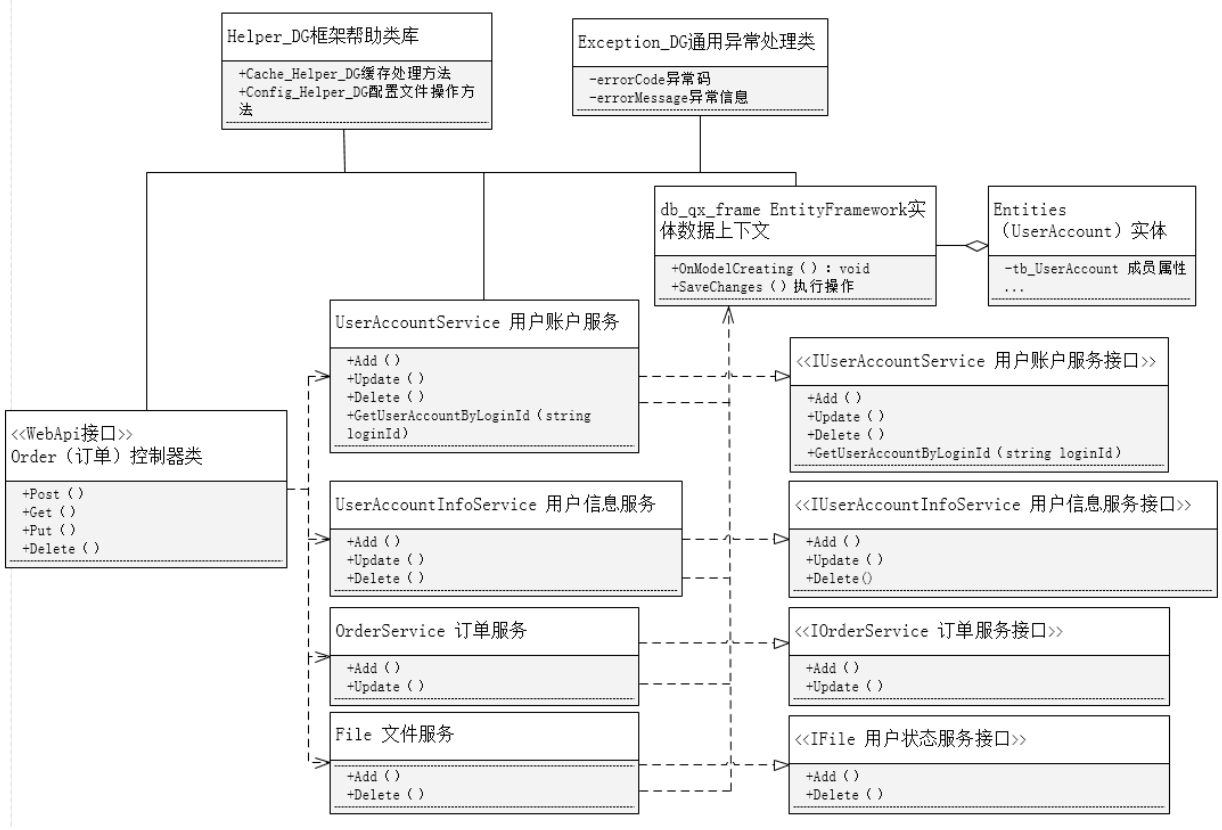

图 5.1　订单发布逻辑类图

Fig .5.1　Publish order case logic class diagram

2．消息推送

在消息推送的逻辑关系中，涉及到了 UserAccount 控制类和 Message 控制类这两个控制器类，它们先后调用完成了整个的消息推送的用例逻辑。在控制类的实现中，分别用事务的方式对 UserAccountService 和 MessagePush 服务类进行了调用，服务类又通过统一的数据上下文完成对实体的操作。在各层类的调用中，不可避免的要使用到边界类，这里我们主要使用的边界类由 Helper_DG，由 qx_Frame 框架提供的帮助类库还有 Exception_DG 由 qx_Frame 提供的通用异常处理类，简化了业务逻辑的实现。消息推送用例逻辑类图如图 5.2 所示。

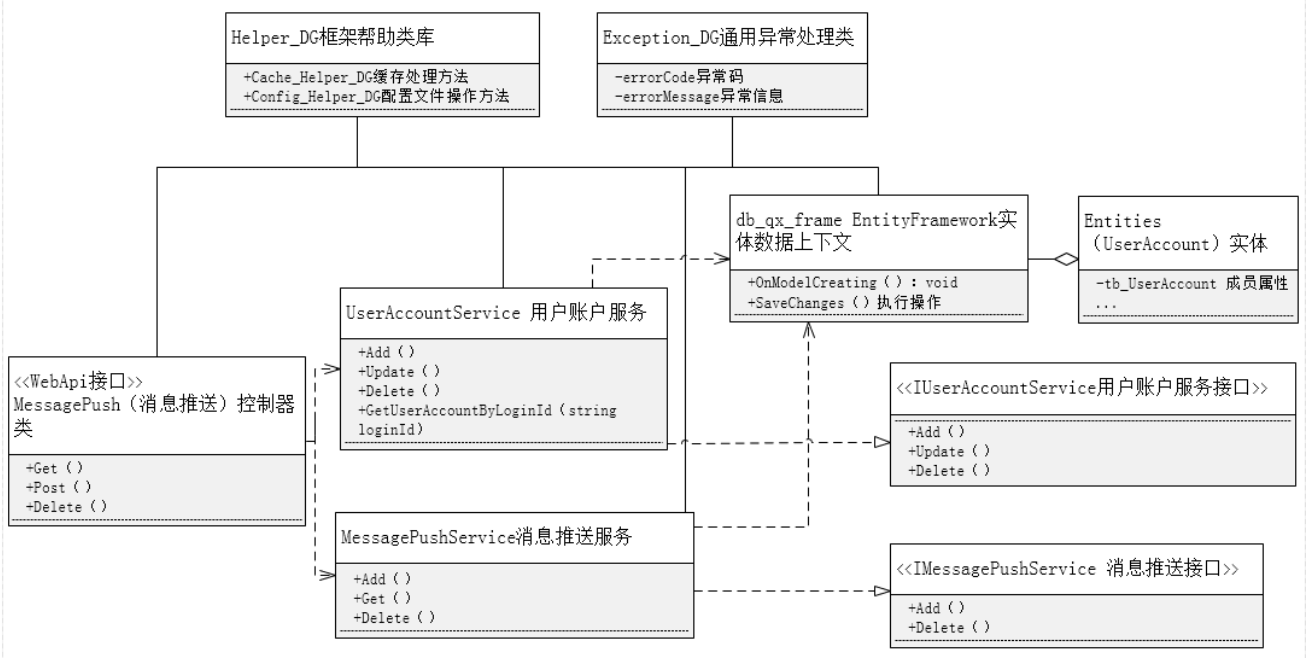

图 5.2　订单发布逻辑类图

Fig .5.2　Publish order case logic class diagram

3．活动管理

在活动管理用例的逻辑关系中，需要涉及到两个控制器类(UserAccount 控制类和 Activity 控制类)，两个控制类的先后调用完成了整个的活动管理用例逻辑。在控制类的实现中，分别用事务的方式对 UserAccountService 和 FavorableActivityService 服务类进行了调用，服务类又通过统一的数据上下文完成对实体的操作。在各层类的调用中，不可避免的要使用到边界类，这里我们主要使用的边界类由 Helper_DG，由 qx_Frame 框架提供的帮助类库还有 Exception_DG 由 qx_Frame 提供的通用异常处理类，简化了业务逻辑的实现。活动管理用例逻辑类图如图 5.3 所示。

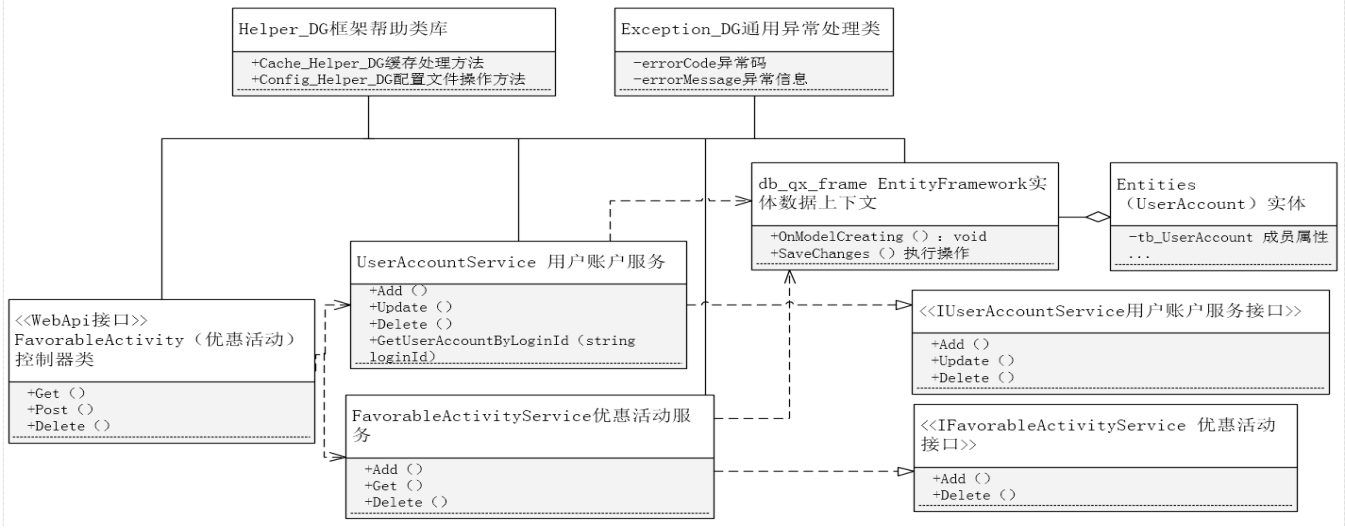

图 5.3　订单发布逻辑类图

Fig .5.3　Publish order case logic class diagram

5.2 业务逻辑交互分析

业务逻辑与软件产品紧绑在一起，所谓软件产品，一定是在某一个领域内实现了某些业务，而业务的逻辑交互分析自然在业务的实现过程中起到了重要的作用。数据访问操作是业务逻辑的一部分，从时间结构上看，先有了业务逻辑的概念，才有数据访问的概念。业务逻辑衍生自软件本身，数据访问，衍生自业务逻辑。为了更好的分析和描述业务逻辑交互，在本小节采用了软件工程建模中常用到的 UML 建模工具--时序图。

时序图主要用于描述按照交互发生的一系列顺序，显示业务对象之间的交互关系。通常用来描述用例的行为，验证用户需求是否已经落实到能够完成这些功能的业务逻辑类中实现。

5.2.1 订单业务

1. 立即抢单

已登录的用户在平台的需求详情页上可进行“立即抢单”操作，系统获取用户的登录 Uid，向服务器发送抢单请求，若多个用户同时进行抢单操作，通过并发锁控制并发数据的实例，只允许一人能抢单成功。立即抢单时序图如图 5.4 所示。

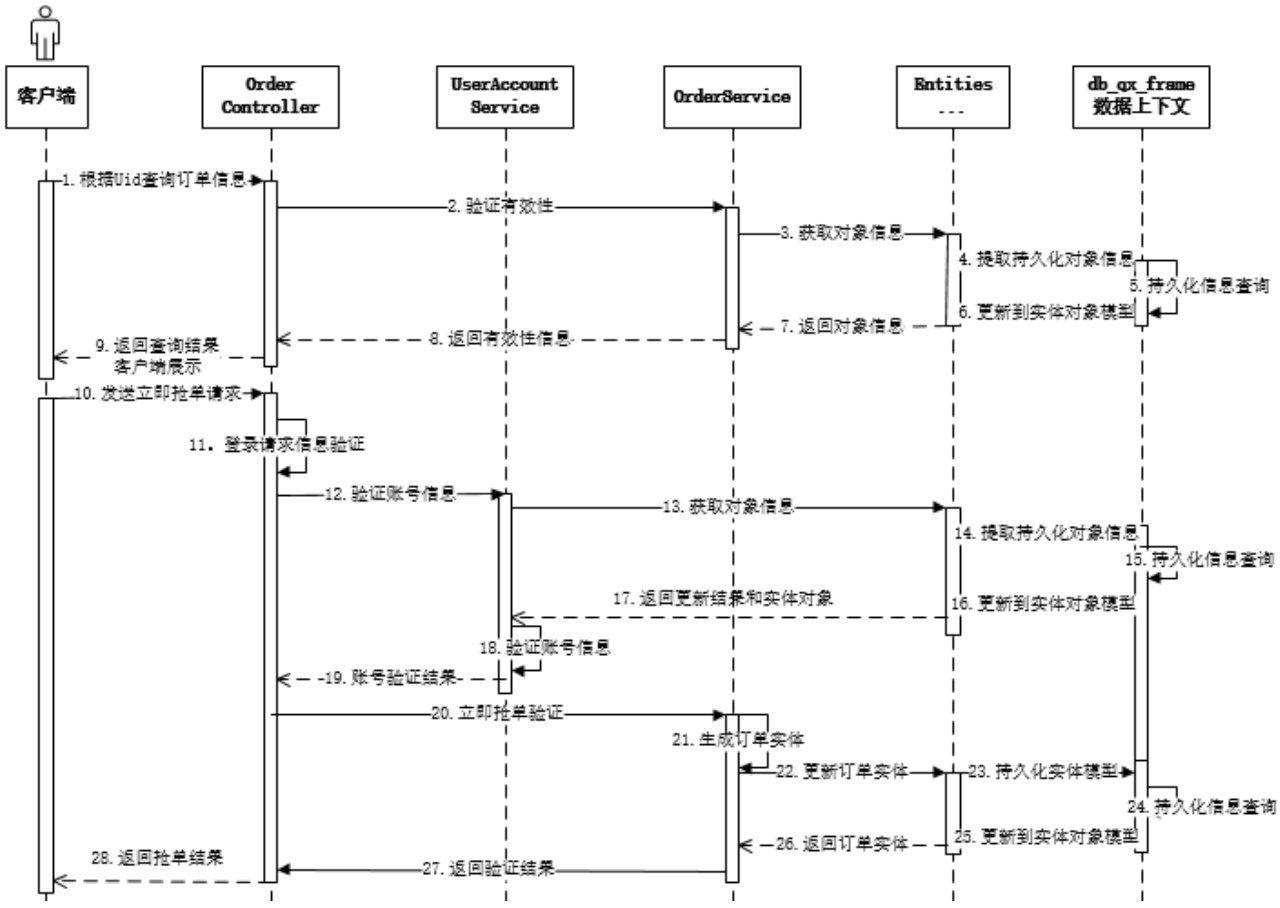

图 5.4　立即抢单时序图

Fig .5.4　Get order immediately sequence diagram

立即抢单的功能逻辑说明如下表 5.1 所示。

表 5.1　立即抢单功能逻辑模型说明

Table 5.1　Get order immediately model detail

名称	类型	说明	资源
客户端	参与者	表示任何请求发起者	-
OrderController	控制类	订单控制的对外控制接口	OrderController.cs
OrderService	实体类	订单控制服务类	OrderService.cs
UserAccountService	实体类	用户账户服务类	UserAccountService.cs
Entities	实体类	数据对象实体	QX_Frame.Data.Entities.cs
db_qx_frame 数据上下文	实体类	持久化操作数据上下文	db_qx_frame.cs

2. 订单状态修改

已登录的用户在个人中心的“我的接单”和“我的发布”中可以查看自己的相关需求订单，并依据当前的个人需求进行订单状态的管理。订单状态修改时序图如图 5.5 所示。

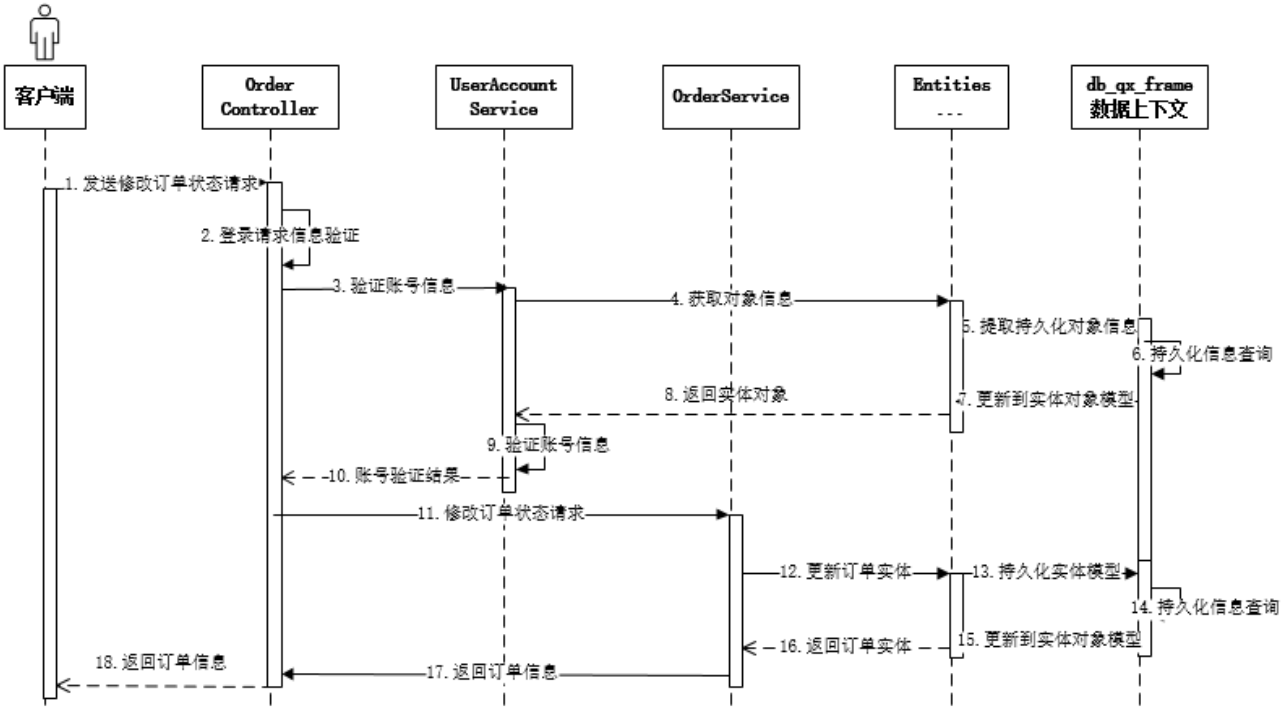

图 5.5　订单状态修改时序图

Fig .5.5　Update order status sequence diagram

订单状态修改的功能逻辑说明如下表 5.2 所示。

表 5.2　订单状态修改功能逻辑模型说明

Table 5.2　Update order status model detail

名称	类型	说明	资源
客户端	参与者	表示任何请求发起者	-
OrderController	控制类	订单控制的对外控制接口	OrderController.cs
OrderService	实体类	订单控制服务类	OrderService.cs
Entities	实体类	数据对象实体	QX_Frame.Data.Entities.cs

db_qx_frame 数据上下文	实体类	持久化操作数据上下文	db_qx_frame.cs

5.2.2 消息业务

已经登录的用户在系统首页菜单栏上可点击“消息”对本人收到的消息进行查看，系统获取用户的登录 Uid，向服务器发送消息查看请求，验证成功返回所有的消息信息。同时，用户对自己接收到的消息可进行修改，主要是指对消息状态的修改，包括“未读”和“已读”。消息查看和状态修改时序图如图 5.6 所示。

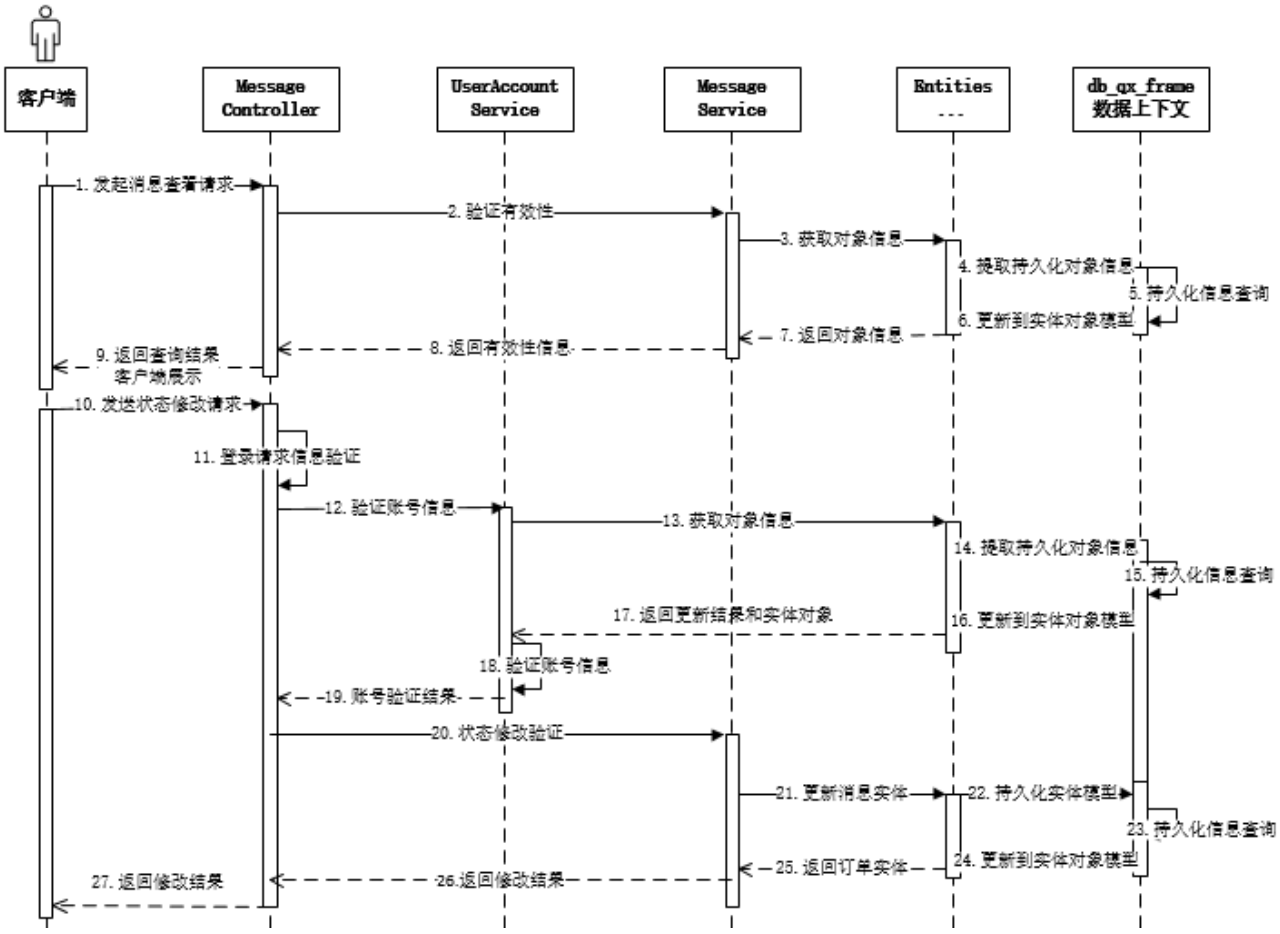

图 5.6　消息查看和状态修改时序图

Fig .5.6　Get message and update status sequence diagram

消息查看和状态修改的功能逻辑说明如下表 5.3 所示。

表 5.3　消息查看和状态修改功能逻辑模型说明

Table 5.3　Get message and update status model detail

名称	类型	说明	资源
客户端	参与者	表示任何请求发起者	-
MessageController	控制类	消息控制的对外控制接口	MessageController.cs
MessageService	实体类	消息控制服务类	MessageService.cs

UserAccountService	实体类	用户账户服务类	UserAccountService.cs
Entities	实体类	数据对象实体	QX_Frame.Data.Entities.cs
db_qx_frame 数据上下文	实体类	持久化操作数据上下文	db_qx_frame.cs

5.3 本章小结

在本章中，提取了系统实现需要用到的业务逻辑类，并对具体的业务逻辑类的进行了详细的分析，明确了系统实现中需要设计的业务对象。在类逻辑关系的分析中，采用了对类之间的调用关系的时序图来进行表述，更好的描述了时间顺序下各业务对象之间的交互。

第六章 系统实现与代码编写

系统实现与代码编写是将软件设计变为可视化的一个过程，这一阶段主要任务是代码的编写与用户交互界面的实现。经过两个月的项目编码周期，本项目的编码工作在小组的积极合作下如期完成。

1.1 系统编码结构

一个系统体系结构的好坏直接决定了系统的性能和可扩展性，它同时影响了开发者的工作效率，因此有一个良好的系统编码结构是一个项目走向成功的开端。如图 6.1 所示为本系统项目编码结构图。

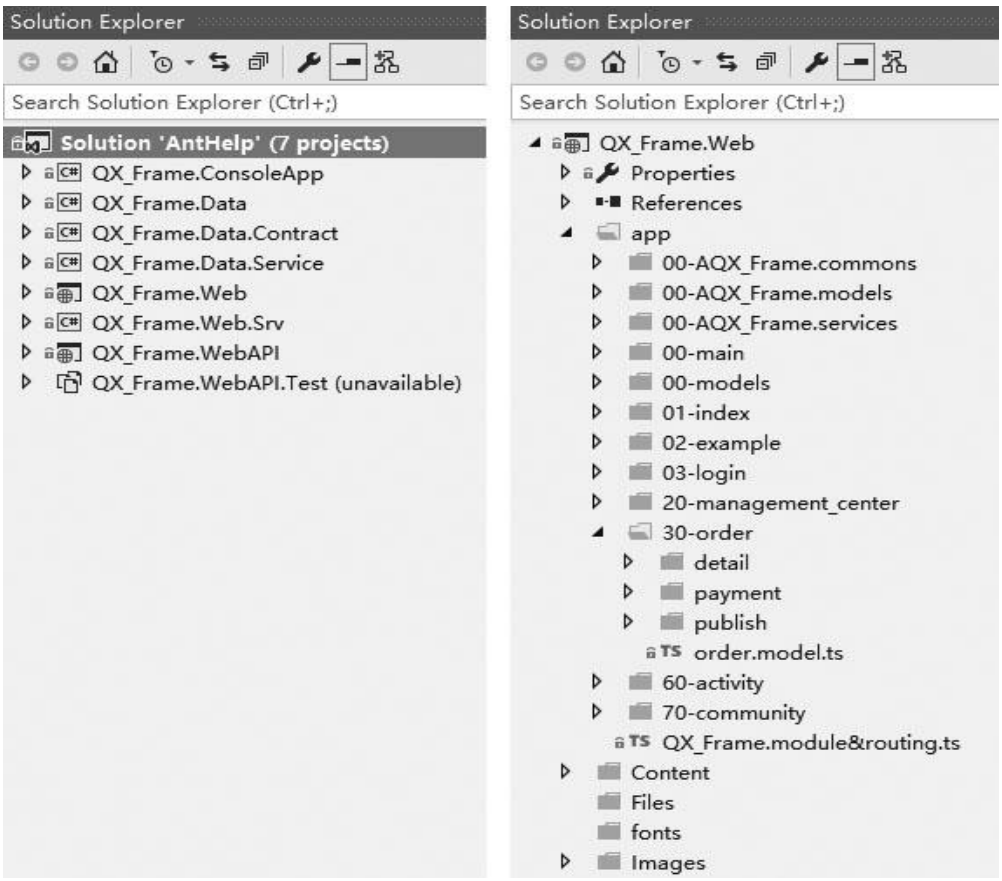

图 6.1　系统代码结构图

Fig.6.1　System code structor diagram

在本系统的开发中，将系统开发划分成了 7 个项目：QX_Frame.ConsoleApp、QX_Frame.Data 、 QX_Frame.Data.Constract 、 QX_Frame.Data.Service 、 QX_Frame.Web 、QX_Frame.Web.Srv、QX_Frame.WebAPI 和 QX_Frame.WebAPI.Test。这七个项目协调工作，共同完成了本系统的业务需求。

6.2 系统业务实现

6.2.1 订单业务

订单业务是整个系统业务实现的核心部分，它包含了订单分类、订单发布、已发布订单的管理、接单抢单操作、已接单订单的管理等相关业务的实现。以下为订单相关业务的详细说明。

1. 订单分类

如图 6.2 为首页显示的订单分类部分。

图 6.2　首页订单分类界面

Fig.6.2　Order category of system index interface

首页的订单分类主要展示了订单的各个分类及分类下的各个订单项，按照用户的需求分析，将订单主要划分为以下 9 类：代理销售、维修装修、汽车服务、免费专场、宠物之家、代帮代购、休闲服务、家政服务和其他服务。发布者在发布时可根据自己的需求选择相应的

分类，便捷有效的进行用户订单的查看。

2. 订单发布

订单发布界面如图 6.3 所示。

图 6.3　订单发布界面

Fig.6.3　Order publish interface

订单发布是通过 Form 表单实现用户的输入，兼容图片上传功能，上传成功后返回服务器图片地址，在表单中进行预览显示。同时，要求对用户输入的数据进行合法性验证：价格输入不能为非数字，电话号码输入格式是否正确等。

3. 已发布订单的管理

在个人中心，用户可以对自己发布所生成的订单进行管理，管理界面和查看订单详情界面如图 6.4 和 6.5 所示。

图 6.4　已发布订单管理界面

Fig.6.4　Published order management interface

图 6.5　查看订单详情界面

Fig.6.5　Order detail interface

在用户对已发布的订单(即‘我的发布记录’)进行管理时，按照订单状态将订单划分为以下 5 类：待支付订单、待接单订单、待确认订单、已结束订单和已取消订单。

在订单发布成功后，将订单状态修改为待支付，用户只有在该订单被支付后才能在平台上对其他用户可见，进而被其他用户接单抢单；在支付过程中，用户账户余额不足时，提醒用户先去充值，充值成功后，且余额充足的情况的可进行支付，支付成功后的订单状态为待接单，会在平台首页或分类页上加载该订单，等待其他用户的接单抢单响应。当其他用户接单抢单成功，并反馈已经完成了该订单的需求，订单状态为待确认，此时需要发布者确认订

单的完成。发布者确认接单者完成无误后，订单状态修改为已结束，该交易结束。已取消的订单生成在未支付与待支付的过程中，即在发布者未支付或发布者已支付、接单者未接单时，发布者可以进行取消该订单的操作。

4. 接单抢单操作

如图 6.6 为订单详情界面，其中包含了用户的接单抢单操作。

图 6.6　订单详情界面

Fig.6.6　Order detail interface

订单详情界面展示了相关订单的主要信息，包括：发布者、价格、分类、简介、地址、图片描述等。该订单在没有被接单或抢单的状态下，有红色的‘立即抢单’按钮，已登录的用户可进行抢单操作，游客只能进行浏览，不能进行抢单操作，进行该操作时，提示用户先登录。需要注意的是，已登录的用户只能抢别人发布的需求单，本人发布的订单不允许本人抢单，避免借助平台刷单的情况。当订单被某一用户接单成功后，红色的‘立即抢单’按钮会置为灰色的‘手慢一步，试试其他订单吧~’按钮，并禁止用户的抢单操作，保持唯一性，即一个订单只能被一个用户所接单。

5. 已接单订单的管理

在个人中心，用户可以对自己已经接收或抢到的订单进行管理，管理界面和已接单订单详情界面如图 6.7 和 6.8 所示。

图 6.7　已接单订单管理界面

Fig.6.7　Received order management interface

图 6.8　已接单订单详情界面

Fig.6.8　Received order detail interface

在用户对已经接收或抢到的订单(即‘我的接单记录’)进行管理时，按照订单状态将订单划分为以下 3 类：待完成订单、已完成订单和已结束订单。

在接单或抢单成功后，订单的为待完成，接单者按照发布者的发布需求，于线下完成该需求，即可在平台上将订单详情中的订单状态修改为已完成。当接单者标记为已完成订单后，需要发布者确认订单的完成，确认完成无误后，发布者将订单标记为已结束，该交易结束。

6.2.2 活动业务

活动业务是指管理员发布平台的相关优惠活动或重点文案性消息通知等，要求用户参与其中或只进行了解的重要信息的展示，如图 6.9 所示为活动中心界面。

图 6.9　活动中心界面

Fig.6.9　Activity center interface

6. 2. 3 消息业务

消息业务是系统消息管理模块的重要体现部分，从用户角色出发可以将消息业务划分为管理员消息管理和普通用户消息管理两种。

1. 管理员消息管理

如图 6. 10 和 6. 11 所示，为管理员消息管理界面，包括消息推送界面和消息管理界面两部分。

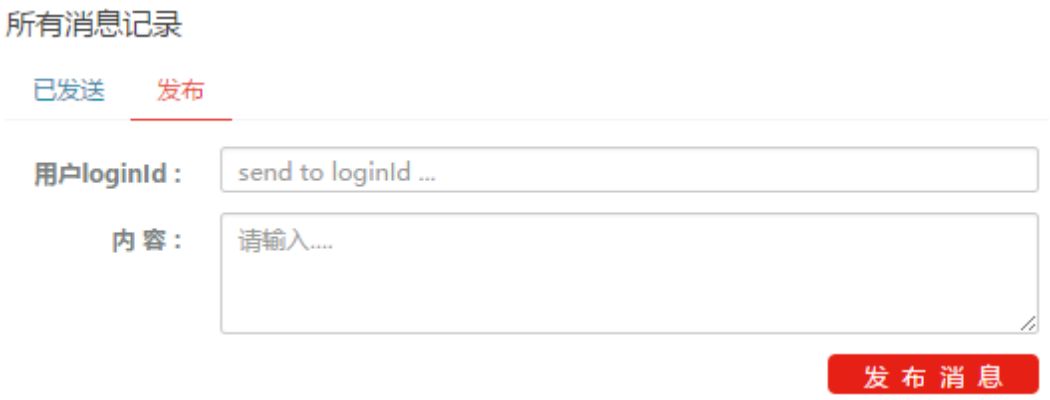

图 6.10　消息推送界面

Fig.6.10　Publish message interface

图 6.11　消息管理界面

Fig.6.11　Message management interface

管理员(即具有管理权限的用户)可对某用户进行消息推送，推送成功后用户可在自己的消息列表中进行查看。同时管理员对已推送的消息可进行撤回操作，撤回成功后从用户的消息列表中删除该条消息，用户不可见。

2. 普通用户消息管理

如图 6.12-6.14 所示，为普通用户消息管理界面，包括未读消息列表界面、查看消息界面和已读消息列表界面。

图 6.12　未读消息列表界面

Fig.6.12　Unread message list interface

图 6.13　消息查看界面

Fig.6.13　View message interface

图 6.14　已读消息列表界面

Fig.6.14　Readed message list interface

当管理员对某一用户进行消息推送后，消息会在用户的未读消息列表中显示，用户可对该条新消息进行查看操作，查看后，消息自动标记为已读消息。用户可在已读消息列表中对消息进行删除操作。为了方便用户查阅，每条消息通知罗列时都对其进行了相应的编号。

6.3 本章小结

本章对系统的编码结构及业务实现进行了分析，通过分析更加明确了系统的目标和整体结构。主要包括订单业务、活动业务和消息业务的实现结果分析，其中订单业务作为整个系统实现的核心业务，详细分析了订单分类、订单发布、已发布订单的管理、接单抢单操作、已接单订单的管理等相关订单业务的实现，使得用户行为分析更加明了，充分体现系统‘你需求，我实现’的总目标。具体代码实现见附录。

第七章 系统测试

软件测试是软件开发及维护阶段中质量控制的不可或缺的步骤之一，以下对本系统的实现进行了软件测试分析和测试实现。

7.1 测试目的

软件测试的目的是尽可能的发现软件中存在的错误，并进行改正，提高软件安全性和可靠性。软件测试的关键在于“分析”和“发现”，分析包含了项目的开发过程的分析和业务实现是否合理化的分析，发现指的是发现软件开发过程中不易发现或者容易被忽略部分的错误。在当今的互联网时代，高速迭代，持续集成及部署变得越来越重要。原来开发人员写完程序再交给测试的模式，会使得软件的开发周期变长。所以近年来开发和测试已经渐渐合体了，开发者开发的过程也是一个自测的过程，测试技能变成了软件工程师的技能树的一部分。

7.2 测试过程

本项目的测试过程包含了单元测试、集成测试、确认测试和系统测试。最主要使用的测试方法为黑盒测试法。通过手动操作该系统，对照系统最初的设计，主要查看是否存在异常以及操作的结果是否正确。

7.2.1 订单管理测试

订单管理测试的过程如表 7.1-7.3 所示。

表 7.1　订单发布测试用例

Table 7.1　Publish order test case

项目/软件	蚂蚁热帮(Ant Help)	**程序版本**	1.0.0
功能模块	订单发布	**编制人**	白甜甜
用例编号	订单管理_1	**编制时间**	2017-5-11
功能特性	对发布的订单数据进行导入		
测试目的	验证是否输入合法的信息，确保数据完整性，以及确保数据写入数据库		
预置条件	已成功登录系统的用户	**特殊规程说明**	价格单位为元
参考信息	需求分析中关于“订单管理”的说明		
测试数据	分类=“家政服务”	描述=“卫生打扫”	单张图片上传预览

	分类=“代帮代购”	价格=“-100”	多张图片上传预览
	分类=“免费专场”	价格=“0”	地址=“123456”
	分类=“汽车服务”	价格=“freei”	地址=“america”
	分类=“休闲服务”	价格=“10.56”	地址=“西青大学城”
	分类=“宠物之家”	价格=“60.00”	地址=“南开周边”
	分类=“其他服务”	价格=“1200”	地址=“北京朝阳区”
	分类=“维修装修”	电话=“abcdefg”	地址=“”
	分类=“代理销售”	电话=“1233456”	价格=“”
	分类=“其他服务”	电话=“12334567891011”	描述=“”

操作步骤	操作描述	数据	期望结果	实际结果	测试状态（P/F）
1	直接点击提交。	都为空	显示警告信息“描述不能为空”、“价格不能为空!”、“地址不能空!”、“电话不能为空!”	（符合）	P
2	在价格和电话文本框内输入汉字	价格=“你好”，电话=“你好”	显示警告信息“请填写合法数字！”	（符合）	P
3	填写表单，点击提交	测试数据	添加成功	（符合）	P

表 7.2　个人中心订单管理测试用例

Table 7.2　Personal center order management test case

项目/软件	蚂蚁热帮(Ant Help)		程序版本	1.0.0
功能模块	我的接单、我的发布		编制人	白甜甜
用例编号	订单管理_2		编制时间	2017-5-11
功能特性	用户对已发布或已接单的订单进行修改操作			
测试目的	验证订单状态是否合理，确保状态成功修改到数据库，并读取无误			
预置条件	已成功登录系统的用户		特殊规程说明	无
参考信息	需求分析中关于“订单管理”的说明			
测试数据	订单状态=“待支付”	订单状态=“已支付”		订单状态=“待接单”
	订单状态=“已接单”	订单状态=“已完成”		订单状态=“完成确认”
	订单状态=“订单结束”	订单状态=“发布者取消”		

操作步骤	操作描述	数据	期望结果	实际结果	测试状态（P/F）
1	‘我的发布’中的待支付订单	无	提示：需求已发布，需要支付后才能对其他用户可见。	（符合）	P
2	‘我的发布’中的待接单订单	无	提示：已完成支付，请耐心等待其他用户的相应。	（符合）	P
3	‘我的发布’中的待确认订单	无	提示：对方已完成需求订单，正在等待您的反馈意见。	（符合）	P
4	‘我的发布’中的已结束订单	无	提示：该订单已完成，交易已结束。	（符合）	P
5	‘我的发布’中的已取消订单	无	提示：该订单已被取消。	（符合）	P
6	‘我的接单’中的待完成订单	无	提示：恭喜您抢单成功，对方正在等待您的完成反馈。	（符合）	P
7	‘我的接单’中的已完成订单	无	提示：您的完成反馈已发给对方，正在等待对方确认。	（符合）	P
8	‘我的接单’中的已结束订单	无	提示：该订单已完成，交易已结束。	（符合）	P

表 7.3　用户抢单测试用例

Table 7.3　User get order test case

项目/软件	蚂蚁热帮(Ant Help)		程序版本	1.0.0
功能模块	用户抢单		编制人	白甜甜
用例编号	订单管理_3		编制时间	2017-5-11
功能特性	用户对已发布的订单可进行抢单操作			
测试目的	验证订单状态是否合理，确保状态成功修改到数据库，并读取无误			
预置条件	已成功登录系统的用户		特殊规程说明	无
参考信息	需求分析中关于“订单管理”的说明			
测试数据	订单状态=“已支付”	订单状态=“待接单”		订单状态=“已接单”

操作步骤	操作描述	数据	期望结果	实际结果	测试状态（P/F）
1	订单详情中	发布者=‘bantina’	提示：自己发布的需求订单	（符合）	P

	‘立即抢单’操作	登录用户=‘bantina’	不能自己抢。		
2	订单详情中‘立即抢单’操作	发布者=‘jack’ 登录用户=‘bantina’	提示：抢单成功。	（符合）	P
3	订单详情中‘立即抢单’操作	发布者=‘jack’ 接单者=‘bantina’	提示：手慢一步，试试其他订单吧。	（符合）	P

小结：数据管理基本功能完善，实际结果与期望结果符合。

7.2.2 消息管理测试

消息管理测试的过程如表 7.4 与表 7.5 所示。

表 7.4　消息发布测试用例

Table 7.4　Publish message test case

项目/软件	蚂蚁热帮(Ant Help)		**程序版本**	1.0.0
功能模块	消息发布		**编制人**	白甜甜
用例编号	消息管理_1		**编制时间**	2017-5-11
功能特性	对发布的消息数据进行导入			
测试目的	验证是否输入合法的信息，确保数据完整性，以及确保数据写入数据库			
预置条件	已成功登录系统具管理权限的用户		**特殊规程说明**	无
参考信息	需求分析中关于“消息管理”的说明			
测试数据	用户 loginId=“”	用户 loginId=“aaa”		用户 loginId=“jack”
	内容=“”	内容=“Welcome”		内容=“欢迎您的加入”
	登录用户=“”	登录用户=“jack”		登录用户=“bantina”

操作步骤	**操作描述**	**数据**	**期望结果**	**实际结果**	**测试状态（P/F）**
1	直接点击发布消息	都为空	显示警告信息“用户 loginId 不能为空”、“发布内容不能为空!”	（符合）	P
2	点击管理中心	登录用户=“”	提示：尚未登录。	（符合）	P
3	点击管理中心	登录用户=“jack”	提示：您不具备管理权限	（符合）	P
4	点击发布消息	用户 loginId =“bantina”,	推送成功	（符合）	P

表 7.5 消息查看测试用例

Table 7.5 Check message test case

项目/软件	蚂蚁热帮(Ant Help)		程序版本	1.0.0
功能模块	消息查看		编制人	白甜甜
用例编号	消息管理_2		编制时间	2017-5-11
功能特性	用户对个人消息进行查看并管理			
测试目的	验证消息状态是否修改合理，并确保数据成功修改到数据库			
预置条件	已成功登录系统的用户		特殊规程说明	无
参考信息	需求分析中关于“消息管理”的说明			
测试数据	登录用户=“”	消息状态=“未读”		消息状态=“已读”
	登录用户=“bantina”			

操作步骤	操作描述	数据	期望结果	实际结果	测试状态（P/F）
1	直接点击消息栏查看	登录用户=“”	提示：尚未登录。	（符合）	P
2	点击未读消息栏	登录用户=“bantina”	显示未读消息	（符合）	P
3	点击未读消息栏的查看	登录用户=“bantina”	消息成功读取，并修改状态为“已读”	（符合）	P
4	点击已读消息栏	登录用户=“bantina”	显示已读消息	（符合）	P
5	点击已读消息栏的删除	登录用户=“bantina”	消息成功删除，数据库数据删除成功	（符合）	P

小结：数据管理基本功能完善，实际结果与期望结果符合。

7.3 测试结论

测试结论是测试报告的核心部分，对整体的测试过程及结果进行评估。本项目的需求覆盖分析及测试覆盖分析如表 7.6 和表 7.7 所示。

表 7.6 需求覆盖分析

Table 7.6 Demand coverage analysis

功能	测试点描述	是否测试	重要等级	是否通过	备注
订单管理	对发布需求数据进行添加，查询，删除以及修改。	是	S 级	通过	无

立即抢单	对已经发布的订单进行抢单	是	S 级	通过	无
活动管理	对活动的发布数据进行添加，查询，删除以及修改。	是	A 级	通过	无
消息管理	对消息推送数据进行添加，查询，删除以及修改。	是	B 级	通过	无

说明：这里的重要等级分为 S 级、A 级、B 级，其中 S 级是最重要的。由表格可知，需求总数为 4 部分，测试通过的需求总数为 4 部分，由需求覆盖率=测试通过需求点/需求总数*100%可得需求覆盖率为 80%。

表 7.7　测试覆盖分析

Table 7.7　Test coverage analysis

功能	测试用例数	执行数	未执行数	通过数	失败数
订单管理	6	5	1	4	1
立即抢单	4	3	1	3	0
活动管理	4	2	2	2	0
消息管理	5	4	1	4	0

由表格中的测试用例及最后的分析得出，虽然本次的测试数据不够充分，但测试的预期目标及基本功能已经实现，测试功能模块通过率高，未发现大的系统缺陷。

第八章 总结

8.1 结论

紧张而有序的毕业设计终于接近了尾声，在这次的毕业设计中，我们有苦有甜， 不管结果如何，这次毕业设计的过程，在我们人生道路上，也算是一次不菲的收获。除了我们在专业技术水平上的提升外，更大的收获是我们团队成员之间的协作能力以及共同解决困难的方法思路，对我们未来的职场道路来说，都是极其宝贵的经验，可以说是我们自己给自己的人生上了想当重要的一课。

在系统的分析设计中，我们对我们以往学习到的专业知识得到了巩固与提高，并能熟练地运用我们的专业知识去详细地设计一个成熟的系统，这些都是长久以来积淀的成果。虽然可能在系统的分析设计中还会有很多的不足和缺陷，但是以目前的成果来看，我们的系统分析设计是相当成功的，至少我们的产品是成熟稳定可靠可上架的。而且在市场上也是具有一定的竞争力，如果能在市场上投入使用，一定会引起一阵轩然大波。

在系统的实现过程中,我利用当下最前沿的技术,Angular2 前端框架进行了项目的搭建，虽然我以前没有用过这项新技术，但是我有信心去学习并使用它，我有一颗追求新技术的狂热的心，事实证明，我的选择是明智而且非常正确的，新技术带来的往往是设计思路上的革新，并不简简单单的是代码上的变化。通过这次对 Angular2 前台开发框架的使用，我对前端的架构有了深刻的认识，我从以往零散的前端设计思路转变到了模块化的系统设计架构思想上来，让我对代码框架代码结构有了新思路。不得不说，在新技术上的使用，让我加快了系统的开发速度，也让我能更多地将精力放在系统业务逻辑的实现上，而不浪费大量时间去思考如何构建一个项目。

除了系统的实现，我在这次小组团队开发中更是学到了宝贵的人生经验，我们的项目是由三个人的小组来共同完成的。因为有了团队合作，使得我们系统的实现有了明确的分工，各司其职的同时有了困难或者问题可以我们以小组的形式进行讨论攻克，不仅仅我们每个人都学习到了类似问题的解决办法，同时我们还能学到优秀的分析问题解决问题的方法思路，对于每个人来说都是宝贵的提升。

8.2 展望

我们的系统经过艰苦卓绝的奋斗终于接近了尾声，在系统功能逐步完善最后投入使用的关键时期，我仍然非常清醒的告诫自己，我们的项目还存在很多设计上的不足以及存在的缺陷。这些在后续的维护中会逐步完善，而且这些设计上的不足，不仅是我们系统未来的发展空间，更是未来其他系统设计道路上的垫脚石。

我认为社区模块和我们的系统主要业务功能联系还不是很紧密，至少在设计功能和业务功能完成的基础上来宏观地看，这两块的功能应该更加紧密地结合在一起，这样才能更加地接近我们的设计初衷，也更加让我们的社区 020 得到良好的实践。

在数据库的设计方面，我认为还有很多差强人意的地方，我们对于数据库的改动太多频繁都是对我们数据库设计不能很友好的具体直观体现，尽管我们对数据库的设计进行了很多次的改动，然而对我们的数据库而言，我仍然不是十分的乐观和满意，我认为在数据库的设计上，范式的使用还是比较少的。尽管当初为了程序的扩展性我们减少了很多的范式的使用，但是在后续的开发中，我发现有很多的范式实现还是必不可少的，在今后的设计中，我会格外地看中这一点，尽可能让最初的数据库设计更加完善，更加合理。

除了设计方面，代码上的希冀便是对代码结构上的统筹规划，我认为有一个良好的代码结构规范对于后期的维护是极有好处的，在这方面，我们做的是很好，但是还有一点不好的地方就是一开始对于代码的分类并没有做的很好，导致后面的代码文件存储路径比较混乱，对于文件的引用和使用都是极其不方便的。

在团队合作上我认为我们的团队成员之间的沟通还明显不足，任何成功的项目背后都隐藏着无数次的沟通，如果没有充足的沟通，我们团队成员之间的好思路，好想法不能及时地被其他成员所了解，同时个人也是很难去独立完成的，长久以往，我们就会错失很多的好思想，好机会，对于个人还是团队来说都是很大的损失。

我希望在以后的人生道路上，我能多多吸取每一次的经验和教训，并且好的方面吸收，将不好的方面及时改进融合，我坚信我能在日后的道路上走的更加的长远。

参考文献

[1] 徐孝凯 贺佳英. 数据库基础与 SQL Server 应用开发. 北京：清华大学出版社. 2008
[2] 明日科技. SQL SERVER 从入门到精通. 北京：清华大学出版社. 2012
[3] 解本巨. LINQ 从基础到项目实战. 北京：化学工业出版社. 2010
[4] 张亚飞. jQuery 全能权威指南. 北京：清华大学出版社. 2012
[5] David Flanagan. JavaScript 权威指南. 北京：机械工业出版社. 2012
[6] Nicholas C.Zakas JavaScript 高级程序设计. 北京：人民邮电出版社. 2012
[7] 小问. 实战 ES2015-深入现代 JavaScript 应用开发. 北京：电子工业出版社. 2016
[8] 徐孝凯 贺佳英. 数据库基础与 SQL Server 应用开发. 北京：清华大学出版社. 2008
[9] 陈会安. JavaScript基础与实例教程[M]. 北京：中国电力出版社，2007.
[10] 张越廷 顾彦玲. ASP.NET 从入门到精通. 北京：清华大学出版社. 2008
[11] 蒋金楠. ASP.NET MVC4 框架揭秘. 河北：工业大学出版社. 2013

附　　录

附录 1　订单控制器接口代码

```
public class OrderController : WebApiControllerBase
    {
        public IHttpActionResult Get(int queryId, int orderCategoryId, int orderStatusI
d, string publisherOrReceiverLoginId, string orderDescription, int pageIndex, int pageS
ize, bool isDesc)
        {
            tb_OrderQueryObject queryObject = new tb_OrderQueryObject();
            queryObject.queryId = queryId;
            queryObject.orderCategoryId = orderCategoryId;
            queryObject.orderStatusId = orderStatusId;

            if (queryId != -1)
            {
                tb_UserAccountInfo userAccountInfo = UserController.GetUserAccountInfoB
yLoginId(publisherOrReceiverLoginId);
                queryObject.publisherUid = userAccountInfo.uid;
                queryObject.receiverUid = userAccountInfo.uid;
            }

            queryObject.orderDescription = orderDescription;//fuzzy query
            queryObject.PageIndex = pageIndex;
            queryObject.PageSize = pageSize;
            queryObject.IsDESC = isDesc;
            using (var fact = Wcf<OrderService>())
            {
                int count = 0;
                var channel = fact.CreateChannel();
                List<tb_Order> orderList = channel.QueryAllPaging<tb_Order, DateTime>(q
ueryObject, t => t.publishTime).Cast<List<tb_Order>>(out count);
                List<OrderViewModel> resultList = new List<OrderViewModel>();
                foreach (var item in orderList)
                {
                    OrderViewModel orderViewModel = new OrderViewModel();
                    orderViewModel.orderUid = item.orderUid;
                    orderViewModel.publisherUid = item.publisherUid;
                    orderViewModel.publisherInfo = UserController.GetUserAccountInfoByU
idAllowNull(item.publisherUid);
```

```
                    orderViewModel.publishTime = item.publishTime.ToDateTimeString_24Ho
urType();
                    orderViewModel.orderDescription = item.orderDescription;
                    orderViewModel.orderCategoryId = item.orderCategoryId;
                    orderViewModel.orderCategory = item.tb_OrderCategory;
                    orderViewModel.receiverUid = item.receiverUid;
                    orderViewModel.receiveTime = item.receiveTime.ToString("yyy-MM-
dd hh:mm:ss");
                    orderViewModel.orderStatusId = item.orderStatusId;
                    orderViewModel.orderStatus = item.tb_OrderStatus;
                    orderViewModel.orderValue = item.orderValue;
                    orderViewModel.allowVoucher = item.allowVoucher;
                    orderViewModel.voucherMax = item.voucherMax;
                    orderViewModel.evaluateUid = item.evaluateUid;
                    orderViewModel.orderEvaluate = item.tb_OrderEvaluate;
                    orderViewModel.address = item.address;
                    orderViewModel.phone = item.phone;
                    orderViewModel.imageUrls = item.imageUrls;
                    orderViewModel.imageDatas = FilesController.ImageDataArrayByNameArr
ay(item.imageUrls.Split('&'));
                    resultList.Add(orderViewModel);
                }
                return Json(Return_Helper_DG.Success_Msg_Data_DCount_HttpCode("get orde
r list fuzzy query by orderDescription", resultList, count));
            }
        }

        // POST: api/Order
        [LimitsAttribute_DG(RoleLevel = 0)]//user
        public IHttpActionResult Post([FromBody]dynamic query)
        {
            if (query == null)
            {
                throw new Exception_DG("arguments must be provide", 1001);
            }
            Transaction_Helper_DG.Transaction(() =>
            {
                tb_Order order = tb_Order.Build();
                order.orderUid = Guid.NewGuid();

                string loginId = query.publisherLoginId;
                using (var fact2 = Wcf<UserAccountService>())
                {
                    var channel2 = fact2.CreateChannel();
```

```
                    order.publisherUid = channel2.GetUserAccountByLoginId(loginId).uid;
                }
                order.publishTime = DateTime.Now;
                order.orderDescription = query.orderDescription;
                order.orderCategoryId = query.orderCategoryId;
                order.receiverUid = default(Guid);
                order.receiveTime = DateTime.Now;
                order.orderStatusId  = opt_OrderStatus.未支付.ToInt();
                order.orderValue = query.orderValue;
                order.allowVoucher = 1;
                order.voucherMax = 5;
                tb_OrderEvaluate orderEvaluate = tb_OrderEvaluate.Build();
                orderEvaluate.evaluateUid = Guid.NewGuid();
                orderEvaluate.publisherScore = 3;
                orderEvaluate.receiveScore = 3;
                using (var fact = Wcf<OrderEvaluateService>())
                {
                    var channel = fact.CreateChannel();
                    channel.Add(orderEvaluate);
                }
                order.evaluateUid = orderEvaluate.evaluateUid;
                order.address = query.address;
                order.phone = query.phone;
                order.imageUrls = query.imageUrls;

                using (var fact2 = Wcf<OrderService>())
                {
                    var channel2 = fact2.CreateChannel();
                    bool isAdd = channel2.Add(order);
                }
            });
            return Json(Return_Helper_DG.Success_Msg_Data_DCount_HttpCode("order publis
h succeed"));
        }
```

附录 2 订单客户端请求代码

```
import { Component, OnInit } from '@angular/core';
import { appBase } from '../../00-AQX_Frame.commons/appBase';
import { PublishAidModel, Order } from './../order.model';
import { appService } from '../../00-AQX_Frame.services/appService';
import { Router, ActivatedRoute, Params } from '@angular/router';

@Component({
```

```
    selector: 'publish',
    templateUrl: 'app/30-order/publish/publish.component.html',
    styleUrls: ['app/30-order/publish/publish.component.css'],
    providers: []
})
export class PublishComponent implements OnInit {
    /**
     * 发布订单
     */
    OrderPublish(): void {

        var self = this;
        self.order.imageUrls = "";
        for (var i = 0; i < self.imageNameList.length; i++) {
            if (self.imageNameList.length == 1) {
                self.order.imageUrls += self.imageNameList[i];
            } else {
                if (i == 0) {
                    self.order.imageUrls += self.imageNameList[i];
                } else {
                    self.order.imageUrls += ("&" + self.imageNameList[i]);
                }
            }
        }

        if (self.order.orderCategoryId == "") {
            alert("orderCategoryId cannot be null!");
        } else if (self.order.orderValue == "") {
            alert("orderValue cannot be null!");
        } else if (self.order.phone == "") {

            alert("phone number cannot be null!");
        } else {
            $.ajax({
                url: appBase.DomainApi + "api/Order",
                type: "post",
                dataType: "json",
                contentType: "application/json; charset=UTF-8",
                data: JSON.stringify(
                    {
                        "appKey": appService.getCookie("appKey"),
                        "token": appService.getCookie("token"),
                        "publisherLoginId": appService.getCookie("loginId"),
                        "orderDescription": self.order.orderDescription,
```

```
                    "orderCategoryId": self.order.orderCategoryId,
                    "orderValue": self.order.orderValue,
                    "address": self.order.address,
                    "phone": self.order.phone,
                    "imageUrls": self.order.imageUrls
                }),
            success(data) {
                if (data.isSuccess) {
                    alert("订单下发成功~");
                    self.imageNameList = [];
                    //数据清空；
                    self.OrderPublishClear();

                } else {
                    alert(data.msg);
                }
            },
            error(data) {
                alert("服务器错误！");
            }
        });
    }

}
/**
当用户关闭/取消发布模态框时，清除已输入数据；
*/
OrderPublishClear(): void{
    var self = this;
    self.order.orderDescription = "";
    self.order.address = "";
    self.order.phone = "";
    self.imageList = [];
    self.showImg = false;
    if (self.order.orderCategoryId == '6') {
        $('.publish_price').val('0');
        this.order.orderValue = '0';
    } else {
        self.order.orderValue = "";
    }
}
}
```

致　　谢

蚂蚁热帮生活服务系统走进了尾声，在这项目的弥留之际，我还有很多应该做的事情还没有做完，那就是对一直一来对我默默付出的人表示感谢，你们的帮助让我一直在心里挂怀久久不能忘却。

首先我要感谢我的指导教师吴晨思老师，是老师从一开始的选题，包括后面的整个毕设过程，都在孜孜不倦地教导着我，对我的毕业设计提出了宝贵的意见，同时老师也对我毕设过程中遇到的困难给于及时的帮助，是老师的温暖让我有了坚持下去的勇气，能让我顺利地完成毕业设计，谢谢你，我的老师。

其次我要感谢和我同组的项目团队成员，是你们和我一起风雨同舟，经历了毕设的整个过程。谢谢你们在我有困难的时候及时伸出援手，我们一起攻坚克服。正因为有了你们，我的毕业设计中不断融入创新的思维，也正因为有了你们，我的毕业设计的每一个缺陷都被及时地改正，也是你们让我感受到了团队的温暖，这种有组织的感觉让我持续感受着温暖，是你们给了我帮助，也是你们及时的关怀让我有了坚持下去的信心，谢谢你们。

虽然我们的项目已经接近了尾声，但我相信，山外青山楼外楼，强中更有强中手。我们的知识储备在应对未来的人生路上的挑战还远远不够，漫漫人生路，我们还需要有更大的努力、更大的进步才能让我们在未来的道路上稳步向前。生命不息，奋斗不止！