|  |  |  |
| --- | --- | --- |
| 南农大  **数据库课程设计报告**  XH2 | | |
|  | 题 目: | 基于C#的校园多功能互助平台 |
|  | 组 长: | 学号: |
|  | 组 员 | 学号: |
|  |  | 学号: |
|  | 学 院: | 人工智能学院 |
|  | 专 业: | 计算机科学与技术 |
|  | 班 级: | 计科171 |
|  | 指导教师: | 黄芬 职称: 副教授 |
| 2021年5月1日 | | |

目录

[**1 课题分析** 3](#_Toc10280515)

[1.1 课题背景 3](#_Toc10280516)

[1.2 研究目的及意义 3](#_Toc10280517)

[1.3 国内外研究现状 3](#_Toc10280518)

[1.4 本文所做的工作 4](#_Toc10280519)

[**2 开发技术、开发工具及技术路线** 5](#_Toc10280520)

[2.1 开发技术及开发工具 5](#_Toc10280521)

[**2.1.1 开发技术** 5](#_Toc10280522)

[**2.2.2 开发工具** 5](#_Toc10280523)

[2.2 技术路线 6](#_Toc10280524)

[**3 功能分析与设计** 7](#_Toc10280525)

[3.1 需求分析 7](#_Toc10280526)

[**3.1.1 用户调研** 7](#_Toc10280527)

[**3.1.2 初步设计** 7](#_Toc10280528)

[3.2 系统设计 7](#_Toc10280529)

[**3.2.1 系统流程设计** 7](#_Toc10280530)

[**3.2.2 系统功能分析** 8](#_Toc10280531)

[3.3 数据库概念结构设计 8](#_Toc10280532)

[3.4 数据库逻辑结构设计 9](#_Toc10280533)

[**3.4.1 E-R图向关系模型的转换** 9](#_Toc10280534)

[**3.4.2 利用数据依赖的理论对关系模式进行分析** 10](#_Toc10280535)

[3.5 数据库物理结构设计 10](#_Toc10280536)

[3.6 模块划分 10](#_Toc10280537)

[**4 实现** 12](#_Toc10280538)

[4.1 前端界面实现 12](#_Toc10280539)

[**4.1.1 用户界面实现** 13](#_Toc10280540)

[**4.1.2 管理员界面实现** 13](#_Toc10280541)

[4.2 后端代码实现 15](#_Toc10280542)

[**4.2.1 C#与SQL Server的连接和交互** 15](#_Toc10280543)

[**4.2.2 C#后台代码文件以及数据库脚本文件关系图** 17](#_Toc10280544)

[**4.2.3 C#“上传图片到服务器”操作的实现** 17](#_Toc10280545)

[**4.2.4 SQL Server建立数据库** 20](#_Toc10280546)

[**4.2.5 SQL Server触发器的设置** 25](#_Toc10280547)

[**5 系统测试与应用** 28](#_Toc10280548)

[5.1 系统测试运行环境 28](#_Toc10280549)

[5.2 数据库测试与运行 28](#_Toc10280550)

[**5.2.1 用户注册** 28](#_Toc10280551)

[**5.2.2 代取快递任务发布** 29](#_Toc10280552)

[**5.2.3 课本转让任务发布** 29](#_Toc10280553)

[**5.2.4 用户接受任务** 30](#_Toc10280554)

[**5.2.5 用户完成任务** 30](#_Toc10280555)

[**5.2.6 管理员管理用户界面** 31](#_Toc10280556)

[**5.2.7 管理员管理任务界面** 31](#_Toc10280557)

[**6 总结与展望** 32](#_Toc10280558)

[6.1 总结 32](#_Toc10280559)

[6.2 不足 32](#_Toc10280560)

[6.3 展望 32](#_Toc10280561)

[**参考文献** 33](#_Toc10280562)

基于C#的校园多功能互助平台

摘要： 本课程设计报告详细地描述了本小组开发出的一套“校园多功能互助平台”软件，从课题背景入手，分析了开发的目的和意义；接着阐明了本设计的开发技术路线以及功能分析与设计，包括需求分析、系统设计、概念结构设计等；最后依据开发路线和设计实现了互助平台，并对应用程序和数据库进行了测试；

关键词：大学生；互助平台；数据管理；数据上传；C#；SQL server

引言：随着网络技术的发展以及物质生活水平的提高，当代大学生的课余生活得到了极大的丰富，校园生活也变得更加便捷。随着时代的进步，同学们的一些学习上、生活上的小需求也产生了变化，但是并没有一个这样的平台如果我们能够为同学们完成一个多功能互助平台，让同学在平台上发布需求以及提供帮助，那么相信同学们的生活能够更加便捷。

**1 课题分析**

1.1 课题背景

在校园生活中，同学们有时会遇到一些学习、生活上的小需求，希望他人能够代劳，例如快递在很远的站点，希望顺路的人能够帮忙代取；外卖到了但是自己有事脱不开身，想要同学能够帮忙签收；新学期到了课本却买不到，希望能够借用或者购买学长学姐的旧课本；自己有了闲置多余的二手物品想要出售……当前大学生类似这样的需求相当普遍。同学们需要一个平台来发布自己的需求，也可以通过解决他人的需求获得收益。

在同学们想要解决这样的需求，通常借助QQ群、QQ空间、微博等平台来解决这样的需求，这样的方式有着信息的整合度不高，时效性不强，安全性难以保障、个人信息容易泄露的缺点。现在没有一个合适的平台来解决同学们多种多样的需求，于是我们想到能否做出一个多功能互助平台供同学们发布他们的需求，并分门别类地加以整合，与此同时，有的同学如果可以帮助他们解决这些需求，也可以利用这个平台来接受这些任务，来为这些同学提供帮助。在帮助他们的同时，也能为自己赚取收益，不仅节省了被帮助者的时间，提高效率，并且帮助者也可以从中获得物质回报，进而继续促进平台的运作。

1.2 研究目的及意义

本课程设计的目的在于综合运用所学程序设计语言以及数据库原理的知识，开发实现上文所述的多功能互助平台，进行测试并运行，以确保平台的可使用性。在开发的过程中复习所学的知识，并在实践过程中积累项目开发经验。

同时将本平台在同学中进行推广和使用，切实解决同学们各种各样的需求，方便同学们的校园生活，提高生活质量。

## 1.3 国内外研究现状

在国内校园中，有一些网络平台可以实现本平台预期开发的一部分功能，例如微信小程序“木木速跑”，可以实现南京市内部分校区内学生“代取快递”的功能，再例如手机应用程序“零点校园平台”，也可以实现类似的功能并且已经有部分同学开始使用。还有一些应用程序的部分功能与本平台预期开发的功能有一定的重合，例如手机应用程序“饿了么”，其中“跑腿代购”可实现“代取快递”以及“代拿外卖”的功能，再例如我校学生普遍使用的“今日校园”手机应用程序，里面内置了一个“出售二手”的平台，在同学之间有一定范围的使用。

在国外校园，从网络调查中我们得知，由于国外电子商务起步较早，一些面向校园的电子商务平台应运而生，随着时间的推移，这些平台的功能也愈加完善，除了最初“出售二手”的功能以外，也扩展了许多“转让课本”等服务学生的功能。

总而言之，尽管国内外已有不少平台实现了本平台预取开发部分功能或者与部分预期开发功能重合，但是并没有一个完善的，功能丰富的，能覆盖大部分学生需求的平台，因此，本平台的开发有着一定的创新意义以及较大的现实意义。

1.4 本文所做的工作

本文数据库系统的相关理论用于实践，在需求分析以及系统开发的过程中，具体所做的工作有：

1. 在该系统的可行性分析和规划阶段，对于互助平台的基本功能做了简单的分析了解，和实现平台的技术路线。
2. 在该系统的需求分析和总体设计阶段，给出了系统实现的功能模块和基本架构。
3. 在详细设计和具体实施阶段完成了对于互助平台的功能模块设计、功能设计、数据库设计，并完成实现了具体的功能。
4. 在原有的基础上对互助平台进行错误的修改与功能的优化，提高用户使用的便捷性。

**2 开发技术、开发工具及技术路线**

2.1 开发技术及开发工具

2.1.1 开发技术

本平台是基于C#开发的，后台数据库采用Microsoft SQL server。

（1）C#

C#（C sharp）是[Microsoft公司](https://baike.baidu.com/item/%E5%BE%AE%E8%BD%AF%E5%85%AC%E5%8F%B8/732128" \t "_blank)发布的一种面向对象的、运行于[.NET Framework](https://baike.baidu.com/item/.NET%20Framework)和.NET Core（完全开源，跨平台）之上的高级程序设计语言。它是Microsoft公司.NET windows网络框架的主角。其语法风格源自C/C++家族：继承了C的过程化编程语言和C++面向对象思想的强大，融合了Visual Basic的高效，具有易上手、功能强大的特性，一经推出便深受世界各地程序员的好评和喜爱。

C#是一种由事件驱动、可视化的编程语言，结合了[Visual B](https://baike.baidu.com/item/VB)asic简单的可视化操作，以其强大的操作能力、优雅的语法风格、创新的语言特性和便捷的面向[组件](https://baike.baidu.com/item/%E7%BB%84%E4%BB%B6" \t "_blank)编程的支持成为[.NET](https://baike.baidu.com/item/.NET)开发的首选语言。它使得程序员可以快速地编写基于Microsoft .NET平台的应用程序，Microsoft .NET提供了一系列的[工具](https://baike.baidu.com/item/%E5%B7%A5%E5%85%B7" \t "_blank)和服务来最大程度地开发利用计算与通讯领域。

C#使得C++程序员可以高效的开发程序，且因可调用由 C/C++ 编写的本机原生函数，而绝不损失C/C++原有的强大的功能。因为这种继承关系，C#与C/C++具有极大的相似性，熟悉类似语言的开发者可以很快的转向C#。

（2）SQL server

SQL是英文Structured Query Language的缩写，意思为[结构化查询语言](https://baike.baidu.com/item/%E7%BB%93%E6%9E%84%E5%8C%96%E6%9F%A5%E8%AF%A2%E8%AF%AD%E8%A8%80/10450182" \t "_blank)。SQL语言的主要功能就是同各种数据库建立联系，进行沟通。按照ANSI（[美国国家标准协会](https://baike.baidu.com/item/%E7%BE%8E%E5%9B%BD%E5%9B%BD%E5%AE%B6%E6%A0%87%E5%87%86%E5%8D%8F%E4%BC%9A)）的规定，SQL被作为[关系型数据库管理系统](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F/696511" \t "_blank)的标准语言。SQL Server是由Microsoft开发和推广的[关系数据库管理系统](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E7%B3%BB%E7%BB%9F/11032386" \t "_blank)（RDBMS）。

SQL Server是由Microsoft开发和推广的关系型数据库，它最初是由Microsoft、Sybase和Ashton-Tate三家公司共同开发的，并于1988年推出了第一个OS/2版本。目前最新版本是2017年份推出的SQL Server 2017，也就是本课程设计使用的版本。

所谓数据库，就是存储数据的地方，需要长期存储在计算机中。数据库中的数据按照一定的数据模型组织、存储和描述，具有较小的冗余度、较高的数据独立性和易扩展性、可以为不同的用户所共享。

在诸多类型的数据库中，关系数据库由于其建立在严格的数学概念基础之上，概念单一，结构简单、清晰、用户易懂易用，存取路径等物理操作对用户透明等特点，成为当今的最主流数据库技术。

关系数据库是由数据表和表之间的关联组成的。这其中的数据表通常是一个由列和行组成的二维表。每一行乘坐一个元组，每一列称为属性，属性是数据库表中存储对象的共有属性。

对数据库的操作可通过数据库管理系统（Database Management System，简称DBMS）实现，它是数据库的核心组成部分，用户对数据库的一切操作，比如定义、查询、更新、插入、添加触发器等控制都是通过DBMS来进行操作的。Microsoft公司也为SQL Server开发了一套专属的DBMS——SQL Server Management Studio（SSMS），本课程设计使用的版本是17.9.1。

2.2.2 开发工具

本系统在Windows 10操作系统上开发，采用Microsoft Visual Studio 2017集成开发环境进行开发，数据库管理系统采用Microsoft SQL Server Management Studio 17.9.1。

（1）Microsoft Visual Studio

Microsoft Visual Studio，简称VS。VS是美国[微软公司](https://baike.baidu.com/item/%E5%BE%AE%E8%BD%AF%E5%85%AC%E5%8F%B8" \t "_blank)的开发工具包系列产品。VS是一个基本完整的开发工具集，它包括了整个[软件生命周期](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E7%94%9F%E5%91%BD%E5%91%A8%E6%9C%9F" \t "_blank)中所需要的大部分工具，如UML工具、代码管控工具、[集成开发环境](https://baike.baidu.com/item/%E9%9B%86%E6%88%90%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83" \t "_blank)（IDE）等等。所写的[目标代码](https://baike.baidu.com/item/%E7%9B%AE%E6%A0%87%E4%BB%A3%E7%A0%81/9407934" \t "_blank)适用于微软支持的所有平台，包括[Microsoft Windows](https://baike.baidu.com/item/Microsoft%20Windows)、[.NET Framework](https://baike.baidu.com/item/.NET%20Framework" \t "_blank)等。[Visual Studio](https://baike.baidu.com/item/Visual%20Studio)是目前最流行的[Windows](https://baike.baidu.com/item/Windows" \t "_blank)平台应用程序的[集成开发环境](https://baike.baidu.com/item/%E9%9B%86%E6%88%90%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83" \t "_blank)。最新版本为 Visual Studio 2019版本，基于.NET Framework 4.7。本课程设计使用2017版本。

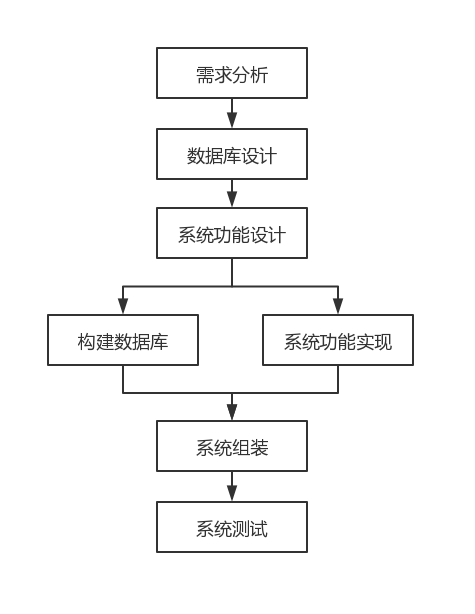
（2）Microsoft SQL Server Management Studio

对数据库的操作可通过数据库管理系统（Database Management System，简称DBMS）实现，它是数据库的核心组成部分，用户对数据库的一切操作，比如定义、查询、更新、插入、添加触发器等控制都是通过DBMS来进行操作的。Microsoft公司也为SQL Server开发了一套专属的DBMS——SQL Server Management Studio（简称SSMS），本课程设计使用的版本是17.9.1。

DBMS可以对[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728" \t "_blank)进行统一的管理和[控制](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6/948689)，以保证[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728)的安全性和完整性。用户通过[DBMS](https://baike.baidu.com/item/DBMS)访问[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728" \t "_blank)中的数据，[数据库管理员](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86%E5%91%98/1216449)也通过DBMS进行数据库的维护工作。它可使多个[应用程序](https://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F/5985445" \t "_blank)和用户用不同的方法在同时或不同时刻去建立，修改和询问[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728)。大部分[DBMS](https://baike.baidu.com/item/DBMS)提供[数据定义语言](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AE%9A%E4%B9%89%E8%AF%AD%E8%A8%80/104009" \t "_blank)[DDL](https://baike.baidu.com/item/DDL/21997)（Data Definition Language）和[数据操作语言](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%93%8D%E4%BD%9C%E8%AF%AD%E8%A8%80/10826383" \t "_blank)[DML](https://baike.baidu.com/item/DML)（Data Manipulation Language），供用户定义[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728" \t "_blank)的模式结构与权限约束，实现对数据的追加、删除等操作。

数据库管理系统是数据库系统的核心，是管理数据库的软件。数据库管理系统就是实现把用户意义下抽象的逻辑数据处理，转换成为计算机中具体的物理数据处理的软件。有了数据库管理系统，用户就可以在抽象意义下处理数据，而不必顾及这些数据在计算机中的布局和物理位置。

2.2 技术路线

本系统在着重分析当前校园二手交易平台、互助平台缺点和不足的基础之上，详细设计了针对校园多功能互助平台的功能，完成诸如代取快递、代拿外卖、转让课本、出售二手等。利用软件工程原理和结构化程序设计思想，结合关系型数据库平台、C#面向对象的程序设计、.NET平台等技术，设计实现了整个多功能互助平台系统。

本平台的开发技术路线如图2-1所示：

图 2-1 开发路线图

**3 功能分析与设计**

3.1 需求分析

**3.1.1 用户调研**

本系统中的用户角色是大学生和平台管理人员，为了能够方便用户操作，让本系统能服务更多的用户，在开发前需要针对实际情况进行详细分析，对各个模块间的跳转进行规划，让系统的功能和用户操作尽可能的符合实际的使用情况。

**3.1.2 初步设计**

根据系统的功能需求分析，系统的主要思路设计如下，这也是我们后续工作设计界面的依据：

（1）用户登录：如果用户没有账号，则点击注册，进入注册界面，注册完成后，

返回登录界面登录，若忘记密码可通过之前保存的手机号找回，用户登录后，可显示登录成功并判断登录的账号是管理员账号还是普通用户账号。

（2）普通用户浏览信息：用户登录后，若为普通用户，则跳转至用户信息界面，可在此界面选择发布任务、接受任务、查看排行榜、查看自己发布的任务与接收的任务并对其进行操作，也能修改之前的个人信息。

（3）管理员用户浏览信息：用户登录后，若为管理员用户，则跳转至管理员操作界面，管理员可修改个人信息，检查所有用户，对指定用户进行封号处理；也可以查看所有任务，关闭指定任务。

（4）用户接受任务：用户可在任务大厅接受任务，发布任务者可看到自己的任务被接受，在接受者完成任务后点击完成按钮，同时输入成交金额。

3.2 系统设计

**3.2.1 系统流程设计**

根据对需求分析阶段所得到的信息进行分析，将系统流程细化到如图3-1所示：

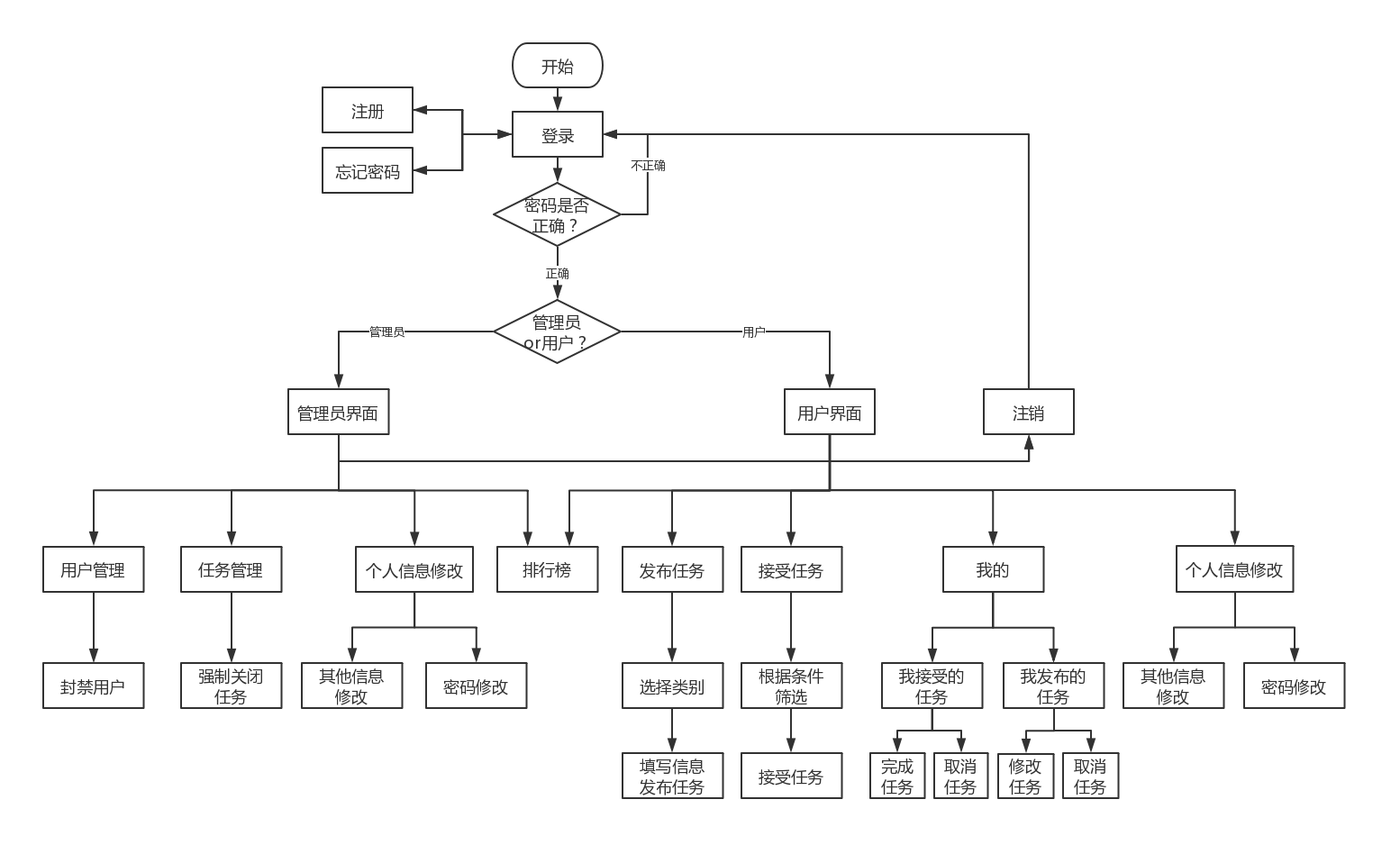


图 3-1 系统流程图

**3.2.2 系统功能分析**

根据上述系统流程分析，系统功能概括为图3-2：

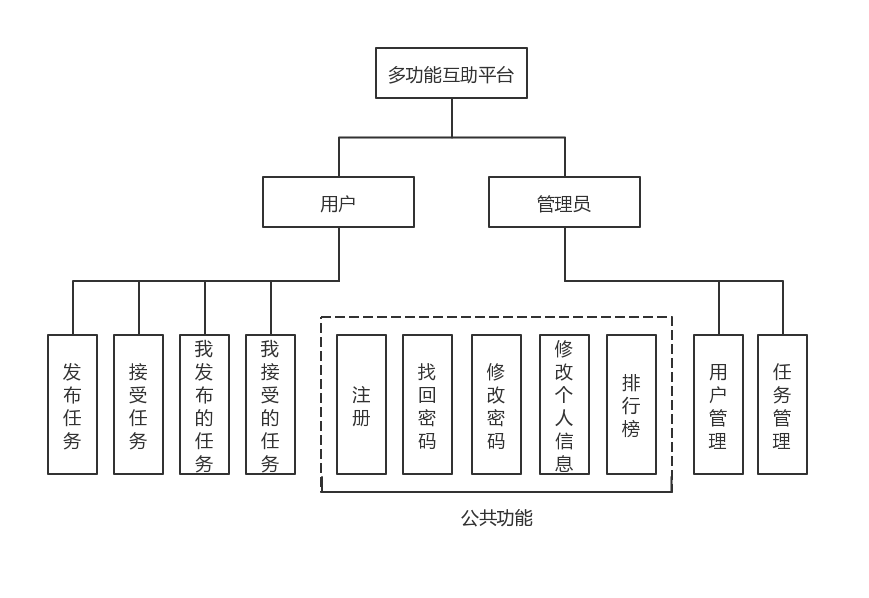


图3-2 系统功能图

3.3 数据库概念结构设计

在需求分析阶段所得到的应用需求应该首先抽象为信息世界的结构，然后才能更好、更准确地用某一数据库管理系统实现这些需求。概念模型是各种数据模型的共同基础。描述概念模型的有利工具E-R模型。

根据需求分析，我们建立了如下的E-R模型，如图3-3所示：

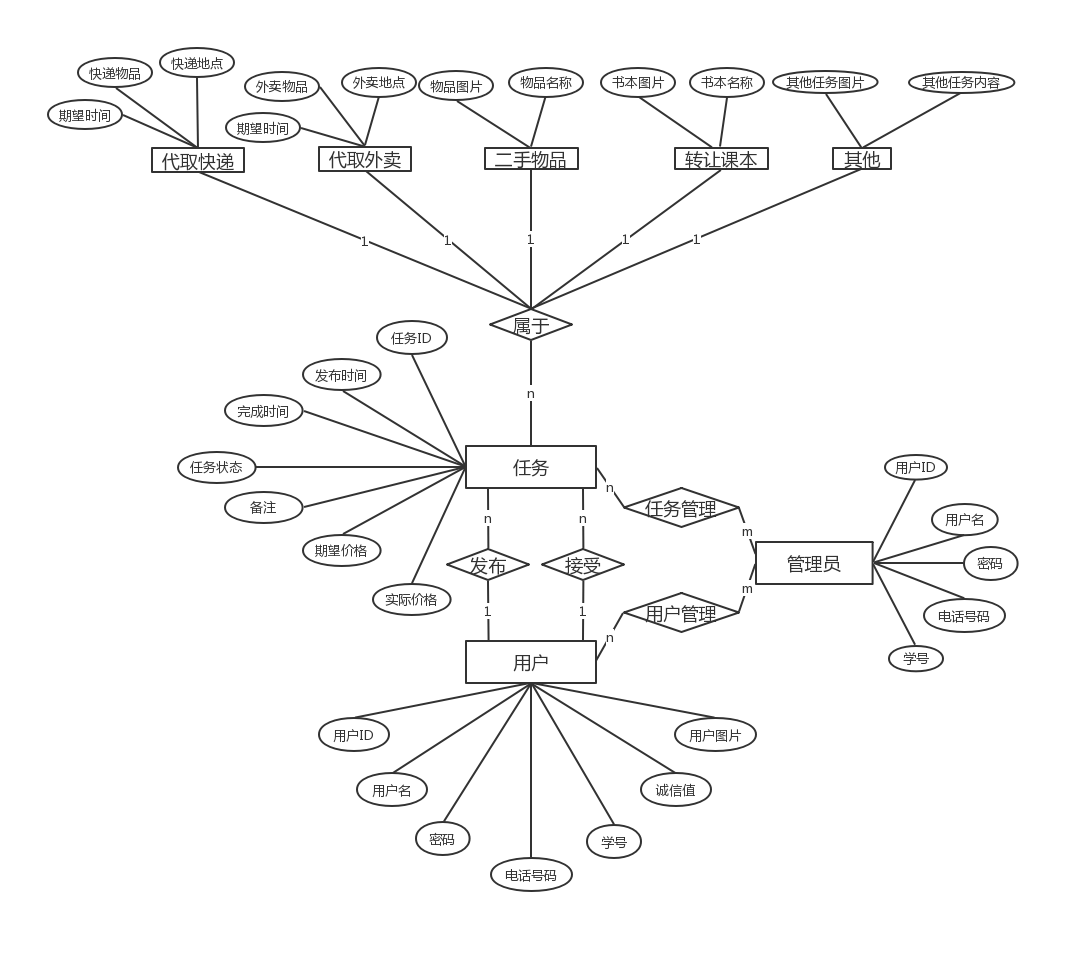


图3-3 E-R模型图

3.4 数据库逻辑结构设计

数据库逻辑结构的设计就是把概念结构设计阶段设计好的基本E-R图转换为与选用DBMS所支持的数据模型相符合的逻辑结构。本课程设计采用的是关系型数据库，因此需要完成E-R图向关系模型的转换。

**3.4.1 E-R图向关系模型的转换**

根据图3-3所示的E-R模型图转换为关系模型，关系的码用下划线标出：

（1）用户信息表（用户ID、用户名、密码、电话号码、学号、诚信值、用户类型）；

（2）任务总表（任务ID、发布者ID、接受者ID、任务类型、发布时间、完成时间、任务状态）；

（3）代取快递任务表（任务ID、快递地点、快递物品、期望时间、备注、期望价格、成交价格）；

（4）代取外卖任务表（任务ID、外卖地点、外卖物品、期望时间、备注、期望价格、成交价格）；

（5）课本转让任务表（任务ID、书本名称、备注、书本图片、期望价格、成交价格）；

（6）出售二手任务表（任务ID、物品名称、备注、物品图片、期望价格、成交价格）；

（7）其他任务表（任务ID、其他任务内容、备注、其他任务图片、期望价格、成交价格）；

**3.4.2 利用数据依赖的理论对关系模式进行分析**

对于上述7种关系模式，我们可以发现每一个非主属性完全依赖于任意一个候选码，并且每一个非主属性不传递依赖于任意一个候选码，所以以上7种关系模式都属于3NF。在数据库的设计中，关系模式满足3NF已经可以满足一定的需求。

3.5 数据库物理结构设计

数据库在物理设备上的存储结构与存取方法称为数据库的物理结构，它依赖于选定的数据库管理系统。为一个给定的逻辑数据模型选取一个最适合应用要求的物理结构的过程，就是数据库的物理设计。

本平台采用的数据库平台是SQL Server，数据库以文件形式存储在计算机磁盘中，其中主要的文件有主数据文件，次数据文件和事物日志文件：

（1）主数据文件

主数据文件是数据库的起点，其中包含了数据库的初始信息，并记录数据库还拥有哪些文件。每个数据库有且只能有一个主数据文件。主数据文件是数据库必须的文件，主数据文件的扩展名是.mdf；

（2）次数据文件

除了主数据文件以外的所有其他数据文件都是次数据库文件。次数据文件不是数据库必需的文件。次数据文件的扩展名是.ndf；

（3）事务日记文件

每个数据库至少拥有一个自己的日记文件（也可以拥有多个日记文件）。日记文件的大小最少是1MB，默认扩展名是.ldf,用来记录数据库的事务以及每个数据对数据库所做的修改。

在SQL Server中，页是数据存储的基本单位，为数据库中的数据文件分配的磁盘空间可以从逻辑上划分带有连续编号的（编号从0开始）磁盘I/O操作在页级执行，SQL Server读取或写入的是所有的数据页。

3.6 模块划分

按照上述分析将系统最终划分为如下4个模块：

（1）用户注册模块：

用户如果希望使用本平台的功能则必须完成注册，需要设定账号，密码，并且上传一张自己校园卡的照片供管理员查验，信息填写完整后即可提交注册申请并使用本平台；

（2）用户/管理员登录模块：

* 用户填写用户账号、密码登录平台，进入用户界面；
* 管理员填写管理员账号、密码来，进入管理员界面；
* 用户/管理员提供“忘记密码”功能帮助找回密码

（3）用户模块：

用户模块又分成如下3个子模块：

a. 发布任务子模块：

用户首先选择自己的需求类别（代取快递，代拿外卖，课本转让，出售二手，其他，共5个类别），进入相应的表单填写相关信息、上传图片（如果有需要）并提交表单；

b. 接受任务子模块：

用户可以根据需求的类别等信息来筛选已经发布的需求，并根据用户指定的方式（例如设定类别，任务状态，发布者ID，是否按发布时间降序等条件）进行筛选排序，并且可以查看需求的详细信息，在任务详细信息界面可以选择接受此需求；

c. 个人信息模块：

* 查看并修改个人信息；
* 查看自己发布的需求，可以进行修改和删除；
* 可以查看自己接受的任务，取消已接受的任务；
* 可以查看自己的信誉值，信誉值是综合用户任务完成比例，取消任务的次数等因素综合计算出来的数值，用于评价用户的信誉水平；
* 查看用户排行榜，如信誉值排行，完成任务数排行等；
* 退出当前账号，回到登录界面；

（4）管理员后台模块：

* 查看已经注册成功的所有用户信息，并且可以根据用户信誉值、校园卡图片等信息对用户进行封号操作
* 查看当前数据库中所有的任务信息，并且可以根据任务内容强制关闭任务；
* 管理员可以修改个人信息以及密码；

**4 实现**

4.1 前端界面实现

本平台前端界面开发主要借助Visual Studio 2017集成开发环境中针对C#窗体应用程序的设计器来完成前端界面的设计与开发工作。开发界面如图4-1所示。

利用这个设计器，可以所见即所得地、便捷地完成窗体的设计，只需要将所需的空间拖拽至窗体，再设置属性即可，设计器将自动地生成后缀名为.Designer.cs的文件，记录窗体以及控件属性信息，在编译时会自动构建窗体，不需要程序员手动编写代码来进行窗体设计，这为我们的开发工作提供了极大的方便。

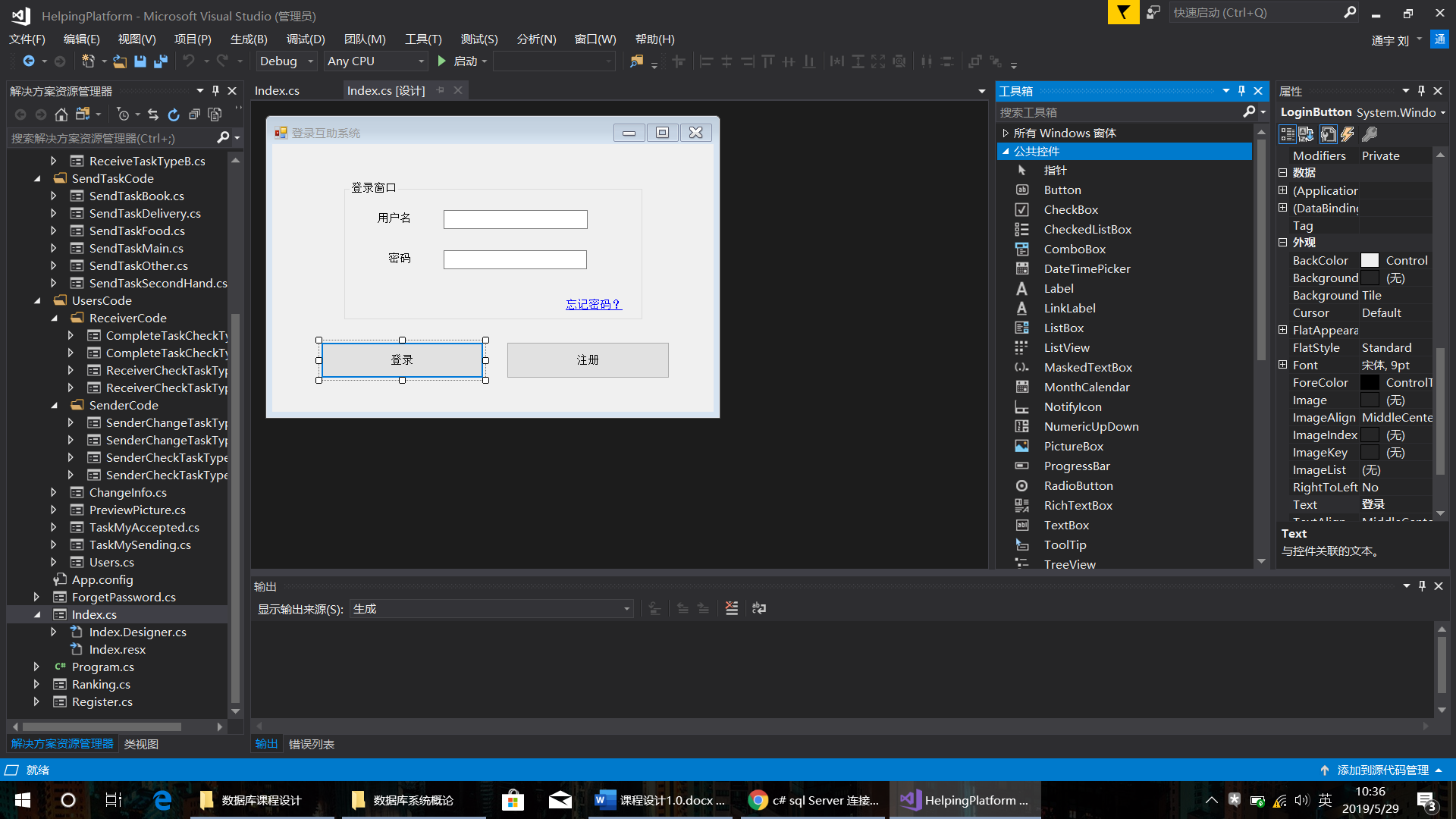


图4-1 前端开发界面示意图

**4.1.1 用户界面实现**

用户界面如图4-2-1至图4-2-11所示，其中图4-2-6发布任务界面以“转让课本”为例，图4-2-8接受任务并确认任务详情界面以“代取快递”种类的一个任务为例，其他界面任务详情界面与图4-2-8类似，图4-2-11排行榜界面用户端和管理员端相同。

**4.1.2 管理员界面实现**

管理员界面如图4-12至图4-14所示，管理员端修改信息，更改密码的界面与用户端相似，排行榜界面与用户端相同，此处均不再赘述。

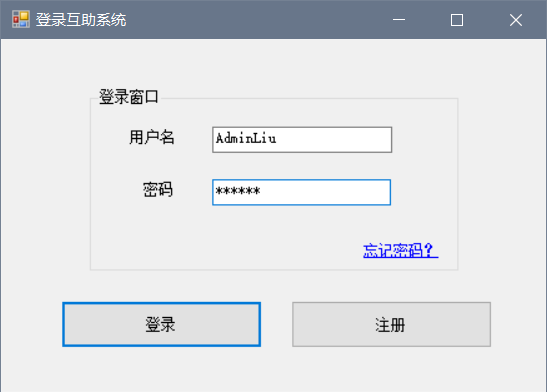
 

图4-2-1 登录界面 图4-2-2 用户注册界面

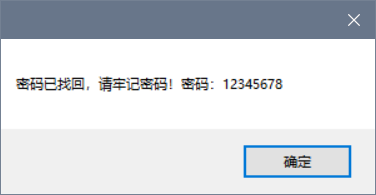
 

图4-2-3 找回密码界面 图4-2-4 找回密码成功界面

图4-2-5 用户端主界面 图4-2-6 发布任务界面

图4-2-7 用户端更改个人信息界面

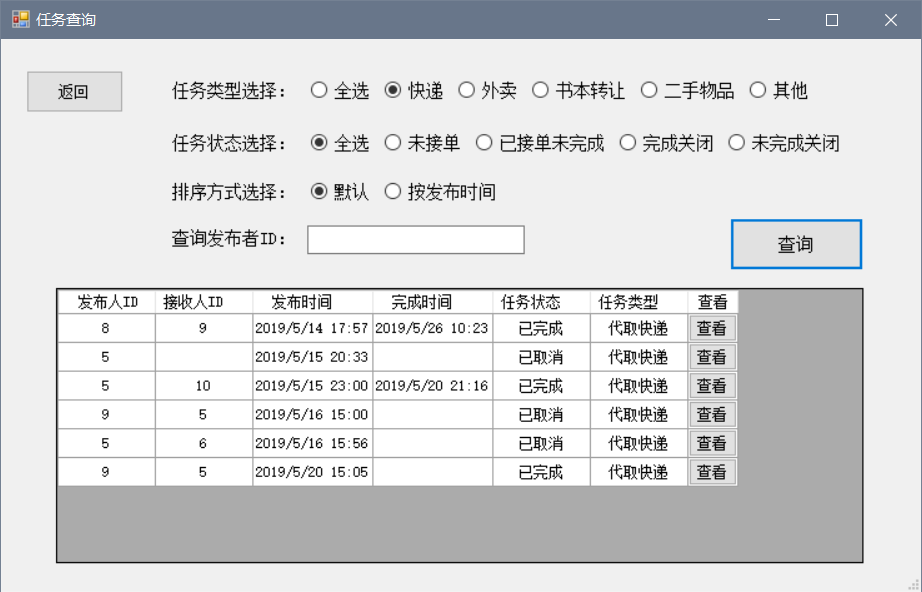
 

图4-2-8 用户端接受任务并确认任务详情界面

图4-2-9 用户端我接受的任务界面 图4-2-10 用户端我发布的任务界面

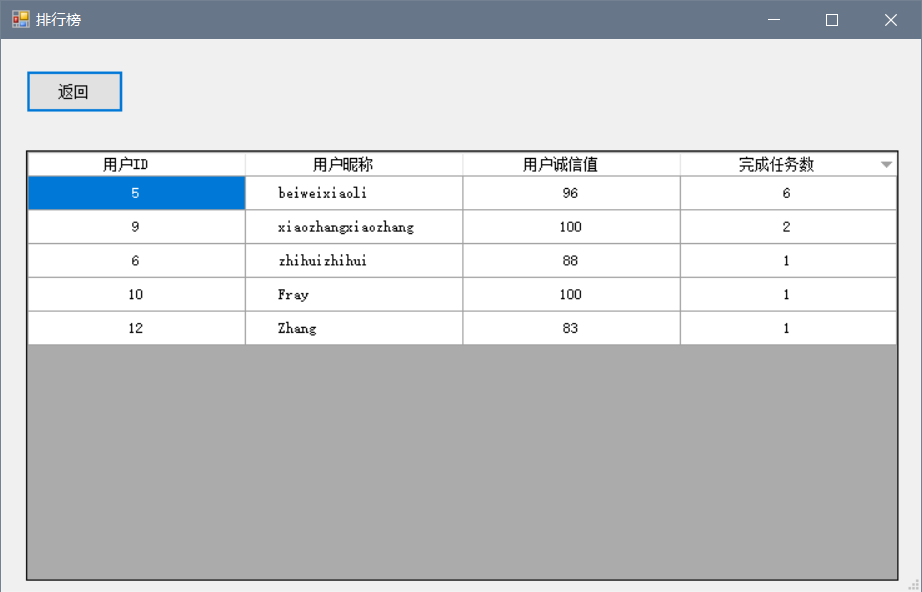
 

图4-2-11 用户端/管理员端排行榜界面 图4-2-12 管理员端主界面

图4-2-13 管理员端用户管理界面 图4-2-14 管理员端任务管理界面

4.2 后端代码实现

由于本平台后端有一定的代码量，因此仅选取核心的和关键的部分代码进行说明。

**4.2.1 C#与SQL Server的连接和交互**

C#是基于.NET平台的，.NET平台提供了许多命名空间，其中允许同本地或远程数据库进行交互式操作时，需要用到的命名空间是ADO. NET，其大部分由System. Data. dll核心程序集来表示，这可以完成应用程序对数据库的不同操作。

ADO. NET支持多种数据源，如果希望与SQL Server进行连接，则需要引入以下命名空间：

using System.Data.SqlClient;

那么我们就可以使用数据提供程序提供的一系列的类来完成与数据库交互的核心功能，一般包括：

（1）SqlConnection类：建立与SQL Server的连接；

（2）SqlCommand类：对数据源执行各种SQL命令；

（3）DataReader类：从数据源中抽取数据（只进、只读数据）；

（4）DataAdapter类：用数据源填充DataSet。

这里以登录界面的按钮的后端代码为例，来说明C#是如何与SQL server进行交互的。

using System;

using System.Windows.Forms;

using System.Data.SqlClient;

……

private void LoginButton\_Click(object sender, EventArgs e)//登录按钮

{

string user = textBox1.Text;

string pwd = textBox2.Text;

//连接数据库，利用SqlConnection类访问Sql Server数据库，设置连接字符串

SqlConnection con = new SqlConnection ("server=.;database=HelpingPlatform;user=AdminHP;pwd=123456");

con.Open();

//利用SqlCommand类对数据库执行SQL命令

SqlCommand cmd = con.CreateCommand();

//拼接字符串，生成SQL语句

cmd.CommandText = "select \* from Users where username='" + user + "'and userpassword='" + pwd + "'";

//利用SqlDataReader逐行读出数据

SqlDataReader dr = cmd.ExecuteReader();

//执行Sql语句后，一定要执行一次Read方法，如果为真，表明已成功读出数据

if (dr.Read())

{

MessageBox.Show("登录成功！用户名：" + dr.GetString(1));

IDUser.SetID(Convert.ToInt32(dr.GetValue(0)));

//判断用户类型

if (Convert.ToInt32(dr.GetValue(6)) == 1)

{

MessageBox.Show("欢迎！用户" + dr.GetString(1));

Users us = new Users();

this.Hide();

us.ShowDialog();

Application.ExitThread();

}

else if (Convert.ToInt32(dr.GetValue(6)) == 0)

{

MessageBox.Show("欢迎！管理员" + dr.GetString(1));

Administrator adm = new Administrator();

this.Hide();

adm.ShowDialog();

Application.ExitThread();

}

}

//未读出数据，表明用户名密码有误

else

{

MessageBox.Show("用户名或密码错误！请重新输入！");

con.Close();

return;

}

con.Close();

}

……

上例中，针对对于数据库的查询操作，读取数据时使用的是DataReader类，这样读取数据库数据是一行一行地进行的，适合于登录界面判断用户名密码是否匹配，注册界面判断用户名是否重复等场合。在本平台的设计中，也用到了DataAdapter类，读取数据库数据是一下读入内存缓冲区，这主要用于需要在界面显示表格的场合，例如筛选任务列表，查看我接受的任务等场合。

另外，在本平台的设计中，有些与数据库交互操作不返回数据库信息，例如更新操作、插入操作，这时候需要使用SqlCommand类中的ExcuteNonQuery()方法，返回受影响的行数以判断操作是否成功。

从上述代码以及分析可见，C#与SQL Server的连接和交互的流程可以概括为如图4-3所示的流程图：

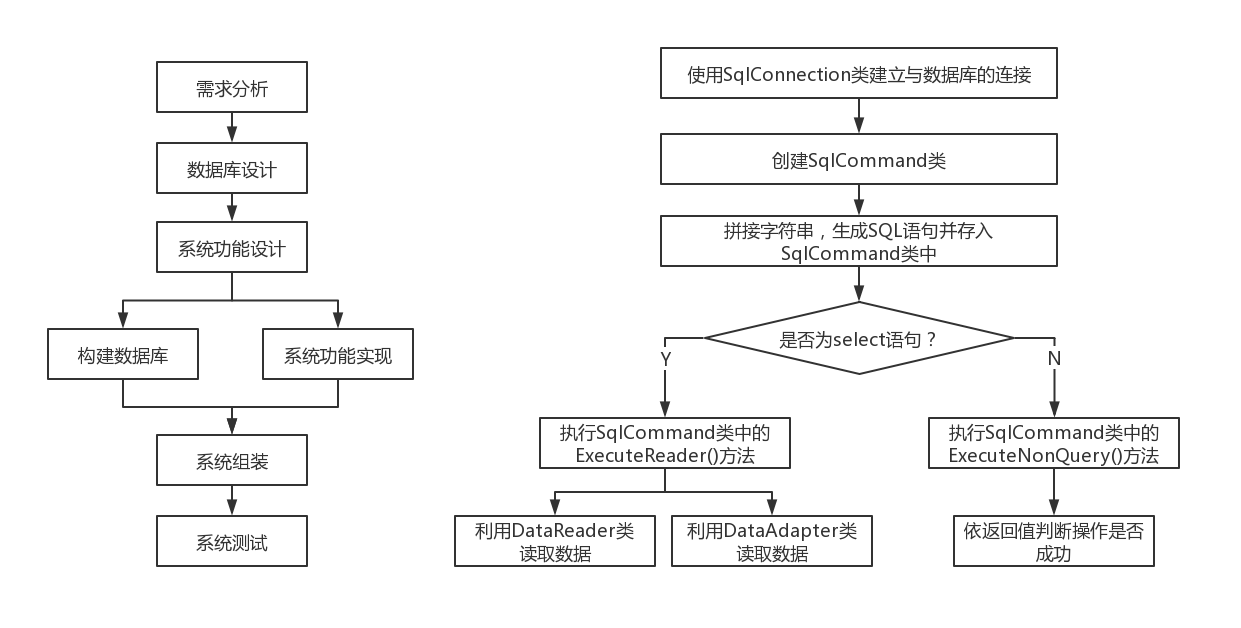


图4-3 C#与SQL Server的连接和交互流程图

**4.2.2 C#后台代码文件以及数据库脚本文件关系图**

C#的源文件的后缀名为.cs，由于源文件数量较多，所以将源文件根据功能进行了分门别类地整理。另外由于本平台是Windows窗体应用程序，所以窗体与源代码有对应关系。图4-4显示了各个源文件之间的关联以及调用逻辑关系。

用于在SQL Server中建立基本表以及触发器的SQL脚本文件共2份，也在图4-4上进行了标注。

**4.2.3 C#“上传图片到服务器”操作的实现**

考虑到租用网络服务器需要支付额外的费用，因此我们选择了以本地磁盘作为我们存储图片的服务器。那么“上传图片到服务器”的操作就简化为将用户选择的图片拷贝到本地磁盘指定的目录中，以模拟上传的操作。

图片文件不能直接存储在数据库中，通常的做法是存储图片文件在服务器的路径，如果需要用到这些图片，应该首先在数据库中查找对应图片的路径，将路径返回程序，再由程序根据路径读取图片，完成图片在前端界面的显示。

这里我们以用户更改信息界面中的上传并修改校园卡图片的按钮的后端代码来说明“上传图片到服务器”是如何具体进行的。

using System;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.IO;

……

private void SubmitButton2\_Click(object sender, EventArgs e)

//确认修改按钮2

{

string newphonenumber = textBox4.Text;

string newstudentnumber = textBox5.Text;

int userid = IDUser.GetID();

//连接数据库

SqlConnection con = new SqlConnection("server=.;database=HelpingPlatform;user=AdminHP;pwd=123456");

con.Open();

SqlCommand cmd = con.CreateCommand();

cmd.CommandText = "select username from Users where userid=" + Convert.ToString(userid);

SqlDataReader dr = cmd.ExecuteReader();

dr.Read();

string username = dr[0].ToString();//读取用户名

dr.Close();

string extension = System.IO.Path.GetExtension(fileName);

string newName = "E:\\HelpingPlatformFiles\\cardPicture\\" + username + extension;//生成新路径

File.Copy(imageFullPath, newName, true);//拷贝文件至新路径

if (File.Exists(newName) == false)//判断是否拷贝成功

{

MessageBox.Show("上传文件失败！请重试！");

con.Close();

return;

}

//进行Sql的update操作

cmd.CommandText = "update Users set phonenumber='"

+ newphonenumber

+ "',studentnumber='"

+ newstudentnumber

+ "',cardpicture='"

+ newName

+ "' where userid=" + Convert.ToString(userid);

int updateflag = cmd.ExecuteNonQuery();

if (updateflag == 1)

{

MessageBox.Show("信息修改成功！\n新手机号码：" + newphonenumber

+ "\n新学号：" + newstudentnumber

+ "\n图片已更新");

}

else

{

MessageBox.Show("修改失败！请重试！");

}

con.Close();

}

从上述代码以及分析可见，本平台“上传图片到服务器”的流程可以概括如下：

① 连接数据库；

② 拼接字符串，设定图片在“服务器”的路径；

③ 将用户所选图片拷贝到设定好的“服务器”的路径；

④ 检查拷贝是否成功，如果成功，将“服务器”的路径存放到数据库中。

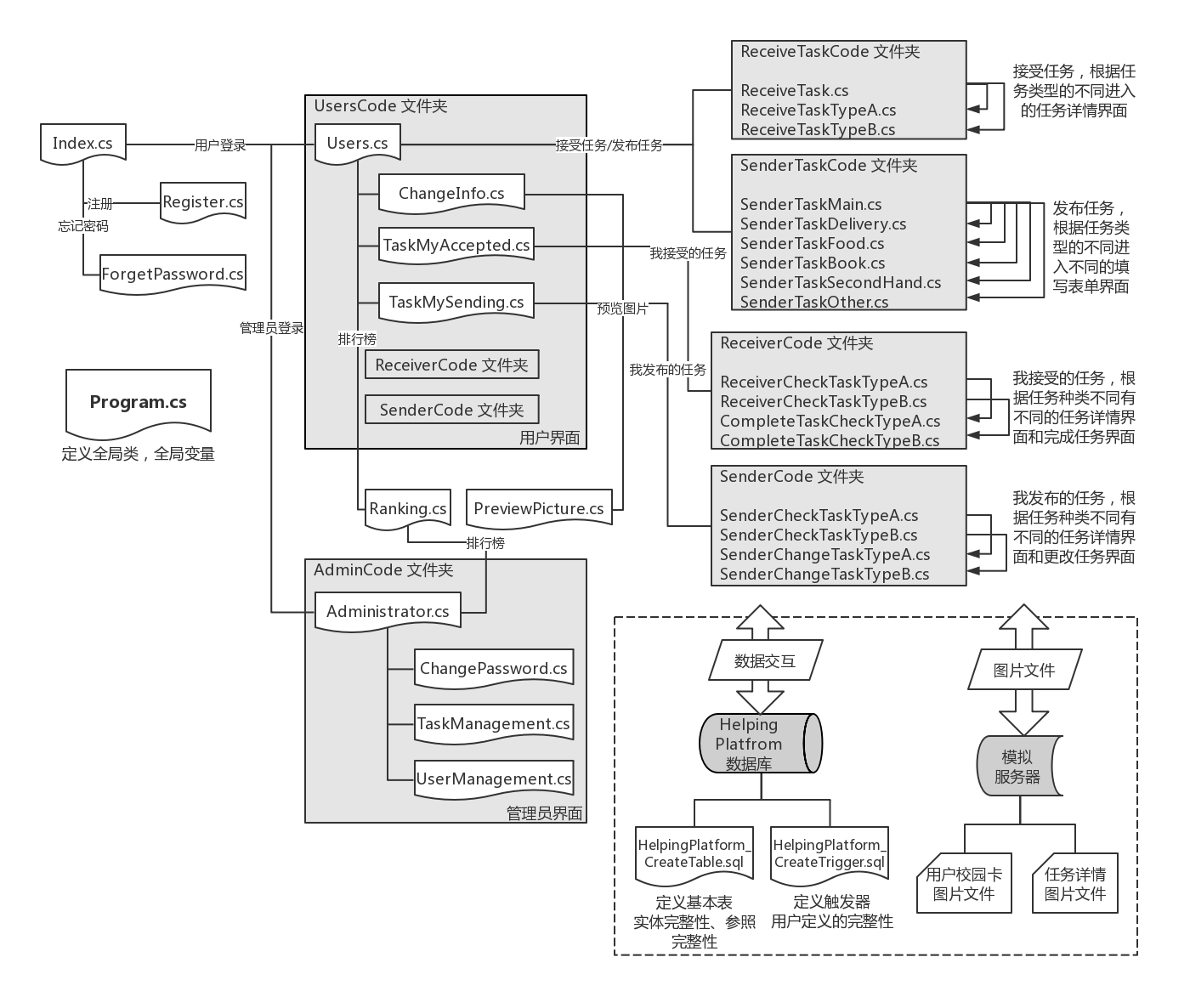


图4-4后台文件关系图

**4.2.4 SQL Server建立数据库**

根据前文所述的关系模型，建立以下7个基本表（表4-1至表4-7）。

表4-1 Users（用户信息表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | userid | 用户ID | int |  | PK | 否 |
| 2 | username | 用户名 | char | 20 |  | 否 |
| 3 | userpassword | 密码 | char | 20 |  | 否 |
| 4 | phonenumber | 电话号码 | char | 15 |  | 是 |
| 5 | studentnumber | 学号 | char | 10 |  | 是 |
| 6 | honesty | 诚信值 | int |  |  | 是 |
| 7 | usertype | 用户类型 | smallint |  |  | 否 |
| 8 | cardpicture | 校园卡图片 | varchar | 200 |  | 是 |

表4-2 Alltask(任务总表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | taskid | 任务ID | int |  | PK | 否 |
| 2 | senderid | 发布者ID | int |  | FK | 否 |
| 3 | receiverid | 接收者ID | int |  | FK | 否 |
| 4 | tasktype | 任务类型 | int |  |  | 否 |
| 5 | releasetime | 发布时间 | datetime |  |  | 否 |
| 6 | finishtime | 完成时间 | datetime |  |  | 是 |
| 7 | taskstate | 任务状态 | smallint |  |  | 否 |

表4-3 Delivery（代取快递任务表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | taskid | 任务ID | int |  | PK  FK | 否 |
| 2 | deliverylocation | 快递地点 | char | 20 |  | 否 |
| 3 | deliverycontent | 快递物品 | char | 40 |  | 否 |
| 4 | deliveryexpecttime | 期望时间 | datetime |  |  | 是 |
| 5 | deliveryremarks | 备注 | char | 100 |  | 是 |
| 6 | expectedprice | 期望价格 | money |  |  | 是 |
| 7 | dealprice | 成交价格 | money |  |  | 是 |

表4-4 Food（代取外卖任务表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | taskid | 任务ID | int |  | PK  FK | 否 |
| 2 | foodlocation | 外卖地点 | char | 20 |  | 否 |
| 3 | foodcontent | 外卖物品 | char | 40 |  | 否 |
| 4 | foodexpecttime | 期望时间 | datetime |  |  | 是 |
| 5 | foodremarks | 备注 | char | 100 |  | 是 |
| 6 | expectedprice | 期望价格 | money |  |  | 是 |
| 7 | dealprice | 成交价格 | money |  |  | 是 |

表4-5 Book（课本转让任务表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | taskid | 任务ID | int |  | PK  FK | 否 |
| 2 | bookcontent | 书本名称 | char | 40 |  | 否 |
| 3 | bookremarks | 备注 | char | 100 |  | 是 |
| 4 | bookpicture | 书本图片 | varchar | 200 |  | 是 |
| 5 | expectedprice | 期望价格 | money |  |  | 是 |
| 6 | dealprice | 成交价格 | money |  |  | 是 |

表4-6 SecondHand（出售二手任务表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | taskid | 任务ID | int |  | PK  FK | 否 |
| 2 | shcontent | 物品名称 | char | 40 |  | 否 |
| 3 | shremarks | 备注 | char | 100 |  | 是 |
| 4 | shpicture | 物品图片 | varchar | 200 |  | 是 |
| 5 | expectedprice | 期望价格 | money |  |  | 是 |
| 6 | dealprice | 成交价格 | money |  |  | 是 |

表4-7 Other（其他任务表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段 | 别名 | 数据类型 | 数据长度 | 码 | 是否可空 |
| 1 | taskid | 任务ID | int |  | PK  FK | 否 |
| 2 | othercontent | 其他任务内容 | char | 40 |  | 否 |
| 3 | otherremarks | 备注 | char | 100 |  | 是 |
| 4 | otherpicture | 其他任务图片 | varchar | 200 |  | 是 |
| 5 | expectedprice | 期望价格 | money |  |  | 是 |
| 6 | dealprice | 成交价格 | money |  |  | 是 |

数据库关系图如图4-5所示：



图4-5 数据库关系图

根据数据库逻辑结构设计，编写SQL语句建立数据库。

具体实现代码如下：

/\*建用户表\*/

drop table if exists Users

create table Users

(

userid int identity(1,1) not null,

username char(20) not null,

userpassword char(20) not null,

phonenumber char(15),

studentnumber char(10),

honesty int,

usertype smallint not null,

cardpicture varchar(200)

primary key(userid)

)

/\*建总任务表\*/

drop table if exists AllTask

create table AllTask

(

taskid int identity(1,1) not null,

senderid int not null,

receiverid int not null,

tasktype int not null,

releasetime datetime not null,

finishtime datetime,

taskstate smallint,

primary key(taskid),

foreign key(senderid) references Users(userid),

foreign key(receiverid) references Users(userid)

)

/\*建代取快递任务表\*/

drop table if exists Delivery

create table Delivery

(

taskid int not null,

deliverylocation char(20) not null,

deliverycontent char(20) not null,

deliveryexpecttime datetime,

deliveryremarks char(100),

expectedprice money,

dealprice money,

primary key(taskid),

foreign key(taskid) references AllTask(taskid)

)

/\*建代拿外卖任务表\*/

drop table if exists Food

create table Food

(

taskid int not null,

foodlocation char(20) not null,

foodcontent char(40) not null,

foodexpecttime datetime,

foodremarks char(100),

expectedprice money,

dealprice money,

primary key(taskid),

foreign key(taskid) references AllTask(taskid)

)

/\*建转让课本任务表\*/

drop table if exists Book

create table Book

(

taskid int not null,

bookcontent char(40) not null,

bookremarks char(100),

bookpicture varchar(200),

expectedprice money,

dealprice money,

primary key(taskid),

foreign key(taskid) references AllTask(taskid)

)

/\*建出售二手任务表\*/

drop table if exists SecondHand

create table SecondHand

(

taskid int not null,

shcontent char(40) not null,

shremarks char(100),

shpicture varchar(200),

expectedprice money,

dealprice money,

primary key(taskid),

foreign key(taskid) references AllTask(taskid)

)

/\*建其他任务表\*/

drop table if exists Other

create table Other

(

taskid int not null,

othercontent char(40) not null,

otherremarks char(100),

otherpicture varchar(200),

expectedprice money,

dealprice money,

primary key(taskid),

foreign key(taskid) references AllTask(taskid)

)

针对上述SQL语句的的说明如下：

（1）在建表前加入drop table if exists 表名的语句，是方便如果建表之后希望对表进行修改，只需要对create table语句进行修改，执行的时候与drop table语句一起执行，DBMS就会删除原来的表再建一个新的表，就可以完成对表的修改。当然对表进行修改可以使用alter table语句，但是这样做不够直观，不能从SQL语句中看出表的结构，由于我们在建表的过程中数据库中没有数据，所以采用了这样的一种方式；

（2）在定义列userid以及列taskid的时候使用了identity(1,1)语句，这条语句的意思是该字段的值会自动更新，其初始值为1，每次递增1。加入这条语句后，这一列数据不需要我们人工进行维护，通常情况下我们不可以直接给identity修饰的字符赋值，否则运行时会报错。这样做的好处是用户ID以及任务ID由数据库自动生成，且不会重复，这样为前端程序的设计提供了方便。

**4.2.5 SQL Server触发器的设置**

由于本平台有设计用户“诚信值”的功能，但是诚信值的计算并不适合在平台程序内实现。原因在于诚信值是根据用户完成任务的数量、取消已接受任务的数量、取消发布任务的数量等诸多指标来综合计算的，具体计算方法如表4-8所示：

表4-8 诚信值计算方法

|  |  |  |  |
| --- | --- | --- | --- |
| 条件 | 发布者  诚信值变化 | 接收者  诚信值变化 | 每个用户初始诚信值90分，上限100分。  诚信值对应等级：  100分 信用优秀  90~99分 信用良好  80~89分 信用一般  79分及以下 信用较差 |
| 完成任务 | +1 | +2 |
| 发布者取消任务 | -2 |  |
| 接收者放弃任务 |  | -2 |
| 管理员关闭任务 | -5 |  |

从上述计算方法上来看，每当任务状态发生变动的时候将会导致诚信值的改变。尽管在平台程序内可以实现这样的计算并将新的诚信值更新至数据库，但是这样容易导致代码的冗余或者对于需要改变诚信值情况的遗漏。

综合上述原因，我们采取数据库技术中的“触发器”来实现诚信值的计算，每当总任务表（table AllTask）的任务状态发生变化时，即触发触发器，根据上述计算方法更新用户表（table Users）中相关用户的诚信值。

具体实现代码如下：

drop trigger CalHonesty

go

create trigger CalHonesty

on AllTask

after update/\*如果总任务表任务状态更新，则触发触发器，这是针对接收任务者的触发器\*/

as

begin

declare @currentid int /\*声明变量记录接收者ID\*/

select @currentid=receiverid from inserted

update Users /\*经过计算更新用户表接收者对应的诚信值\*/

set honesty=90+2\*(select count(distinct taskid)

from AllTask

where receiverid=@currentid and taskstate=3)

+1\*(select count(distinct taskid)

from AllTask

where senderid=@currentid and taskstate=3)

-2\*(select count(distinct taskid)

from AllTask

where senderid=@currentid and taskstate=4)

-2\*(select count(distinct taskid)

from AllTask

where receiverid=@currentid and taskstate=5)

-5\*+(select count(distinct taskid)

from AllTask

where receiverid=@currentid and taskstate=6)

where userid=@currentid

update Users /\*检查诚信值是否大于100，否则置为100\*/

set honesty=100

where honesty>100 and userid=@currentid

end

drop trigger CalHonesty2

go

create trigger CalHonesty2 /\*这是针对接收任务者的触发器\*/

on AllTask

after update

as

begin /\*与上一个触发器基本相同，只是@currentid记录的是接收任务者的ID\*/

declare @currentid int

select @currentid=senderid from inserted

update Users

set honesty=90+2\*(select count(distinct taskid)

from AllTask

where receiverid=@currentid and taskstate=3)

+1\*(select count(distinct taskid)

from AllTask

where senderid=@currentid and taskstate=3)

-2\*(select count(distinct taskid)

from AllTask

where senderid=@currentid and taskstate=4)

-2\*(select count(distinct taskid)

from AllTask

where receiverid=@currentid and taskstate=5)

-5\*+(select count(distinct taskid)

from AllTask

where receiverid=@currentid and taskstate=6)

where userid=@currentid

update Users

set honesty=100

where honesty>100 and userid=@currentid

end

针对上述SQL语句的的说明如下：

（1）SQL Server中对于触发器的语法比较特别，需要注意：

SQL Server在定义触发器的时候，前面不允许有其他语句，需要利用go语句将与其强行隔开；

SQL Server不支持触发方式为before的触发器，默认触发器是行级别触发器，不能使用for each row语句说明这是行级别触发器；

SQL Server触发器的动作体应该写在as begin … end之间；

（2）SQL Server可以通过declare 变量名 变量类型语句声明变量，变量名规定应该由@开头；

（3）SQL Server内置了一个逻辑表inserted，这个表用于存放数据库进行insert、update、delete语句后更新的记录，因此可以通过这个逻辑表，找到任务表中相关的用户ID，进而利用这个用户ID完成用户表中诚信值的修改。

**5 系统测试与应用**

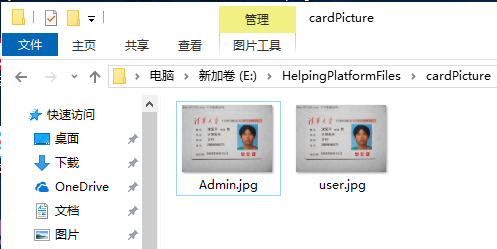
5.1 系统测试运行环境

测试系统软件环境：操作系统为Windows 10并且已经安装.Net Framework 4.6.1，已经安装数据库SQL Server 2017和SQL Server Management Studio 17.9.1；

测试系统硬件配置：处理器Intel Core i5-7200U，内存8G，外存256G固态硬盘，显示适配器AMD Radeon R7 M440。

5.2 数据库测试与运行

**5.2.1 用户注册**

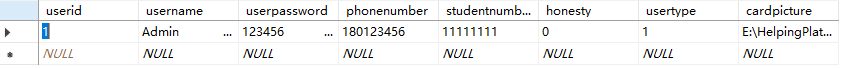


图5-1 用户端注册界面、传入数据库的数据以及“服务器”文件

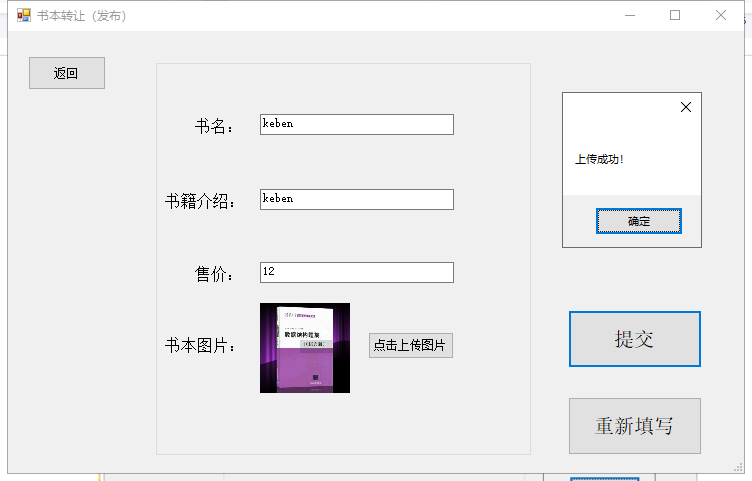
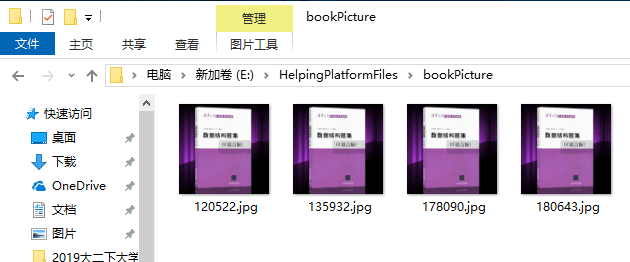
**5.2.2 代取快递任务发布**





图5-2 代取快递任务发布界面及传入数据库的数据

**5.2.3 课本转让任务发布**

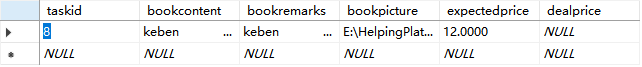


图5-3 课本转让任务发布、传入数据库的数据以及“服务器”文件

**5.2.4 用户接受任务**





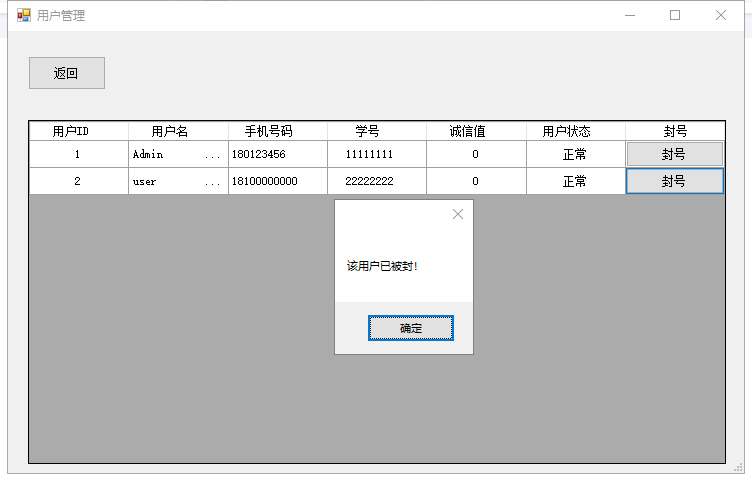
图5-4 用户接受任务界面及传入数据库的数据

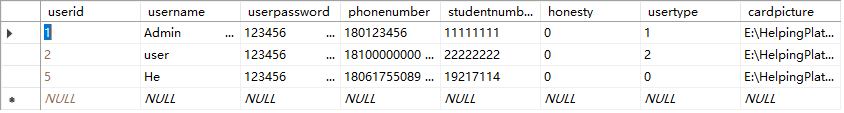
**5.2.5 用户完成任务**



图5-5 用户完成任务界面及传入数据库的数据

**5.2.6 管理员管理用户界面**



图5-6 管理员管理用户界面及传入数据库数据

**5.2.7 管理员管理任务界面**



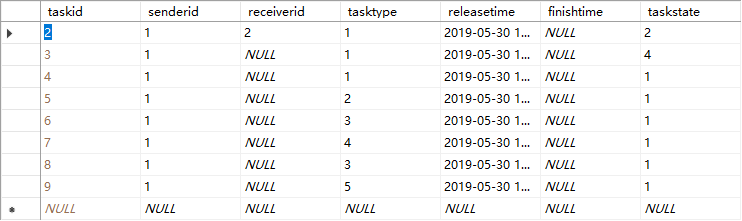


图5-7 管理员管理用户界面及传入数据库的数据

**6 总结与展望**

6.1 总结

本系统在着重考察研究目前大学生生活学习习惯的基础之上实现了如下核心功能。

* 用户注册登录：只有在系统内存在的用户才可进行登录操作，忘记密码的用户可根据注册时的手机号重新修改密码。
* 发布任务功能：用户登录后可以发布代取快递、代取外卖、转让课本、二手物品、其他共5类任务，以便供他人接单帮助自己完成任务。
* 接受任务功能：用户登录之后可以前往任务大厅，根据自己的情况筛选想要接受的任务。
* 用户中心：用户在登录后可以在用户中心修改个人信息、修改密码、查看自己接受与发布的任务，以及实时的接单排行榜。
* 管理员中心：管理员登录后可对所有用户进行封号操作，对任务进行关闭操作。

通过对本课题的研究以及原型系统的设计与开发，在对数据库理论的知识有了更深的理解，能更加熟练地使用SQL语句进行，同时也学习了C#的基本语法并可以使用C#完成基础的软件开发。在开发实践的过程中，本小组成员分明确，积极合作，相互纠错，在交流中互相提高。在开发系统的过程中，我们也出现了许多不曾预料到的bug，但总能在团队合作下，通过查阅资料、网上搜索等方式将问题完美解决。这次课程设计不仅提高了小组成员的编程能力，也意识到了团队合作的重要性，对软件的开发过程也有了一定的理解和认识。

6.2 不足

目前，本系统的大体核心功能都基本实现，但是整个系统的应用上来说还是有很多的不足，包括：

* 用户输入期望时间的操作需要按照指定格式输入，用户如果不按照规定的格式输入时间则会报错，用户使用上则会产生一些不便；
* 接受任务、我接受的任务和我发布的任务界面的列表无法自适应填充列表宽度，只能做到将列宽调整为能使列表内容完全显示的宽度，尽管不影响使用但是界面显得不美观；
* 管理员端对于任务的管理仅限关闭任务，对于用户的管理也仅限封号，未能扩充更多的功能；
* 并未真正地实现将图片上传到网络服务器，平台暂时只能在本地运行，未能上线投入真正的使用；
* 整体界面设计有待改进，可以设计得更加美观。

6.3 展望

本系统面向大学学生这一个群体，希望能为广大学生节省时间、提高效率，同时也能在完成任务的同时赚取一份收入，在同学与同学之间实现互惠双赢。

本系统是对于“多功能互助平台”的一次尝试，仅仅只能做出一个雏形。希望今后能以本次课程设计为基础，在已有的工作成果之上优化平台的功能，美化前端界面，争取能够真正上线运行，相信这样的多功能互助平台有着美好的前景和极大的潜力。

**参考文献**

[1] 王珊, 萨师煊. 《数据库系统概论（第五版）》 高等教育出版社.

[2] 蔡朝辉, 安向明, 张宇. 《C#程序设计案例教程》 清华大学出版社.

[3] H. M. Deitel, P. J. Deitel, J. Listfield, T. R. Nieto, C. Yeager, M. Zlatkina. C# HOW TO PROGRAM 清华大学出版社.

[4] 何鹏飞, 王征等. 《C#实用编程百例》 清华大学出版社.

[5] 姚永一. 《SQL Server数据库实用教程》 电子工业出版社.

[6] 壮志剑. 《数据库原理与SQL Server》 高等教育出版社.