**青岛农业大学**

**理 学 与 信 息 科 学 学 院**

**求职与招聘系统**

（C#程序设计大作业）

设 计 题 目 基于云数据库的求职与招聘系统

学生专业班级 计科1900

学生姓名（学号） Paimon 20190000000

设计小组其他同学姓名（学号） 无

指 导 教 师 Teacher

完 成 时 间 2021/10/26

设 计 地 点 信息楼112

2021年10月26日

目录

[一、求职与招聘系统设计任务和目的 3](#_Toc86224683)

[1.1任务 3](#_Toc86224684)

[1.2目的 3](#_Toc86224685)

[二、设计要求 3](#_Toc86224686)

[2.1模块要求 3](#_Toc86224687)

[2.2数据库要求 3](#_Toc86224688)

[2.3安全性要求 3](#_Toc86224689)

[三、设计内容 4](#_Toc86224690)

[3.1 登录模块 4](#_Toc86224691)

[3.2 注册模块 5](#_Toc86224692)

[3.3 企业用户主界面 6](#_Toc86224693)

[3.4 （企业）收到的简历页面 7](#_Toc86224694)

[3.5 （企业）需求发布页 8](#_Toc86224695)

[3.6 需求信息详情页 8](#_Toc86224696)

[3.7 求职者用户主界面 9](#_Toc86224697)

[3.8 （求职者）简历回复情况页面 10](#_Toc86224698)

[3.9 （求职者）我的简历 11](#_Toc86224699)

[四、结果与分析 11](#_Toc86224700)

[五、改进和建议 12](#_Toc86224701)

[六、总结 12](#_Toc86224702)

[6.1 感想 12](#_Toc86224703)

[6.2心得 13](#_Toc86224704)

[七、程序测试运行环境 13](#_Toc86224705)

# 一、求职与招聘系统设计任务和目的

## 1.1任务

使用C#，在Visual Studio 2022环境（.NET SDK 6.0）中设计一个完善、安全、稳定且符合现代设计风格的实用性招聘系统。

## 1.2目的

锻炼程序设计实战能力，体现面向实际需求程序设计，掌握灵活变通使用代码解决问题的能力。

# 二、设计要求

## 2.1模块要求

登录模块、注册模块、主界面模块、企业信息发布模块、招聘信息查看模块、应聘信息反馈模块、留言板模块。

## 2.2数据库要求

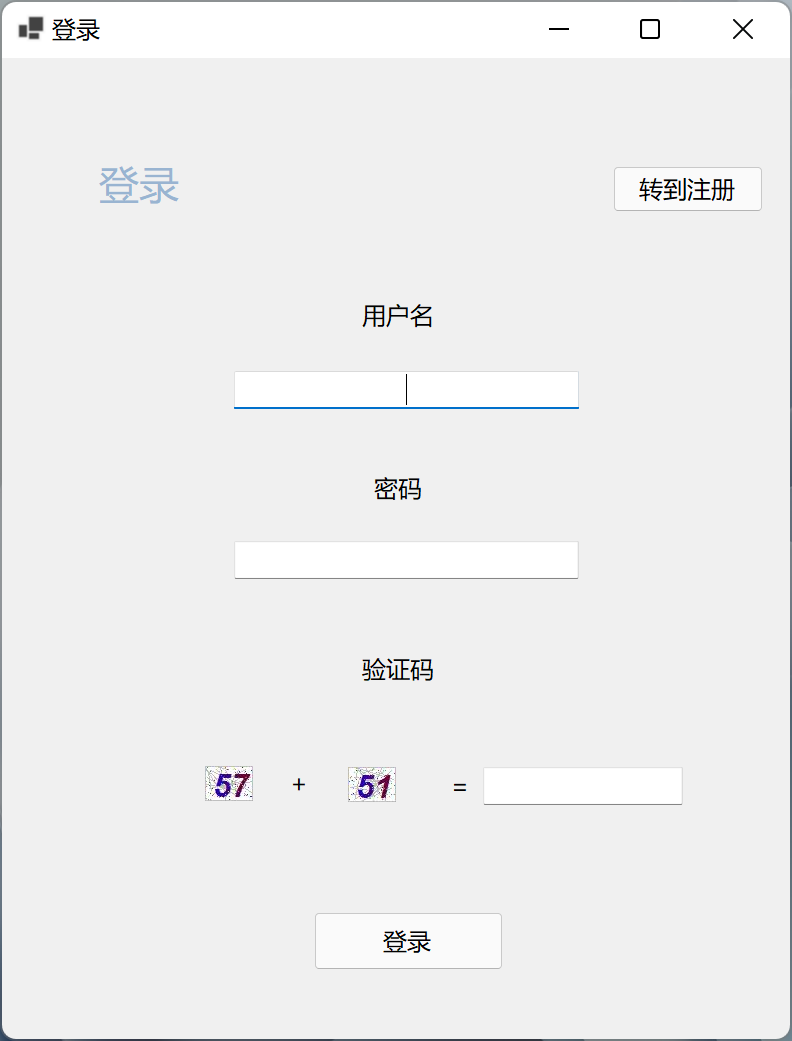
包含五个数据库表，apply（应聘信息）、comment（留言信息）、request（招聘信息）、resume（个人简历表）、user（用户信息表）。数据库使用云数据库MariaDB，以便在不同电脑上都正常运行本程序，该数据库运行在CentOS 8（Linux内核）系统的弹性云计算服务器上，服务器开放3306端口。

## 2.3安全性要求

用户名采用Base64加密算法加密后存放在数据库中，用户密码采用md5加盐存放在数据库中。所有非int类型数据均使用base64加密。向表中插入数据前先检查是否已经存在同id条目以避免意想不到的错误。

# 三、设计内容

## 3.1 登录模块



用户名设计输入为中文，密码可输入字母、数字、符号，验证码使用Drawing库生成图片，为了防止机器人自动登录和爆破密码，增强验证码的真人过滤强度，令用户把两个验证码图片中的数值相加，判断两数之和是否正确，如果正确，则通过验证，执行SQL语句查询数据库 ”SELECT \* FROM USER WHERE name = ‘” + $name + “’ and pw = ‘” + md5(“salt666”+$password+”salt888”) +”’” 如果能查询到记录，证明用户名密码正确，登录成功。

## 3.2 注册模块



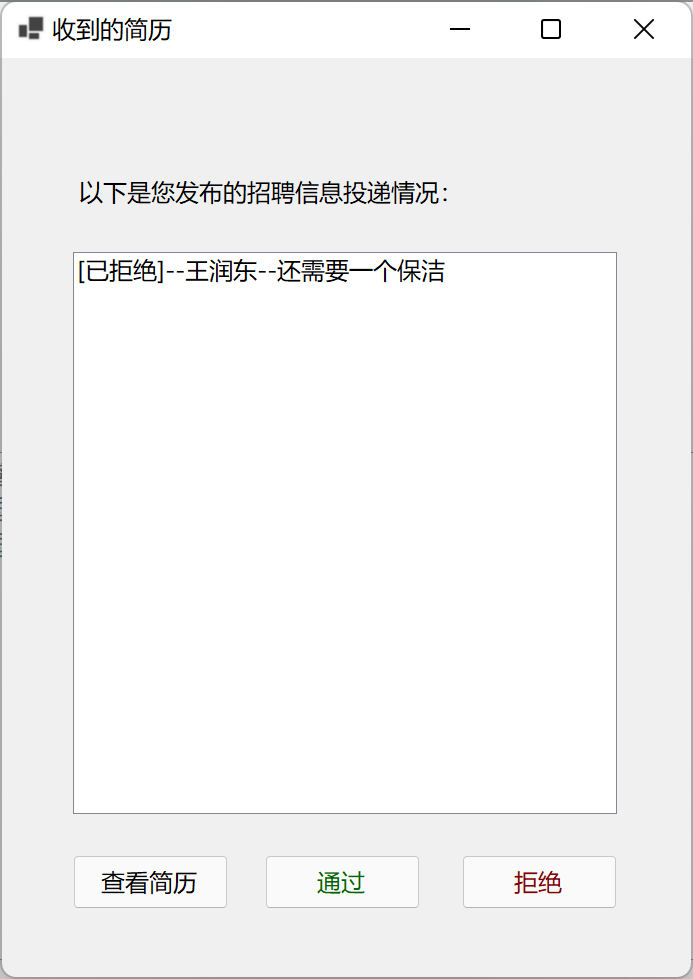
点击提交的时候，先使用正则表达式 "^(?![0-9]+$)(?![a-zA-Z]+$)[0-9A-Za-z]{2,256}$" 匹配密码，判断密码是否长度在2-256之间，是否同时包含数字和字母（最低用户密码安全性要求），再判断两次输入的密码是否相同，如果相同，用户端检测完毕，进入数据库查询环节，如果自定义的用户名再数据库中已被注册（不允许存在重名用户），则提示用户名被占用，令用户更换用户名，如果没有，则插入注册信息到数据库中。其中注册信息，用户名和用户类型两项使用base64加密存储，密码加盐之后取MD5存储。最后自动跳转到登录界面，把刚刚输入的用户名传递进去，用户只需要输入密码即可登录刚才注册的账户。

## 3.3 企业用户主界面



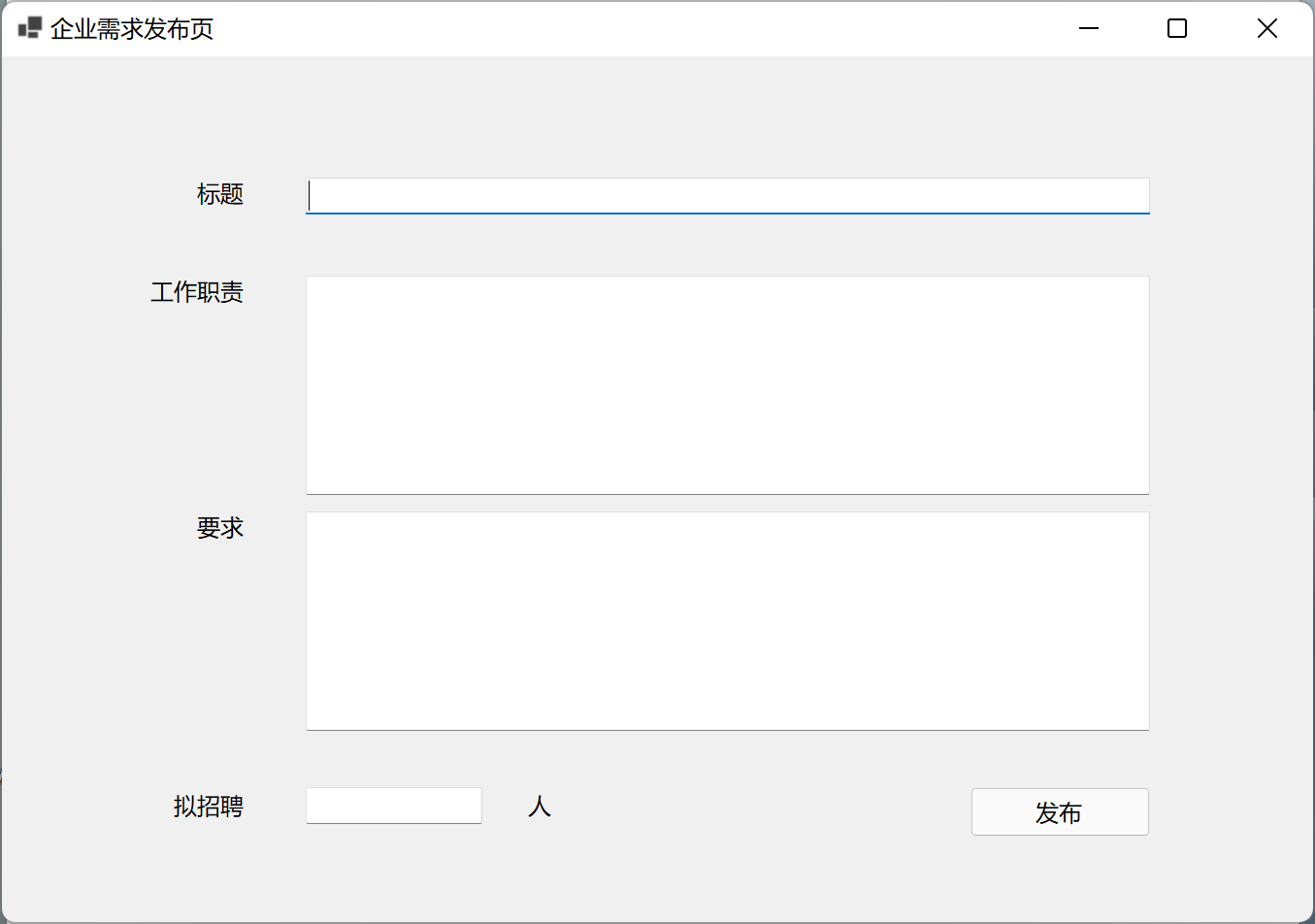
包含注销、收到的简历、发布招聘需求、删除已发布招聘信息、查看招聘信息池详情页。注销按钮点击即可退出当前账号回到登录页面，收到的简历按钮是企业用户独有的功能，可以查看谁给自己发布的需求投递简历了，然后及时处理（通过或拒绝）。发布招聘需求按钮即可新增一个招聘信息，所有用户均可查看并留言。删除按钮只可以删除自己发布的信息。查看按钮即查看选中条目的招聘信息详情，并可以写留言。

## 3.4 （企业）收到的简历页面



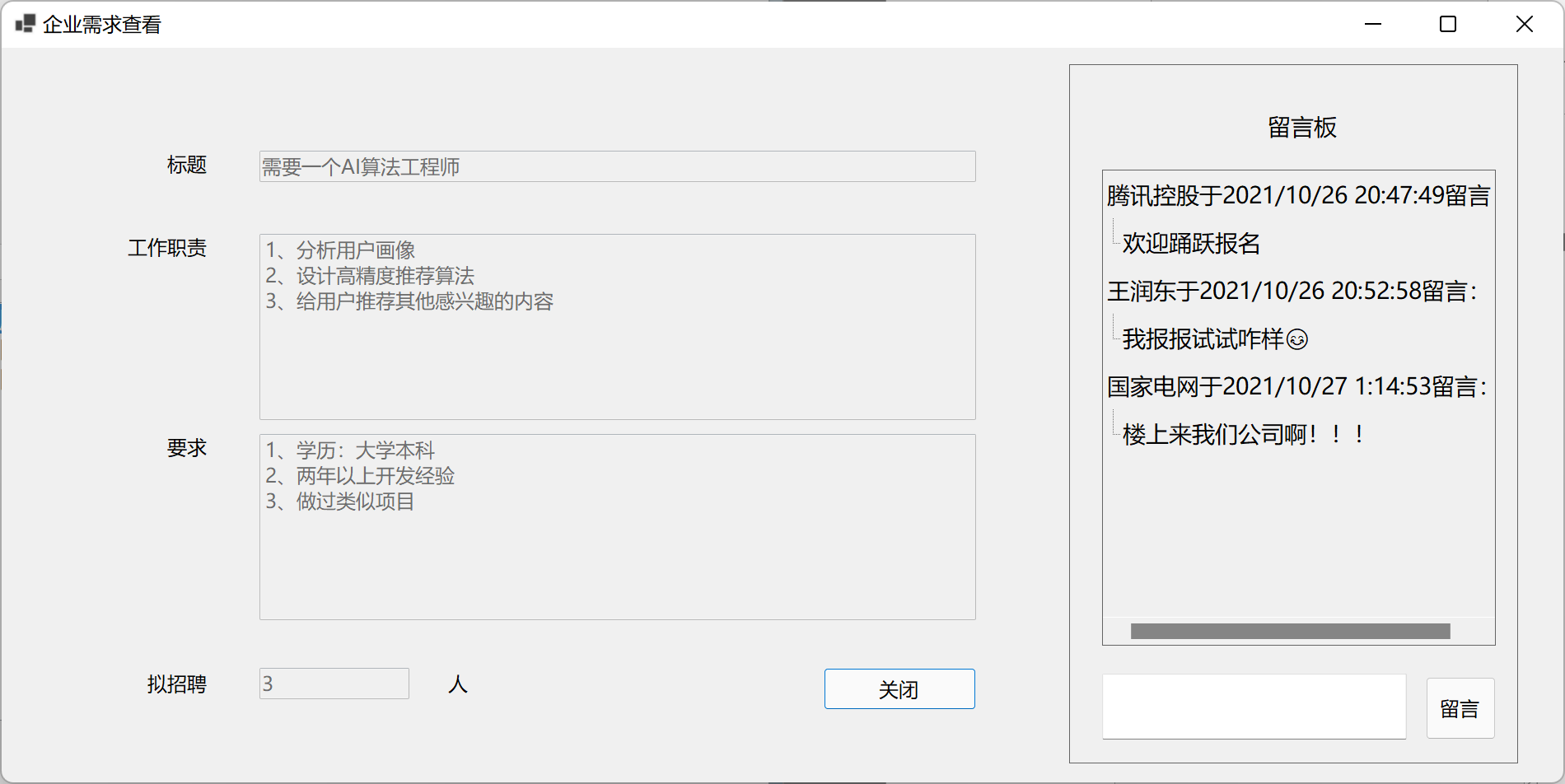
在此页面，如果有人（求职者）想应聘该职位，求职者会发送自己的简历到这里，可以点击查看这个求职者的简历，然后选择通过和拒绝。

## 3.5 （企业）需求发布页



在这里，企业用户可以发布自己的招聘需求，需求信息会被系统使用base64加密算法加密后存放到数据库中。

## 3.6 需求信息详情页



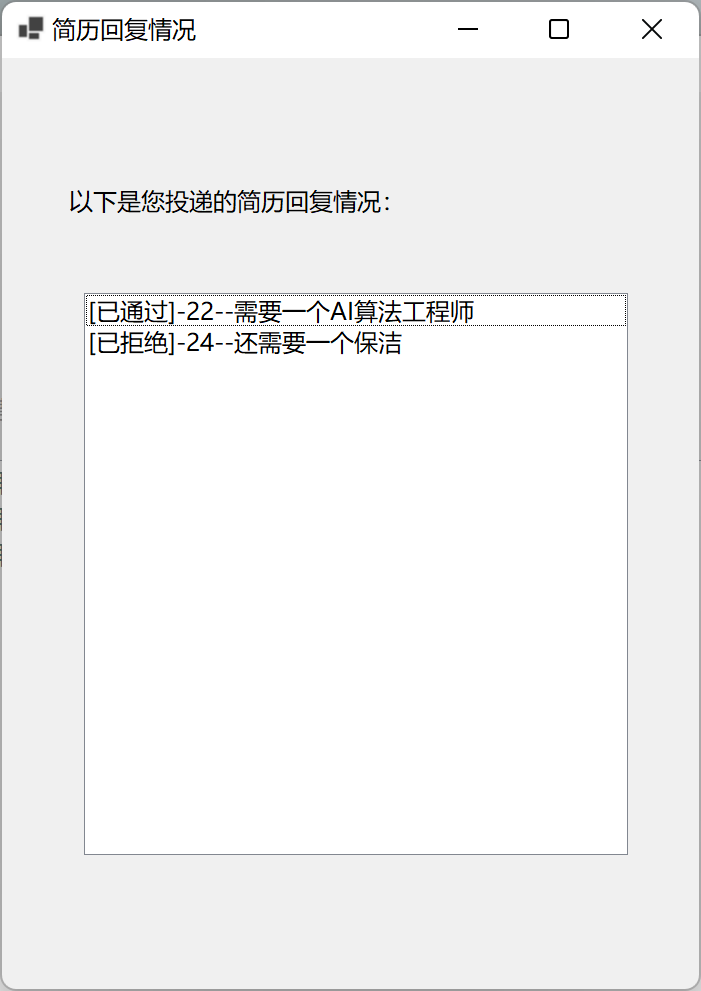
在此页面，任何用户可以查看任何需求的详细信息，并可以进行留言，留言会记录当前用户名和时间，留言内容同样经过base64加密存储。

## 3.7 求职者用户主界面



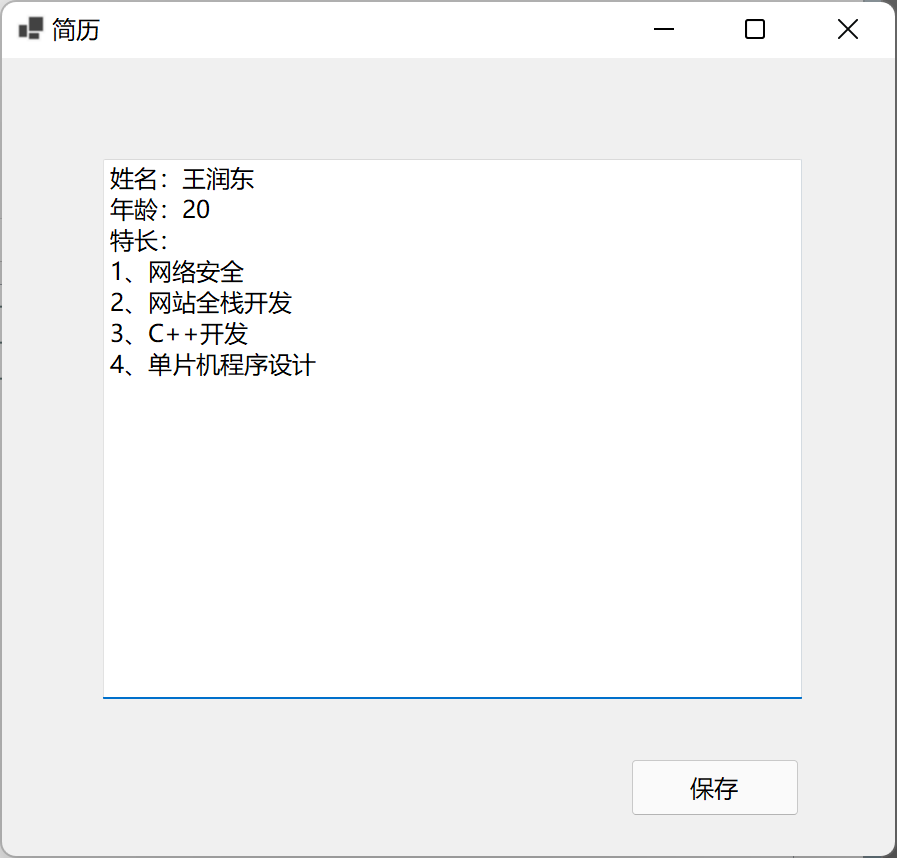
该页面针对求职者，可以查询目前的招聘信息，拟招人数，招聘单位名称，点击查看可查询具体招聘单位需求和岗位描述。发现适合自己的点击下方的投递简历，即可把自己的简历发送给目标单位。右上角“我的简历“按钮可以维护自己的简历信息。”查看简历反馈“按钮可以查看已提交的简历单位的反馈情况（未处理、已通过、已拒绝）。

## 3.8 （求职者）简历回复情况页面



页面由求职者主页调用，可以查看自己提交简历的条目和单位的反馈情况。

## 3.9 （求职者）我的简历



该页面可以显示和维护自己的简历信息。

# 四、结果与分析

程序运行情况如期，但是在开发前期，偶现MariaDB连接timeout，数据库查询结果显示慢，提示数据库同时连接数到达上限，前期分析认为是三层嵌套SQL查询使用了三个Connection再加上云服务器网速低（4Mbps）和Base64解密造成的，于是更换两层嵌套。后期发现问题并未解决，经过分析和研究发现，与数据库建立连接的函数放在了while(reader.Read())中，重复执行10次左右，遂在三层嵌套中，最内层连接函数执行了至少20次，并未被及时使用Close()函数回收，因此造成了连接数满。优化方案：建立连接时一次性建立三个连接，后期使用reader.Close()回收上一条查询结果，在循环开始使用reader.Read()更新一条查询结果。

同时，经过多次实验研究，Base64解密耗时可以忽略，因此数据库结果查询慢应该是属于客户端网络原因。于是测试客户端网络，由于网络类型为4G LTE，平均网络延时高达70ms，最低延时40ms，最高延时460ms, 在执行多条SQL语句的时候，延时造成的影响更加明显。更换5G网络后，查询时间缩减至之前的1/4，可见客户端的网络环境是影响查询速度的重要因素。

在程序设计初期为了美化窗口，拟使用毛玻璃特效做窗口背景，但是查阅资料，aero特效为Windows Vistal / 7 独有，在Windows 8 – Windows 11中已经取消，无法使用。

# 五、改进和建议

可以设计一个表，支持超长简历内容：将简历内容转换成base64编码存储，为了打破MariaDB限制varchar最大长度21844，可以把转换之后的字符串拆分成若干段，每段包含21843个字符，添加一个order列，每段使用order列排序，可支持无限段，这样简历的最大长度就取决于数据库所在硬盘剩余空间了。

# 六、总结

## 6.1 感想

我们对于一个新接触的事物，总会和之前接触的类似事物做比较。就比如我在使用C# 编写程序的时候，总会和PHP语言作比较，里面很多概念都是不同的，比如C# 会更加严谨一些，PHP相对比较随意，包容性也强。这两种语言各有各的好处，比如PHP就不用考虑变量类型，因为字符型可以与整型直接判断大小和相等，变量在使用之前无需声明，变量类型自动判断并转换，函数高度封装，C# 使用三行代码完成的任务，PHP只需一行。在使用C# 的时候，我觉得无时不刻都在使用Encoding.GetEncoding("utf-8").GetBytes()、Convert.ToString()、Convert.ToInt32()等类型转换函数，它太需要考虑变量类型了，虽然这样比较严谨没什么不对，但是我感觉过于严谨会令程序设计者分心，无法专注于主要事务的开发，本末倒置。

在设计窗口的时候，受前端设计的影响，总是想设计一些动画、渐变、动态阴影、半透明，使用一些美观的字体，但是使用C# 制作动画过于复杂，不像前端设计一样一行代码即可实现，试了试用网络上的解决方案制作动画，大概需要20行才可以完成CSS一行代码的功能，而且功能不稳定，容易产生其他连锁性错误，说明封装程度不高，遂放弃动画制作。在设置其他字体的时候，发现设置字体之后，在屏幕缩放比率不为100%的时候，控件位置定位错误，而且窗口大小在运行时会变成设置值的三倍甚至五倍，而且无法通过设置或代码更改，查阅资料，是C# 本身的问题，它本身就不支持在不同屏幕缩放比的系统下正常运行，无解，遂放弃使用。

## 6.2心得

通过本次程序设计，我明白了，程序设计者需要站在用户角度思考问题，方便用户操作，思考用户的各种需求，面向这些需求寻找解决方案，而不是一味站在开发者角度，设计一些用户难以使用的功能。刚开始自己使用自己写出来的程序时，觉得很难用，也没有什么提示性的语句，如果我不是设计者，我不会想到这个按钮具体是干什么的，于是，我改变了设计思路，在主页上，我站在一个求职者的角度，我思考我需要这个程序给我提供什么功能，然后我站在企业的角度，设计出一些按钮和名字，后期针对这些按钮进行有目的的开发。

# 七、程序测试运行环境

1. CPU: Intel® Core™ i7-1065G7 (8 CPUs),~1.5GHz
2. 内存: 16G DDR4
3. 开发环境: Microsoft Visual Studio 2022 RC
4. .NET版本: 6.0.0 RC
5. 屏幕分辨率: 2736\*1824
6. 高分屏缩放比率: 200%
7. 系统: Windows 11 Home 21H2 ( x86\_64 )
8. 网络: 中国移动 4G LTE
9. 服务器系统: CentOS 8.3 （64位）
10. 服务器带宽: 4Mbps
11. 服务器物理地址: 北京
12. 服务器CPU: Intel® Xeon® Platinum 8269CY CPU @ 2.50GHz （2 CPUs）
13. 服务器内存: 2G
14. 服务器公网IP: 39.107.xxx.xxx
15. 服务器MariaDB监听端口: 3306